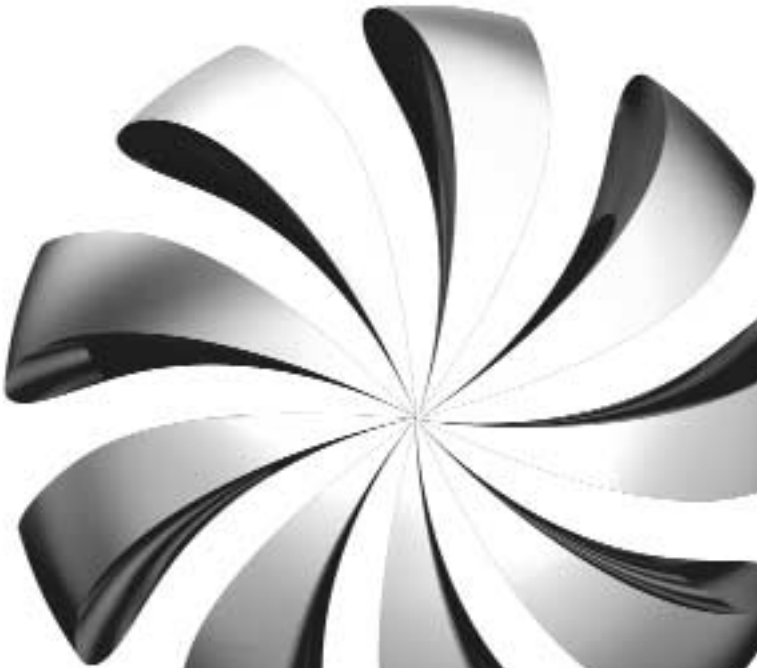
Teaching Gamecraft

C U R R I C U L U M

The theory, planning, art, production, and design of video games

Lane Daughtry
John Gabriel
Ryan Greene
Jason MacCoy
Rick McCann
Anthony Rossano



MESMER

Teaching Gamecraft
The theory, planning, art, production, and design of video games

ISBN: 0-9707530-6-3
Printed in the United States of America
First Printing: July, 2003

Authors:
Lane Daughtry
John Gabriel
Ryan Greene
Jason MacCoy
Rick McCann
Anthony Rossano

CONTACT INFORMATION

Mesmer Inc. provides the content of the book as is, and makes no warranties regarding the accuracy or completeness of the material within it. That having been said, we welcome your feedback. Please tell us how we can make this book better and better!

Questions of a technical nature, for instance those regarding software installation or hardware configuration, will not be answered.

You may email us at: info@mesmer.com

You may write to us at:

Mesmer Animation Labs
1116 NW 54th Street
Seattle WA 98107

You may call us at: 206.782.8004

Please check out our other wonderful course offerings, onsite classes, and distance learning at http://www.mesmer.com.

DIGITAL CONTENT FOR THIS BOOK

All the scene files, models, and other digital information used in this book may be downloaded free of charge from http://www.mesmer.com/

The file package is called gamecraft.zip, and is located in the File Bank on the Mesmer web site. Either scroll through the list of files to locate gamecraft.zip, or do a search for Courseware.

AUTHOR BIOS

Lane Daughtry has worked as a Maya instructor at Mesmer Animation Labs, Technical Director at Escape Factory Games, and many digital artist roles on several personal mod projects. He's been working with 3D art packages for the last eight years and counts Maya and Lightwave as being near and dear to his heart.

John Gabriel is a Chicagoan trapped in the body of a Seattleite. After graduation with a BFA from the University of Georgia studying under Rocky Sapp, John moved to Los Angeles for continued graduate level 3D research. Post LA, John moved to Seattle Washington, accepting a position with Monolith Productions before moving to his current gig as 3D animation Instructor at Lake Washington Technical College. He now lives on a nice little hobby farm in the foothills of the Cascade Mountains in Washington State.

Ryan Greene currently works as a 3d artist for Microsoft Game Studios, and has played a key role in art creation for Flight Simulator, Combat Flight Simulator, and Train Simulator. He is a Certified Discreet Instructor for 3d studio max and Character Studio. He is the author of 3ds max Illuminated: Foundation, and teaches classes at Mesmer. He has trained game artists from Microsoft Game Studios, Nintendo, KnowWonder, and several other studios. Before working in games, Ryan had worked as a 3d artist for TV and video production. He is a graduate of the University of California at Davis, and the School of Communication Arts. When he is not creating artwork or training, he trains in Filipino Martial Arts, and spends too much time playing shooter games on the internet.

Jason MacCoy previously worked as a 3D artist for Sierra Online creating concept art, character designs, 3D models, animations, level designs and storyboards. After Sierra Online, Jason has pursued his love for teaching, conducting classes in 3D Max, Maya, and Drawing for 3D production. Currently Jason is teaching at University of Washington, Lake Washington Tech, and Seattle Central. His 3D classes include modeling, texture mapping, and animation. The drawing classes cover figure drawing, storyboarding, and character design. Jason believes that the true potential for gaming is education and that the current entertainment video games of today are laying the foundation for the interactive media of tomorrow. urrently, he is working on ways of integrating more interactive computer animation into the educational experience of learning.

Rick McCann is a driven professional who maintains a high standard of excellence in his work. As a Maya instructor, he makes it his goal to impart each of his students with the entirety of his knowledge. As an artist, Rick is consumed by creative ingenuity and constantly pushes himself to attain a level of skill that reflects his dedication to the arts. This desire has allowed for the creation of a wide variety of both professional learning and entertainment material. Recent notable highlights of Rick's career include the co-creation and illustration of a published role playing game, The Riddle of Steel, and an educational DVD on mental ray. Currently, Rick is responsible for the education of rapidly expanding Maya user population, and spends his days teaching and creating learning material at Mesmer Animation Labs.

After receiving his Bachelor of Arts in Psychology from the University of Washington, **Anthony Rossano** went to work for the Microsoft Corporation. In 1989 Anthony Rossano deserted his post at Microsoft to found Mesmer, Inc. As a Producer at Mesmer, he combined his love of how the human mind works with his passion for computer technology, and headed up a team that produced interactive media content for Microsoft, Delta Airlines, Sierra Online, Softimage, Edmark, AT&T, REI, and many others. Anthony has lately overseen the development of digital books, web-based learning, DVD and e-commerce projects, as well as this courseware. Anthony is the author of Inside Softimage 3D, published by New Riders Publications, and XSI Illuminated: Foundation, and XSI: Illuminated Character, published by Mesmer Press.

Acknowledgements

Anthony says: "I want to thank the State of Washington for providing us with the grant that paid for all this fabulous learning material. I want to thank the other authors – John, Jason, Lane, Rick, and Ryan – for the *incredible* effort they put out, on a short time line, with meager resources. The sweat, tears, and brain power they put into making this courseware is an inspiration to me."

Rick says: "I thank all those who came before me and contributed to my knowledge and understanding: Lane Daughtry, Theron Benson, Paul Lewis, David Choi, Tony White, Tony Gale, and all the students I've taught who have asked questions. Thanks also to the team at Mesmer: Karen Zinker, Nancy Davidson, Brian Demong, and Anthony Rossano. Thanks to Alias|Wavefront for producing Maya. And lastly, thanks to my wife, Jaimy McCann, whose contributions to my life are endless."

Jason says: "I would like to thank my wife Heather for all of her insight and support through my transition from working in the industry into teaching. Also, thanks to my students, who teach me something new every day."

Ryan says: "I'd like to say thanks to my wife Stephenie, who put up with me working on yet another book, the folks at Mesmer, and all the artists whom I've worked with over the course of my animation career."

John says: "I would like to thank my wife Rhonda for all her help. I would like to leave you with something important: the best Pizza joints in America.

#1 Uno's/Due's Chicago location only
#2 Pizzeria Regina Boston
#3 Barnaby's Chicago
#4 Lou Malnatti's Chicago
#5 Saviano's Seattle
#6 Ray's Famous Pizza NY

Lane says: "I'd like to take this moment to remind everyone in my life just how precious they really are. Without you all in my life my days on this planet would be empty and grey. I want to specifically thank my mom and dad for their amazing restraint during my teenage years, I want to thank God, and His right hand man Jesus Christ, for absolutely everything, and of course the big guy, Anthony Rossano, for giving me the opportunities he did. Oh yeah - and also the many fine people who have worked so hard on my tattoos."

INTRODUCTION FROM THE PRODUCER

It was a dark and stormy night one winter 2002 when Frank Agnello stopped by Mesmer for a chat. He had just come up from San Francisco. He was looking for something to do, and I had some ideas about applying for State curriculum grants. Together we hunted down the opportunities, and settled on a 'Skills Standard Implementation' grant proposal. There was a generic state Skills Standard for Digital Media Specialist, and we wanted to expand it to specifically cover the process of Video Game creation. To make a long story short, we won the grant and brought the resulting grant funds to Lake Washington Technical College which became Mesmer's partner on the project. Frank moved to work directly for LWTC, and we got busy on the project. We created an industry panel of heavy hitting games experts to guide the process and help us determine what we should be doing. Matt Ontiveros, then of EA, Marina Fish of Game House, Suzanne Kaufman of Suckerpunch, James Gwertzman of Escape Factory, Paul Lewis of Outcast, David Choi then of Humongous, and Jay Balakrishna were early members of the evolving board.

With the input of the board we decided on a course of action. We would create a series of curriculum units that schools could use as a foundation to build a successful games program on top of. We would stay away from specific software development and programming content (leaving that, perhaps, for a second grant).

We would start with an analysis of the art of games, the history of the movement, and a description of games categories. Then we would cover games preproduction, planning, and art direction. Next we would split out and build courseware modules for two different 3D modeling and animation programs, Alias|Wavefront Maya and Discreet 3DS Max. Finally we would integrate the whole in a class on the Unreal Editor and the Unreal Tournament 2003 engine. I was leading the project development, so the task of building the content team fell to me. I went around like Ackroyd and Belushi in 'The Blues Brothers', or Lee Marvin in 'The Dirty Dozen', tapping the guys I knew were the best and explaining to them how important this project was to all of us, to education, in fact, to the free world as we know it today.

Graciously, Lane Daughtry, Ryan Greene, Rick McCann, Jason MacCoy, and John Gabriel agreed to join me as authors, and we set to work. It is said of politics that, like making sausage, it's best not to know too much about what goes into it. Perhaps the same is true here. The important part is that the authors put out tremendous energy, performed above and beyond the call of duty, and poured heart and soul into the process. Nine months later, the sausage was made and you have it now in your hands.

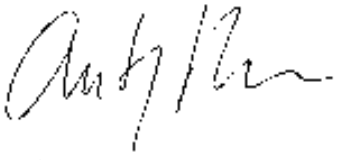
We hope this courseware will be of use to you in structuring your classes. Please do with it what you will - feel free to use all of it, or just some of it. Don't feel obligated to use the material in the order presented. You can certainly use just the quizzes if you wish, or just the course outlines. Feel free to re-order lessons, shuffle pages, or even translate the work into Swahili. The important part is to use it however it helps you to make your games program better.

There are five sections: 1) Philosophy and Taxonomy of Games, 2) Game Preproduction, 3) 3DS Max for Games, 4) AW Maya for Games, and 5) Level Design with Unreal. Each of the five sections has a topic list you may manipulate into a course outline, a detailed content section, quizzes, and projects. You may choose to lecture from the detailed project section. The projects are designed to teach by action, and provide the students with step-by-step instructions to follow while the instructor catches his or her breath. There are even guidelines included for assessing the student work on the projects, and quiz answers with explanations!

Your students will need academic copies of either the Max or Maya software, and a private copy of Unreal Tournament 2003 to do the exercises and projects. There are course materials, images, scene files and Unreal assets that accompany the classes, and these may be downloaded from the Mesmer file bank under Courseware (http://www.mesmer.com/) We also reference text books and classroom resources in the course outlines. Those resources are detailed in the beginning of each chapter.

We sincerely seek your feedback. Since this was a one time grant project, we cannot promise to update the book, but if you find areas that need correction or improvement, please let me know just in case.

Enjoy!

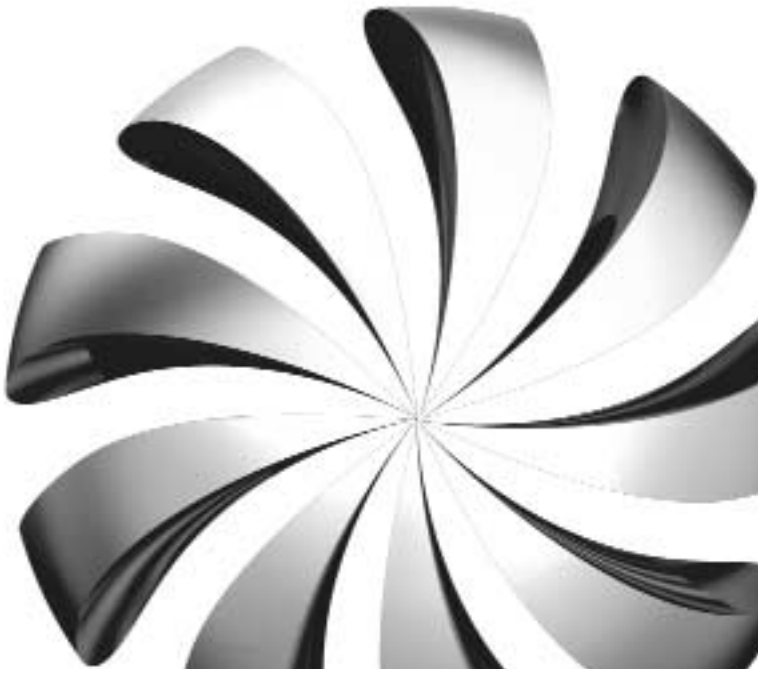


Anthony Rossano

Table of Contents

Philosophy and Taxonomy of Games

Anthony Rossano



MESMER

1.1 Course Overview

Course Synopsis

This course covers both the categorization of games and the motivations behind human game play. Over the three decades that Computer Based Games have been evolving, various styles have come into and out of fashion. A foundation to participation in the Computer Game business is a thorough understanding of these different game styles, and the language necessary to describe them.

An understanding of why people play games is a fundamental requirement of creating games that people want to play. The second section of this course discusses human motivation to play games of differing types, and puts forth several theories of explanation.

In-class game play for learning and demonstration can be used to augment lectures.

Course Prerequisites

There are no academic requirements prerequisite to this course; however, students should have played video games before, and be ready to express opinions regarding their experiences.

Course Outcomes

After completion of course content, students will be able to converse knowledgeably with game professionals concerning the many different genres of Computer Based Games, properly categorize new games into a taxonometric system, and opine on the human motivation driving game play.

Course Contact Hours

Instructor Led: 40 hours
Lab and Project Time: 4 Hours

Software Required for Course

No Software is Required

Media Accompanying Class

Instructor may refer to http://www.klov.com/ (the killer list of videogames) for visual reference

Required Books for Class

The Medium of the Videogame, by Mark J.P. Wolfe, published by Univ. of Texas Press, 2002, ISBN: 029279150X
Game Design: The Art and Business of Creating Games, by Bob Bates, Prima Tech Press, ISBN# 0761531653

Bibliography

The History of Computing Project: http://www.thocp.net/software/games/games.htm

1.2 Course Outline

Taxonomy of Games

- The Genesis of the Computer Game
- Foreword to The Medium of the Video Game (Baer)
- Reading: The Video Game as a Medium (Wolf)
- Games through the Film Scope
- Monographs on Games
  - Play it Again, Pac Man (TMOTVG)
  - Archetypes on Acid (TMOTVG)
  - The Changing Face of Videogames
- Taxonomy based on technology and hardware
  - Mainframes – 1965 – 1974
  - Arcade Machines
  - Home PCs
  - Handheld Games
  - Consoles
- A Technology Based Taxonomy
- A Player Based Taxonomy
- A Point of View based Taxonomy
- Taxonomy of Interactive and Non-interactive games
- A Genre Based Taxonomy of games

Philosophy of Games

- Human Motivations
  - Entertainment vs. Escapism
  - Hunt and Kill
  - Identity Issues
  - Relationship building
  - Group status and Identity
  - Community
  - Belonging
  - Power over Environment
  - Collecting things
  - Saving Life versus Taking life
  - Dexterity and Practice: A fitness Display?
  - Urgency, Stress, Anxiety and Thrill, Peril
  - Sports Games
  - The Role of Statistics
  - The Need for Speed
  - Transcending Physical Limits
  - The Impact of Task Length
  - The Impact of Feedback
  - Time versus Points: The High Score and the Token economy
  - Computers versus People, People versus People
  - What's in Game Play?
  - Goals
  - Structure and Progression
- Why do People Make games?
- The Role of Morality in Games

1.3 Course Content

Taxonomy of Games

A taxonomy is described by dictionary.com as:

1. The classification of organisms in an ordered system that indicates natural relationships.
2. The science, laws, or principles of classification; systematics.
3. Division into ordered groups or categories

A taxonomy is a set of assumptions held in common by a group of people to facilitate language regarding a specific topic. Through ordered language we may express abstract ideas, and with the same set of words to describe our ideas, experiences and memories, we can negotiate a conversation with other people who share the same taxonomy.

When people share taxonomy and language they can communicate not just what is and what was, but also what could be, and what will be built.

Traditional taxonomies, like the classification of matter into Solid, Liquid, Gas and Plasma, or the Phylogenic taxonomy of organisms, usually suggest that elements may belong to only one category, and that there are categories that are exclusive of other categories. This seems to be done by establishing tests and creating arbitrary category names based on results. For instance:

'Gives Birth to Live Young, Has Hair, Nurses, No Pouch' = Mammal.

Games provide a challenge to that idea. There are many different ways to classify games, and some overlap. Maybe in the future we'll create arbitrary names for categories and assign divisions, like 'First Person Perspective, Collects Points, Timed' = Contest.

Until then, we can explore different taxonomies, and blend them as we need to.

The Genesis of the Computer Game

In this section of the course the student should pick up a little background information on thinking critically about games. People are not yet accustomed to analyzing games the same way they might analyze literature, or film. The monographs listed below are several (flawed) takes on games as a medium.

As a teacher, you should be using this section to get the students warmed up to thinking about games critically, and drawing conclusions about how games differ and are the same. There are a few nuggets in the readings about why people play games – students should begin to consider that question as well.

Extra credit goes to students who think about who makes games, and why.

Foreword to The Medium of the Video Game (Baer)

Baer's fascinating (if-self aggrandizing) foreword instructs the reader that the idea of using electro-mechanical devices for entertainment is much older than the Nintendo Super-NES.

Other questions for discussion are: What is a 'video game'? Does it require a CPU? A screen? An electronic display? Is Mechanical feedback good enough? Is pre-created display content enough? What are the implications of pioneering patents on interactive entertainment?

Reading: The Video Game as a Medium (Wolf)

In trying to describe the 'Medium' of the video game, Wolf quickly finds that he must define what exactly video games actually are. He quickly gives up at that pursuit, and shifts attention to a historical description of what games have come along, differentiated primarily by the physical aspects of the game machines themselves.

Questions for student discussion could include: What does Wolf mean by medium? How is the medium of the game different than the medium of TV, given that they both show on Television screens?

How important is the physical form factor of the game machine (Sit down, standup, console, PC) to the category of medium?

Games through the Film Scope

Because so little scholarly work has been penned regarding games as a legitimate movement or art form, and since so much has been written on Film as movement and art, many students of film (and producers of film) have been compelled to try their hand at describing the medium of the game in terms of film metaphor. While they are mostly wide of the mark, this practice of conceptualizing and abstracting the medium is a helpful thought exercise.

Students might start a discussion of Games as Film here:

Why are film people compelled to describe games? Why are film and games so often compared as media (and as sales figures)? Why were so many people convinced that film and games could co-exist? Is it better to make a game from a film, or a film from a game? (Author's note: this may be moot soon, as many entertainment projects are considered as both from the start of the project – see Matrix Reloaded/Enter the Matrix as an example.)

Do games threaten film and the film industry?

Space (TMOTVG)
Time (TMOTVG)
Narrative (TMOTVG)
Genre (TMOTVG)
Character (GD:ABCG)

Monographs on Games

Play it Again, Pac Man (TMOTVG)

Bernstein's attempt to apply gonzo journalistic style to describing the effect of games on pop culture is flat. While his jangling syntax pretends the ephemera of hipness, his disordered thoughts ultimately confuse more than enlighten. Students will need to search around the potholes in his prose to mine nuggets of social truth. What does gaming do for adolescence? How will our pop culture change as a result of games? How do game protagonists look different from Film protagonists? (Author's note: Certainly they are more often women, and certainly they shoot more people…)

Archetypes on Acid (TMOTVG)

Ms. Tews introduces two areas of interest in this monograph: games in the context of cognitive science and behaviorism, and basic human archetypes in games. As with the other monographs in this book, Tews throws out a lot of specious psychobabble, and students need to understand they must filter her work to separate the wheat from the chaff. Of particular novelty is Tews' explanation of the third person perspective in Tomb Raider as "allowing the male player to hide once again behind a protective woman as he formerly hid behind his mother."

Students might profitably discuss the limited material on the cognitive science of video games, and why there is so little of it.

Can games affect thinking? Or do they just reflect it? Will games move from young men to old women, just as they moved out of the arcade and into the home? Are there limits to the attraction of video games? What were the initial demographics of Television, and how did the patterns of television consumption change over the first 50 years, and could this inform us on the path games might take?

On the topic of archetypes, Tews mentions only a few. Students could look at works by other authors (Shakespeare? Kippling? Stephen J. Cannell?) for ideas on what other archetypes to look for in games.

The Changing Face of Videogames

Now we can start building several different possible categorizations of Video Games. Ultimately, we'll be able to test a given game against these categorizations to determine where in the taxonomy tree that game belongs.

Taxonomy based on technology and hardware

In many ways, the evolution of games has been described by the evolution of technology. Each generation of game makers works within the confines of what technology is available to them, and what they think will be commercially available within a period of time equal to a game development cycle (1 to 4 years). In this breakdown, the technology is defined by the hardware it runs on rather than the sophistication of the code itself.

One truism is that as time passes, the cost of computing power goes down. The corollary to that first law is that as time passes the median computing horsepower purchasable per dollar spent increases.

As the cost of a computing unit declines, more people can acquire that unit.

The result is an evolution towards ever greater numbers of computing devices that can be used for games.

In our taxonomy, one question that will describe a game is the hardware it runs on. The question would be "What does it play on?" We would answer: "It's a mainframe game" or "it's a handheld game".

Mainframes — 1965 — 1974

Mainframes were so expensive that only very large corporations and some government agencies could have them laying around. They tended to be cared for by a small cadre of hard-core geeks, most of whom were programmers, and wanted to spend as much time with these amazing machines as possible. Since using the machines for entertainment could delay the inevitable return home at the end of the day, and because programming is a challenge, the development of games was probably inevitable.

Typical Game: Space War. Space war was played on a 2D CRT display (the author had the privilege of playing on a TI vector-graphics display as well) and was similar in concept to Asteroids. Rudimentary physics slowed the ship.

Representative Members of this Set:

MULTICS (1968) and SPACE TRAVEL (SPACE WAR) first written
(http://livinginternet.com/?i/iw_unix_dev.htm) The First Computer Game!
PDP- 7 - 1970 (MULTICS becomes Uniplexed Information & Computing Service (UNICS) perhaps just because Ken Thompson wanted to run Space War on a PDP-7. This may indicate that games actually influenced the whole landscape of computing, as UNICS became UNIX, begetting IRIX, SOLARIS, VMS, Linux, BSD, WinNT and many others)
Dec Vax 11/780 – Lots of text based games
IBM System 360 – 1977 -

ARCADE MACHINES

These machines were purpose built computer systems designed usually to play just one game, and fitted with custom controls for that game. For instance, Centipede had a fire button and a track ball, Asteroids had just four buttons, Robotron had two joysticks.

These machines are usually played in public spaces, and are coin operated, or use credit type cards.

Representative members of this set:

Stargate
Asteroids
Street Fighter
Karate
Pac Man
Donkey Kong
Pole Position

HOME PCS

Home built PCs were certainly more available than the Mainframes, so there were many more games. However, by today's standards a very tiny percentage of the population ended up buying (let alone using) a Personal Computer, until the IBM PC came out.

Now that there were people to trade games with, the medium of transferring games became an issue. Early home systems hit on the idea of using an acoustic coupler and a simple standard cassette tape deck. Now games were portable and transferable!

Representative Members of this Set:

Altair 8800 (1974)
TRS-80 Models 1 and 3 – 1976 (8080 based)
Apple 1 – 1977 (6502 based) A very popular game machine
Commodore VIC-20
IBM PC and descendants
Apple Macintosh and Descendants

HANDHELD GAMES

Handheld computer games became the sales solution to the problem of making money from computer games. These small, cheap units were often huge commercial successes. The limitations of the hardware (cheap, small screens with limited display resolution) simplify the games that work well on them. The success of these games proves that simple, small games running on slow hardware are still competitive. Look to these games as precursors of what we are starting to see on modern cell phones.

Representative Members of this Set:

1970 The first LCD handheld computer game is released by NINTENDO.
1976 Mattel Electronics introduces a LED-based hand-held electronic game platform. The platform can support different games, each is sold as a unit. The football game becomes incredibly popular with 10 – 20 year old males, despite the display consisting of less than 20 Red LEDs and some green lines painted on top of the plastic. These sell very cheaply. Both Football and Baseball were reissued by Mattel in 2000.
1979 Microvision (Milton Bradley) released, with interchangeable games on cartridges. It has a 16*16 monochrome display.
1989 Nintendo Gameboy (100 million units sold! The GameBoy, GameBoy Pocket, GameBoy Color, and the new GameBoy Advance.

PDA machines: The Newton, The Palm, others
3G Cell phones from Sony/Ericsson, Motorola, Nokia, others

Consoles

Consoles are personal game machines that are generally dedicated to the task of playing games. These are generally much less costly than general-purpose computers.

Representative members of this Set

1970 A hockey game is sold by Ralph Baer to Magnavox
1972 Magnavox introduces the Odyssey home video system
The Atari Home series
The 3DO Multiplayer
The Nintendo NES, Super NES, N64, GameCube
The Sony PlayStation, PS2
The Sega Genesis, Dreamcast
The Microsoft Xbox

A Technology Based Taxonomy

This part of the taxonomy represents the medium of the game and also the graphical STYLE of the games. In the categorization of the game we might ask "what kind of content was in the game?" It is also a description of the historical evolution of games, as different technologies and content styles come in and out of fashion. This might be considered a question of the game engine used.

We could ask the question "What kind of engine did the game use?" We might answer with "It was a 2D Side scroller".

Representative members of this Set

2D Space Game
Text based Adventure and Decision Game
Static screen Q&A
Side Scrolling (Left and Right)
Maze Scrolling (Left, Right, Up, Down)
Sprite Based
Film/Animation Clip Based
2.5D (Faux 3D, still Sprite based)
3D Environment

A player- based Taxonomy

Reading: Character in Game: player identification? (GD:ABCG)

This part of the taxonomy concerns the player or players in the game. How many are there? Do they take turns? And what is their representation in the game? Are they omniscient observers or present participants? Can other players see them?

Representative members of this Set

Single Player
Turn based Multiplayer
Simultaneous split screen Multiplayer
Single Screen LAN Multiplayer
Direct connect Multiplayer

Massively Multiplayer (internet or proprietary network)

A POINT OF VIEW BASED TAXONOMY

This is one of the most obvious and evident ways to categorize a game. It overlaps frequently with the technological categorization of the game, since so frequently the purpose of new technology development is to enhance the player experience through point of view.

Representative members of this Set

Omniscient Observer / Fly on Wall
First Person
Third Person

TAXONOMY OF INTERACTIVE AND NON-INTERACTIVE GAMES

Believe it or not, not all games have been the interactive first person type we see so much of today. Different game types explore different methods of entertainment, and have corresponding different lengths and playability. Zork, Dragon's Lair, Myst, Resident Evil, and Doom are all examples of different levels of interaction.

Representative members of this Set

Immersive Decision Based
Story Driven Decision Based
Limited Interaction
Freely Interactive (Real Time)

A GENRE BASED TAXONOMY OF GAMES

Imitation is the sincerest form of flattery. So one good (successful) game usually sprouts legions of imitators. When a sufficient group of similar games has developed, a genre exists. Usually genre doesn't rely on technology. In fact, technology innovations drive further development of genres. Genre describes a set of characteristics that bind games together, including goal, gameplay style, control method, setting, player identification and others.

As games evolve, new genres emerge.

Representative members of this Set

Space games
Shooters
God games
Driving/racing games
Sports games
    Hockey Games
    Soccer Games
    Baseball Games
    Football Games
    Skate/Bike Stunt games
Role playing games
Sims
Flight Simulators
Fighting Games
Contest Games
Platform Games
Maze Games

Exploration Games
Strategy and Resource Games
Education and Training Games
Casual Games and Puzzles

Philosophy of Games

Why do people play games?

Central to understanding what games are is why people make them and play them. Good games tap into some deeper motivation that existed already within us, and allow us to express that motivation. Good games allow us to leave the confines of our own bodies, personalities, societies, morality, and much more.

You can use this list to start discussions of human motivation, and what games they might be expressed in.

Human Motivations

Entertainment vs. Escapism

Simply leaving reality is a human desire. To what degree is all entertainment escapism? Are games more or less escapist than, say, Film?

Hunt and Kill

Do humans still harbor the instinct to hunt prey? Certainly we retain the ability to be excited by being hunted. Which games most successfully explore the elevated metabolism and excitement of being on either side of the hunt?

What is the role of "scaring" in games (Quake leaps to mind).

Identity Issues

In most real time games the player actually becomes transmogrified into a different character on screen. Sometimes this character is an animal, sometimes a burly warrior. Sometimes a pretty woman! How important to game play is the transformation of the player?

Relationship building

Humans have innate and hardwired functions that are expressed in groups, clans and societies. Many games have exploited issues of cooperation, social contract, membership and belonging.

Group status and Identity

In a game, the player can become a new individual. The worth of that individual is often quantified in the game, with points, magic, or status. What human motivations play in this? Are games a second chance for low status players in the real world?

Community

What does the size of the community have to do with game play? What is the optimal size for clan cooperation?

Belonging

Often in group and network games, players can adversely affect members of their own clan. How do social norms, customs, and taboos evolve in online communities? How are they enforced? Where do online norms differ from real life norms and conventions?

POWER OVER ENVIRONMENT

Another human trait is the desire to bend the environment to his purpose. Many games seem custom built to provide the player with resources to command and tools to build with. Discuss these urges, and the games that enable them:

Creativity and Creation: Age of Empires, Black & White

Building Things: SimCity, SimEarth, SimAnt, the Sims, etc..

Constructing a Fantasy: Ultima, Zork, Warcraft

The urge to Problem Solve: Myst, Ico

Strategy: Railroad Tycoon, Starcraft, The Art of War

COLLECTING THINGS

Maybe the bag-lady impulse to collect things and carry them around exists in all of us to varying degrees. It's certainly true that a great many games feature collecting and carrying elements that might prove useful later in the game.

What is the different between collecting potions, spells, armor, and the like, and just collecting stars, coins, or points? Is there one?

Sonic the Hedgehog vs. Gauntlet

SAVING LIFE VERSUS TAKING LIFE

In William's classic Defender (and Stargate) the player has two continuous goals that must be balanced: killing all the enemies (the Yllabians and Iratas) while rescuing and saving the humanoids.

In Donkey Kong (and many other games), saving the Princess is the ultimate reward and goal. Why do these ideas motivate humans? Is this evidence of an altruism gene in Human Nature?

DEXTERITY AND PRACTICE: A FITNESS DISPLAY?

In the early years of the game revolution, much was made of the hand/eye coordination issue. Young people were training to process visual feedback, making decisions and letting delegating actions to varying parts of the human wetware system.

To what degree is dexterity itself a motivation? Do people play dexterity games for the same reasons they juggle? Is a speedy nervous system a fitness display?

Or are dexterity and practice games compelling for the same reason that people with Obsessive Compulsive Disorder (OCD) do little things over and over?

Tetris, Robotron 2084, Super Monkey Ball

URGENCY, STRESS, ANXIETY AND THRILL, PERIL

Thrill seeking is a common human behavior. Many people experience elevated blood pressure and heart rate as pleasurable. Increasing hormones related to fight or flight reflexes often give the player a sense of power and strength. Hormones often enhance performance.

Do games fill a need to exercise our limbic system? Our adrenal glands?

Doom, Quake, Space Hulk, Mortal Kombat, Tomb Raider

Sports Games

Sports games are a tremendously popular genre. Why? Why do people play sports, why do they watch sports on TV, and is playing a sports game somewhere in between the two?

The Role of Statistics

Many sports games rely on a player knowledge of the virtual actor and their strengths and weaknesses. Many players compulsively memorize statistics. Is playing a known statistical character against someone else's also known statistical player similar to trading cards?

The Need for Speed

People like the sensations of travelling fast, flying, leaping and running. Games can provide these without the usually attendant effort and danger portions. How much of this is a factor in different genres of games, like motorcycle racing (such as Road Rash) and flying games (Afterburner, F16 Eagle, The Red Baron, etc), as opposed to games where the point of view is usually stationary, like Myst and Riven?

Transcending Physical Limits

Beyond the physical desire to zoom through space, how much of gaming is transcending the limits of the player's physical body?

The Impact of Task Length

Different genres of games feature different lengths of game play, and are broken down sometimes into different phases of game play, such as getting to the castle, and then fighting the Boss. While players will in general spend very long periods of time in one sitting working to solve game problems, the length of each task is important. If the tasks take too long, the player becomes tired or less interested. Task completion is a form of positive reinforcement.

What are different forms of task and the differences in time taken in different genres?

Contrast Sonic the Hedgehog, with Pole Position, with Riven, with Legend of Zelda.

The Impact of Feedback

Feedback can be obvious, such as the entire environment moving when a 3D first person game is played. It can be subtle, like a slow change in the environment from day to night (Star Wars Rogue Squadron 2)

Sometimes feedback is minimized to reduce player visibility.

Some games arguably had little or no feedback at all, like Dragons Lair, since the player sees the same thing each time.

Time versus Points: The High Score and the Token economy

A common feature of arcade games was the option for the high scorers to add their names or secret identities to a list that the game displayed in public in between games.

How much do you think that pride motivates the player to work hard to achieve the top spot? Does rewarding the player with a high score reinforce behavior?

A token economy is one where players trade for a currency that is not easily convertible in the rest of the world. Generally player points stay in the game, but recently changes have made it possible for players to compete with each other in national competitions (Golden Tee) and even have economies where the currency of the virtual realm is convertible into American dollars (EverQuest).

How are points and magic, etc., less real than dollars? Or are they as real?

COMPUTERS VERSUS PEOPLE, PEOPLE VERSUS PEOPLE

First computer games almost always pitted the human player against an array of computer-driven opponents. As games progress, more frequently players are competing against other human players. What does this say about how engaging interaction with other people is? Why are people more engaging than computers that follow simpler rules? Are humans hardcoded with tools to decipher and predict the actions of other people? Is that what makes competing against other humans fun, or is it just less meaningful to beat the computer than to win against a human?

WHAT S IN GAME PLAY?

Gameplay integrates answers to all the philosophy questions, and expresses them in the genre categories. (GD:ABCG)

GOALS

Discuss defining hierarchies of goals in games. For instance, staying alive comes before ending the level,,which comes before accumulating points, which comes before doing the level with a good time, which comes before solving all puzzles, which comes before finishing game.

Goals can also be as simple as getting the chaingun or hitting a power-up. Goals in driving games are the most straightforward – pass checkpoints en route to a finish line.

STRUCTURE AND PROGRESSION

Some games are broken into levels where there is little structural difference (Robotron). Other games have constantly evolving thread of change, like Legend of Zelda. In some games the progression is the reflection of the story (the Star Wars Vector Arcade game). In some games the structure involves rapid shifts among different whole genres, like the Tron stand-up arcade game.

When are progressions to different game styles needed?

WHY DO PEOPLE MAKE GAMES?

This is a topic for another curriculum, but you can ask your class to consider why they think pioneers like Nolan Bushnell (Atari) or John Romero (id) bother to make games. Is it just because flipping burgers doesn't pay as well (barring bankruptcy)? Why them? What is it about someone that makes them a potential game maker?

THE ROLE OF MORALITY IN GAMES

Videogames have been controversial since they came on the public scene in America in the 1970s. The general form that the controversy takes is this: Parents and Lawmakers become concerned about the influence that games might have on the youth of America, then parent-driven public interest groups raise media awareness, the press gets excited, politicians make speeches condemning games, the story fades out, and the cycle repeats again.

Usually some innovation in game design or game play drives the outrage of the parent groups. When Mortal Kombat was released, the adult version featured special combination Fatality moves, including pulling a player's spine out of his body and shaking it at the crowd. This sparked considerable dismay.

When Quake was released, many people were concerned by the dark and sometimes occult symbols in the game, and the generally creepy feel.

The biggest crisis to date occurred after the release of Grand Theft Auto 3, a game in which the player becomes a street hood, and works his way up the organized crime hierarchy. With a name that included the word "theft", as well as ad billboards showing virtual hookers mounted on city buses all over America, parents could no longer claim to not know what kids (and adults) were buying and playing.

The resulting uproar included talk show condemnation and the passage of the first legislation criminalizing the sales of video games to minors, in the state of Washington.

What role does public morality have on the game business?

What social norms are violated in games? Which are the norms that make parents and legislators so upset?

How does sex and violence in games compare to TV and Film? Does the passivity of just watching violence on TV absolve the viewer in a way not possible for an involved game player? What evidence is there linking game play and actual social misanthropy?

Some cultures (notably Japan) have a different game culture that emphasizes bright colors and cute gameplay. Why are games here and there so different?

1.4 Quizzes and Projects

Project 1: Describe and Defend

The teacher assigns three (known) games from www.klov.com to each student (these can be the same for each student, or different). Students write a short paper of 2 to 4 pages, categorizing the games, and defending the choices of categorization.

Set a specific due date, and delivery medium and specification, like "word doc, 2- 4 pages, single spaced, starting with one paragraph executive overview".

Assessment

Student papers should be judged on these criteria at a minimum, teacher may add more!

1) Was paper delivered on time and to specifications?
2) Spelling errors present?
3) Does the teacher agree with the student classification?
4) Is the student argument cogent? (Even if the teacher does not agree with classification, a good argument is worth something).
5) Was the description complete, or just enough to get by?

Project 2: Invention

Each student writes a paper describing a new and innovative game idea. The game description should include a discussion of what market segment the game appeals to, and why. The motivations of the gamers, the rewards and punishments of the game, the structure of game play, and other aspects of the human motivations exploited by this game must be included in the paper.

Assessment

Student papers should be judged on these criteria at a minimum, teacher may add more!

1) Was paper delivered on time and to specifications?
2) Spelling errors present?
3) Was the game idea original? If it sounds too much like an existing game, dock points.
4) Would the game find a publisher? In other words, is it marketable?
5) Could the game be played casually with a minimum of start-up time and learning curve?
6) Does the game tap aspects of human nature, like violence, altruism, collecting things, flying?
7) Did the student describe the game in a way that sounds fun?
8) Did the student identify a target niche market of people?

Multiple Choice Final Exam

A list of game titles is presented, each with multiple choice options that describe different category aspects. Students must choose most appropriate category for game. Teacher may use this list, or present his/her own, based on in class discussions.

Questions:

1) SpaceWars
   a. Shooter
   b. Physics
   c. Mainframe

2) Doom
   a. Shooter
   b. Strategy
   c. Platform

3) EverQuest
   a. Shooter
   b. Strategy
   c. Platform

4) Tomb Raider
   a. Shooter
   b. Education
   c. Sim

5) Disks of Tron
   a. Shooter
   b. Strategy
   c. Platform

6) Street Fighter
   a. Side Scroller
   b. 3D
   c. First Person

7) Stargate
   a. Single Player
   b. Split Screen Multiplayer
   c. Network Multiplayer

8) Dragon's Lair
   a. Omniscient Observer
   b. First Person
   c. Third Person

9) Myst
   a. Immersive Decision Based
   b. Limited Interaction
   c. Real Time

10) Mortal Kombat
   a. Fighting Game
   b. Role Playing Game
   c. Shooter

ANSWERS

1) c. Space Wars was the FIRST video game, and ran on a Mainframe. It used physics, but physics is an attribute of many games (almost all…) so not a taxonometric distinction.

2) a. Doom was a very early and very influential entry into the first person, real time 3D (2.5D) shooter genre.

3) b. EverQuest isn't a shooter, and jumping from platform to platform isn't a big part. It's mainly a Role Playing Strategy game.

4) a. Tomb Raider is mostly a shooter.

5) c. Disks of Tron was an innovative game combining a contents with leaping from disk to disk… therefore an early 3D Platform game.

6) a. Street Fighter is a 2D sprite based Side Scroller.

7) a. Only one person at a time may play Stargate.

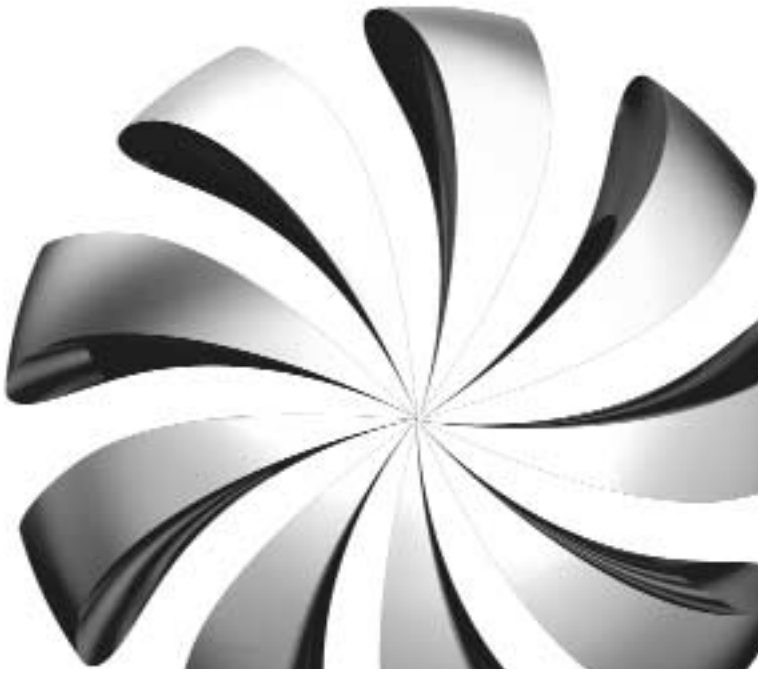
8) a. Dragon's Lair is a Video-disc based animated clip game. The player observes the action from various vantage points.

9) a. Myst (and Riven as well) is not a real time game, and there are no interactive sections. Players make choices with clicks.

10) a. Mortal Kombat is an early 2D sprite based fighting game (and a good special effects movie based on it).

Video Game Preproduction

Jason MacCoy,
John Gabriel



MESMER

2.1 Course Overview

Course Synopsis

The purpose of this course is to develop skills and experience related to the importance of preproduction in the creating of a successful project.

This course integrates with the other areas of Game Development because it will lay the blue print for what is created, how it should look and in what sequence it is to be produced.

Items covered will include:

1. Understanding the fundamentals and aspects of preproduction in the overall project cycle
2. Working on a specific game level project
    a. Level design
    b. Character development
    c. Concept art
    d. Storyboarding
3. Learning how to use preproduction content for actual production
4. Final evaluation of preproduction content

Each one of these areas will be discussed and explained in detail

Course Prerequisites

No prerequisites

Course Outcomes

Students will gain skills in drawing, project planning, researching references, character design, and storyboarding and general concept development.

A talented student would be capable of working for various game companies creating concept art, storyboards and character design.

Course Contact Hours

The estimated time to complete this module will take approximately 200 hours.
You can divide the module into two phases:
The first phase will be a research and planning phase. (75 hours)
The second phase will be a production phase (the phase where you create you deliverables such as drawings, models and concept art. (125 hours)

Software Required for Course

Photoshop 7, Power point, High speed Internet access, Microsoft Word.

Media Accompanying Class:

Shrek DVD, to show elements of preproduction
Scene file of character turn around used for modeling
Slides of storyboards possibly presented as an animatic
Painted character designs examples

REQUIRED BOOKS FOR CLASS

Photoshop Illuminated: For Animators, by Jaimy McCann, 2003, published by Mesmer Press, ISBN: 0-9707530-3-9

BIBLIOGRAPHY

Film Directing Shot by Shot, by Steven D Katz, published by Michael Wiese Productions, ISBN #0-941188-10-8

Storyboards: Motion in Art, by Mark Simon, published by Focal Press, ISBN# 0-240-80374-4

Digital Cinematography and Directing by Dan Ablan, published by New Riders, ISBN# 0-7357-1258-1

The Art of 3D Computer Animation and Imaging, by Isaac Victor Kerlow, John Wiley and Sons Inc, ISBN# 0-471-36004-X

Maya 2 Character Animation, by Nathan Vogel, Sherri Sheridan, Tim Coleman, published by New Riders (pages: 1-101)

Game Design: The Art and Business of Creating Games, by Bob Bates, published by Premier Press, ISBN# 0761531653

Gardner's Guide to Animation Scriptwriting: The Writer's Road Map, by Marilyn Webber, Nic Banks, Bonney Ford, published by Garth Gardner Co, ISBN# 0966107594

Maya Illuminated: Games, Lane Daughtry, published by Mesmer Press 2003, ISBN: 0970753012

2.2 Course Content

Lecture: Pre-Production fundamentals

Style considerations

This is the game's look and feel ranging from artistic style to audio and interface design.

Realistic

Is the game's environment similar to ours? Are the characters lifelike or fantasy?

Genre style

What is the core game play:

A. Strategy?
B. Puzzle?
C. Constructive story?
D. Sports?
E. Simulation?
F. Action?
G. Adventure?

Unique personal style

The game reflects a particular artist's personal style and flair.

Game play features

Artificial intelligence

Artificial intelligence or AI is driven by the game's code written in C++ or JAVA (mainly C++). This code enables your characters to interact with the other elements in the game's level; i.e. Non-player characters, environmental elements, villains etc.

Saving

Does the game let you save throughout the game or do you have to start over every time that you play?

Keyboard functions

How do the keys affect the player character? Is it easy to learn how to control the character with the keys or does it take time to be able to be proficient at controlling your character with the keys?

Using the interface

Interface design deals with usability and the ergonomics of the computer screen. How is the Interface laid out? Does the interface display a great deal of information on the screen or does the viewer have to summon the icons with the keyboard? What is the Visual metaphor of the interface?

Manual for game

This document or small booklet gives directions on how to play the game, tips on game play, as well as the background story of the game's plot.

Interactivity style

How will the player interface with the game environment?

Free camera

The camera in the game moves automatically to suit the player's best possible vantage point.

CONTROLLED CAMERA
The player can control the camera.

FIRST PERSON
The camera is set to give the impression that the player is "looking through the eyes" of the character they are currently controlling

THIRD PERSON
The camera is set to give the impression that the player is a few feet behind the character they are currently controlling

CINEMATIC
Small pre-rendered movies that play within a game. Often these movies are used to tell the plot of the game's story line or used to entice reward the player.

IN GAME
A Cinematic that is intertwined with the regular game play.

PRE RENDERED
A Cinematic that is unable to be interacted with.

CUT SCENES
Cinematic plays as a reward for player after reaching a certain milestone within the game

INTRODUCTIONS
A Cinematic that plays at the beginning of the game to set-up the story line or plot of the game

THE WHOLE PRODUCTION CYCLE

HOW PREPRODUCTION FITS INTO TOTAL PRODUCTION CYCLE

ORIGINATION AND DESIGN
Coming up with the idea
Brainstorming alone or with 3 to 7 people.
Fleshing out some details
Talking and drawing through character, visuals and scenarios of the game and game play.
Deciding on themes and look
Researching and/or developing the look of the characters and environments within the game as well as look of the interface.

TARGET MARKET FOR PROJECT
Who are they? Age, gender, skill with technology.
What do they want?
What type of Genre do they want to play?

ORIGINALITY VS. PREDICTABILITY?
Is the game Unique or does it play and look just like every other game?

USER S TECHNOLOGY?
What are the hardware specifications of the user? What type of games will he/she be able to play based upon his/her computer hardware?

PREPRODUCTION

Researching
Using reference material or experts to familiarize yourself with the game's content
Defining actual details
Draw and articulate what the model will specifically look like.
Producing usable content
Produce model sheets and turnarounds for 3d artists to use for production
Trying out variations
Creating different versions of the same art to see what works and looks best

PRODUCTION

Utilizing Preproduction
Distributing concept art, design documents and any related creative material to staff
Making the content
Fabricating the 3D, 2D art and animations on related software
Exporting to engine editor
Taking 3D, 2D art and animations and exporting the data to a game engine (code that runs the game usually made in C++). (The games programmer facilitates usually exporting to a game engine).

TESTING

IF THINGS WORK

Game does not crash
Characters interact with game an environment as expected
Game engine works as expected

IF THINGS LOOK GOOD

Models, animations, and environments look as expected

IF GAME IS FUN TO PLAY

Levels load in a timely manner
Game play makes sense to player
Story line is well written and easy to understand.
Interactivity is workable for player

RELEASE

Finishing touches
Packaging is designed
Last minute changes
Any bugs in the game are fixed
Pieces of art, programming or audio are taken out or added as needed.

SCHEDULING

MILESTONES

SETTING GOALS

A series of deadlines are established to make the game's workload more organized and attainable. Usually this involves a three part deadline series as follows:

a. ALPHA (mostly design/pre-production work and organization at this point

b. BETA (The game (art animations, environments, programming) is 80% completed. Game is playable to an extent. Testing of game begins

c. GOLD or Golden Master: The game is ready to be shipped to the distributor for mass production, if there are any minor changes to the game they are made at this time.

SHOWING YOUR PROGRESS

At these Milestones meetings and walk throughs are held to determine how well the game's progress is going.

BEING REASONABLE

Technical Artists are asked to devote a great deal of time on a video game production. Furthermore, speed and skill are expected of these artists. The client, directors, project mangers and artists all must work together to overcome obstacles and time issues in a manner that is acceptable of all parties.

TESTING

SEEING IF THINGS WORK

The game's content and art is constantly being tested and scrutinized before it enters the level. Art Directors and programmers work together to make sure artwork will be compatible with the game level.

FIXING PROBLEMS

Artwork is fixed or changed by the artist then tested in the game once again

SCRAPPING IDEAS THAT CAUSE PROBLEMS

Many times due to artistic, time-related, or technical reasons, artwork is often scrapped.

UNEXPECTED

Planning time for the unknown

Extra time is allotted for any unexpected mistakes or upgrades made in the game. This time varies depending on the scope of the project.

DECISION MAKING

Often times it is necessary to make difficult decisions in the midst of a game project i.e.: Reassigning an artist to another part of the level or firing staff. Also the game's size and scope may be reduced to fit certain needs.

RESOURCES FOR PRODUCTION

PEOPLE

Ability
Raw talent
Experience
Knowledge
Personality
Availability
Times of year
Industry conditions
In house or outsource

MONEY

Business considerations
Budgets
Salaries
Equipment
Responsibilities
Agreements

TIME

Marketing considerations
- Holiday sales
- Competitors release dates
- Worldly events
- Anniversaries

Project related, how long for each?
- Origination and Design
- Preproduction
- Production
- Testing
- Release

TECHNOLOGY

What software programs you have
Image editing, like Photoshop
3D animation, ex. Maya, 3ds max, Softimage, Lightwave.
Plugins
Custom programs for specific uses
Hardware abilities/limitations
Types of hardware

Identifying the unknown

Identify what you need to find answers to
Having a list of questions to find answers to
Doing research
    Web searches
    Library
    Videos
    Experts
Asking the right questions
Talking to the right people
Using Chat rooms and message boards

Technical limitations

Polygon count
Platforms
Current and future technology, release date

Texture mapping aspects

How large of pixel size
Bump mapping
Alpha channels
Reflection maps
New/Other

Frames per second

Animation considerations
Cut scenes

Managing resources

Naming conventions

Categories
Using numeration
Alphabetization

Storage conventions

File formats
File sizes
Locating

Sharing content

Checking stuff out
Building up ideas
Getting ideas rejected

RESEARCHING AND REFERENCE IMAGES

Web search
Search engines
Search by images only
Advanced web searching

LIBRARY BOOKS

Free information
Scanning materials from book

LOCAL BOOKSTORE

Getting the latest books
Learn from the professionals
Specific books

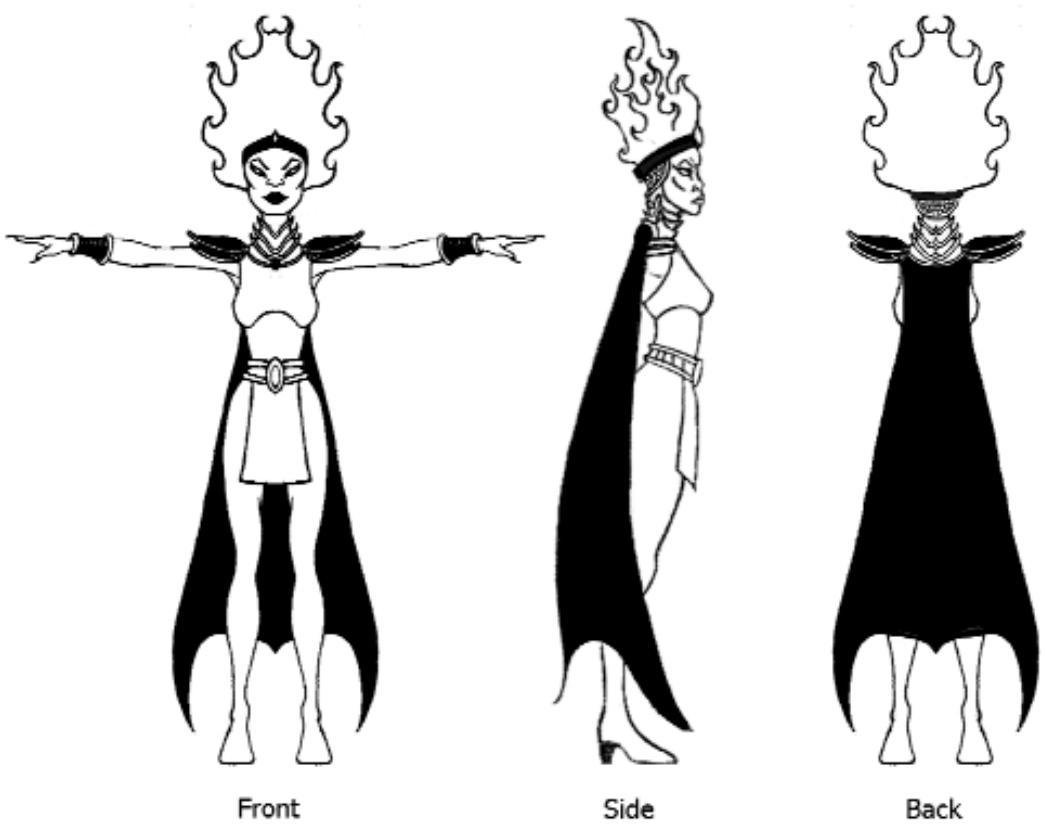
DIGITAL PHOTOGRAPHS

Take your own images
Using images with Photoshop

CONCLUSION

In this course students have experienced creating the types of content needed to make a common 3D game level. The student has learned how to make: a game level schematic, character turn arounds, and concept art for buildings and scene objects. The experience of creating game content for a 3D game level is good training for working in the industry and will create content that the student could use in a future portfolio.

2.3 Quizzes and Projects



This is an example of a character turn around for the villainess.

Project: Game Level Design

General overview

In this lesson, you will learn about:

Creating actual 3D game preproduction art work including:
Creating Character turn arounds, Creating concept art, and Designing a game level schematic.
Designing a game level schematic.
Creating Character turn arounds
Creating concept art

Introduction

In this chapter you will experience creating the types of content needed to make a common 3D game level. The student will learn how to make: a Game level schematic, Character turn arounds, and Concept art for buildings and scene objects. The experience of creating game content for a 3D game level is good training for working in the industry and will create content that the student could use in a future portfolio.

This is an example of concept art used to design the villain's castle.

CREATING A GAME LEVEL SCHEMATIC

Before one starts to model the 3D terrain it is important that a schematic is created to help determine where strategic game play elements will be located. The job of the level designer is to create good looking game environments that are fun and interesting to play. By placing opponents, obstacles and rewards in the environment the level designer can create a level that will challenge the player. The schematic is the first step in the preprocession process to create the level. Like a thumbnail sketch it can be changed and improved to incorporate the best ideas.

This is an example of a simple game level schematic.

TOPICS COVERED

In this tutorial you will learn how to make a basic game level schematic including tactical placement of game play elements.

YOU LL LEARN HOW TO:

The tools you will use for this tutorial will be a pencil and paper.

The concepts will include designing a game level including tactical placement of game play elements.

RESUME RELATED EXPERIENCE

Doing this project will give students experience working the same kind of game content that professionals work on

POSSIBLE TEAM COLLABORATION

If students work together on a level they will experience that advantages and challenges of working collaboratively on a project.

INDUSTRY RELATED CONTENT

Character turn arounds, level designs and concept art are all common items found in game production. Thus, this exercise will create relevant portfolio material.

OVERVIEW: DRAW THE GAME LEVEL SCHEMATIC

Create a large square area for you game level to be drawn in.

Draw a river going through the middle of your paper

On one side of the river create the hero/heroine's village.

On the other side draw where the villain's castle/dwelling is.

Draw where the bridge goes over the river.

Place tactical objects including: placing rocks that provide partial cover from someone shooting a bow, place trees, bushes and rocks that the character and monsters have to move around.

DESCRIPTION OF PROJECT

Terrain schematic mapped on grid

Have the students create an image of the layout from a top down view, this is called a schematic and can be used to show you where buildings, landscape features, characters and game play elements will be. Later this schematic can be used to map a subdivided grid of geometry and provide a guide for modeling the terrain and placing various types of geometry.

CHARACTER TURN AROUND FOR MODELING

This is a drawing of the character from the front, side and back. It defines the character's proportions, outfit and general look from different angles. The turn around can also be used in the character modeling process

The following website shows the use of the character turn around in the 3D modeling process.

www.3dtotal.com/ffa/tutorials/max/joanofarc/body1.aspCharacter turn around painted for texture mapping

Once the character turn around is drawn, it can be scanned in and painted in Photoshop. This painted turn around can be use later as a guide when you are painting the texture map for the 3D character.

This website is a good reference for painting line drawings in Photoshop

www.agent44.net/stepbystep1.htm

www.agent44.net/stepbystep2.htmConcept art of objects and structures used for modeling, texture painting

Draw scene objects from different angles to define what they will look like when they are modeled out in 3D.

This website has some good drawings of concept art.

www.primal-soft.com/screenshots2001conceptart.shtml

CHARACTER TURN AROUND EXERCISE

Draw a male and a female; pick one to be the hero/ heroine, and the other to be the villain/villaness. Draw each character from the front, side and back. Draw the characters with feet spread slightly apart and hands down to the sides. Make sure that they are the same sides in all views. This will help for 3D modeling later on. A good website to see examples of character turn arounds is: www.fineart.sk/character/page_01.htm

MALE

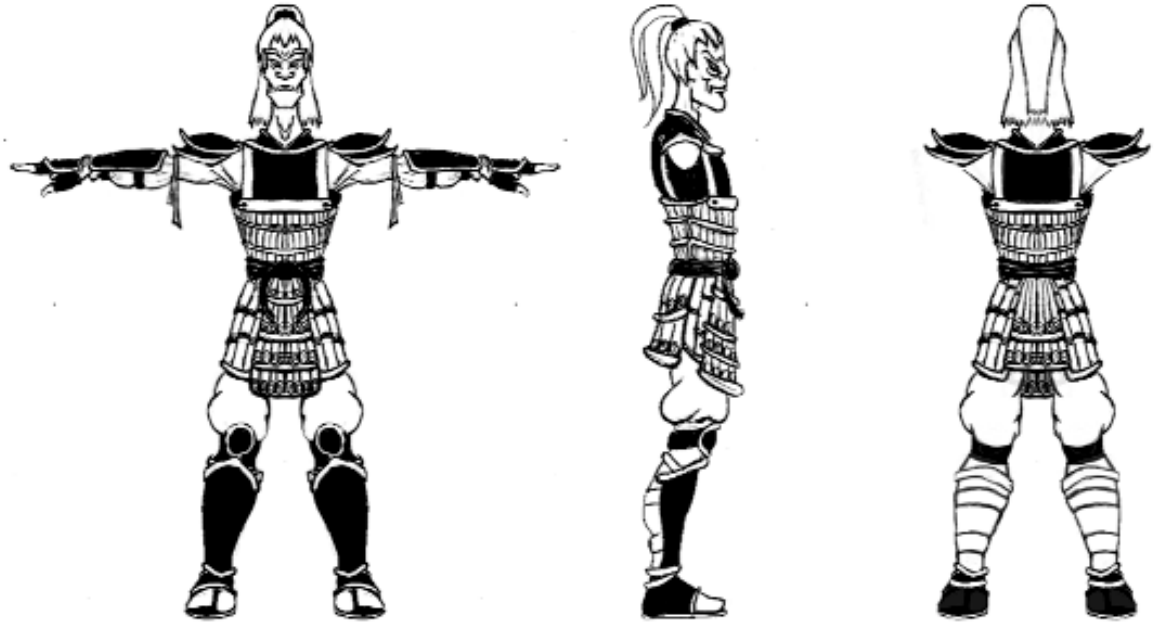
Draw a character turn around of the male, front, side and back

FEMALE

Draw a character turn around of the female, front, side and back

MONSTER

Draw a character turn around of the monster, front, side and back



This is the Hero, in this turn around I have tried to define the armor carefully for the modeling..

Concept art

Here is a good website for examples of concept art: www.primal-soft.com/screenshots2001conceptart.shtml

Village huts

Draw concept drawing of this item

Do a web search and find reference images for this subject

Villain s place

Draw concept drawing of this item

Do a web search and find reference images for this subject

Bridge

Draw concept drawing of this item

Plan out a place on the bridge where a "game play" element will happen, for example, if you step on a certain stone that section of the bridge will collapse and the character will fall into the chasm below, or if you move too close to a statue on the side, fire will come out. These game play aspects make an environment interesting and more exciting to design

Do a web search and find reference images for this subject

ENVIRONMENT OBJECTS

Trees

Draw concept drawing of this item
Create 3 different types for variety
Do a web search and find reference images for this subject

Rock

Draw concept drawing of this item
Create 3 different types for variety
Do a web search and find reference images for this subject

Bushes

Draw concept drawing of this item
Create 3 different types for variety
Do a web search and find reference images for this subject

Other objects misc.

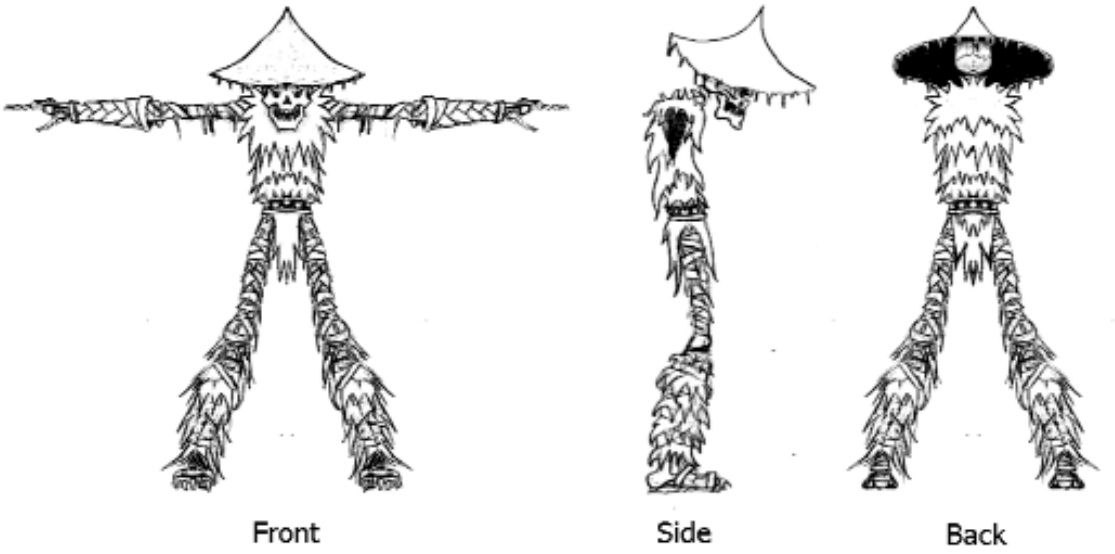
Draw concept drawing of this item
If another type of object would add to the environment then create concept art for it too
Do a web search and find reference images for this subject

ENVIRONMENTAL SCENES

General landscape

To get an idea of what the overall environment may look like, you can do drawings of the entire scene

Chasm under bridge



The Skeleton archer is drawn to show details of his bandages and proportions.

This drawing is another posed variation of the skeleton character as a boatman with a pole. Drawing variations is a good way to experiment with character design and try out new ideas.

This drawing shows the character posed and lets us know a little bit about the personality of the character, how they may move and how they may stand.

This hut is drawn to look like it is from an ancient village. Natural materials are used for the roof and walls to give it an ancient feel.

This is the area where the character or monster will fall if they fall off the bridge, you could make it a river, a spiked pit or a lava flow as examples.

Final evaluation of Student Work

Game design art project preproduction, total points will depend on length of class. Total of 100 points. For extra points instructor can add points for drawing scene objects like: trees, rocks, and other items. 10 points each.

Make sure final preproduction content relates to original goals

True to intended style?
Relates to needed content?
Functions to help with actual production?

Criteria for grading:

1. Character's anatomy must line up in height in all three views. 20 points

Character must be drawn from all three views. 10 points

Character should not be posed, arms and legs should be slightly out to the sides to help with 3D modeling. 10 points

Character turn around should be painted to help with texture mapping. 10 points

2. Create a drawing of at least one village hut. 10 points

Criteria for grading:

Student must use two-point perspective so that the drawing shows at least two sides of the building. 10 points

3. Create a drawing of the bridge that spans the chasm. 20 points

CRITERIA FOR GRADING:
Drawing must incorporate a strategic game play element like a stone that will shoot fire when stepped on. 10 points

Student must use two-point perspective so that the drawing shows at least two sides of the bridge. 10 points

4. Create a drawing of the villain's castle/dwelling. 20 points

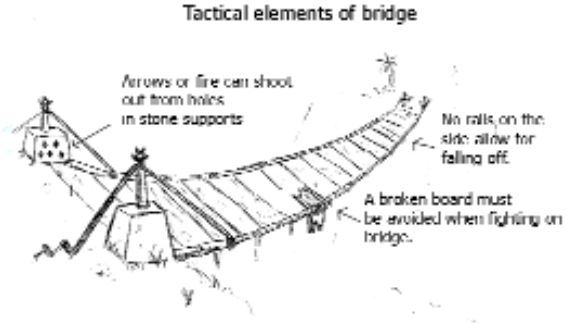
CRITERIA FOR GRADING:
Student must use two-point perspective so that the drawing shows at least two sides of the building. 10 points

5. The dwelling should communicate a general sense of evil. 10 points

6. Additional objects

CRITERIA FOR GRADING:
Additional scene objects should clearly look like the object they are supposed to be. This would include accurate proportions and including the key elements of the object. 10 points

This is a second variation on the hut design. A real village would have more than one type of hut so it is good to try out a couple different versions. Once again we are trying to make it look like it belongs in an ancient village.



The illustration of the bridge indicates where tactical elements may be placed, however, often times things need to be moved around or changed depending on technical limitations or game play factors.

The Villain's castle should look impressive and intimidating. This drawing shows if from a low angle making it look more powerful and dominating. Also, attempt to include cool design features, in this case, flaming smoke stacks..

Game Preproduction Quiz

1. What are the three angles a character "turn around" should be drawn from?
   a. Front, Right, Left
   b. Back , Front, Top
   c. Front, Side, Back
2. Choose the wrong answer: Why is it important to create a turnaround of a character?
   a. So that you can work out the character's costume for modeling.
   b. So you can establish the character's proportions for modeling.
   c. So you can waste time on creating artwork you will never use.
3. Why is it important to do preproduction before making 3D models?
   a. So you know what they should look like when you start making the model.
   b. To Share design ideas with decision-makers
   c. Both A and B
4. Doing preproduction can save you time in production later on.
   a. True
   b. False
   c. Sometimes.
5. Drawing skills are not important in creating character turnarounds.
   a. True
   b. False
   c. It depends
6. Milestones or deadlines are generally set up into three categories: What are those three categories?
   a. ALPHA, BETA, DELTA
   b. BETA, GOLDEN, TESTING PHASE
   c. ALPHA, BETA, GOLD
7. Siggraph is considered the "Big Convention" for the video game industry.
   a. True
   b. False
   c. Only every Other Year
8. What does E3 stand for?
   a. Enter Electronica Exposition
   b. East Edutainment Expo
   c. Electronic Entertainment Expo
9. When a game reaches a "Beta" deadline, generally, the game is playable but has a few bugs.
   a. True
   b. False
   c. Except for MS products
10. A Cinematic within a video game can be interacted with your keyboard or joystick.
    a. True
    b. False
    c. Depends

ANSWERS:

1. c. The most useful perspectives are front, side, and back. Top could be helpful as well.

2. c is Wrong: You need to design the character first. You will use the art.

3. c. Both A and B are only a few of many good reasons to draw out characters.

4. a. True. Creating preproduction lets the team know what things should look like and help keep the art styles together. Not creating preproduction can leave things ambiguous and require the team to stop production to work out details.

5. b. False. Knowledge of the human body and good art skills help create effective drawings.

6. b. ALPHA, BETA, and GOLD. Generally a game is broken down into three "smaller" deadlines commonly referred to as an ALPHA, BETA, or GOLD deadline.

7. b. False … E3 is the big gaming convention. Siggraph is generally a convention for computer graphics hardware, software, and motion picture related companies.

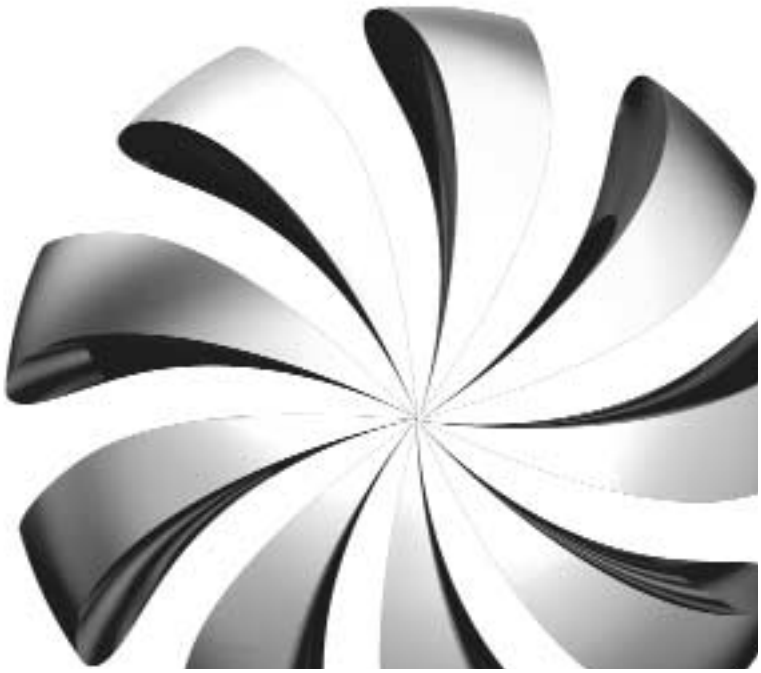
8. c. Electronic Entertainment Expo

9. a. True. Usually when a game reaches a Beta deadline the game is "playable" but has some problems. In the Beta phase game play is tested along with a myriad of other potential problems such as graphics issues, programming bugs, and hardware/software compatibility issues. Also, in the Beta deadline, game play and overall playability, or how "fun" a game is to play, is evaluated.

10. b. False…A cinematic is a pre-rendered little movie and can only be watched.

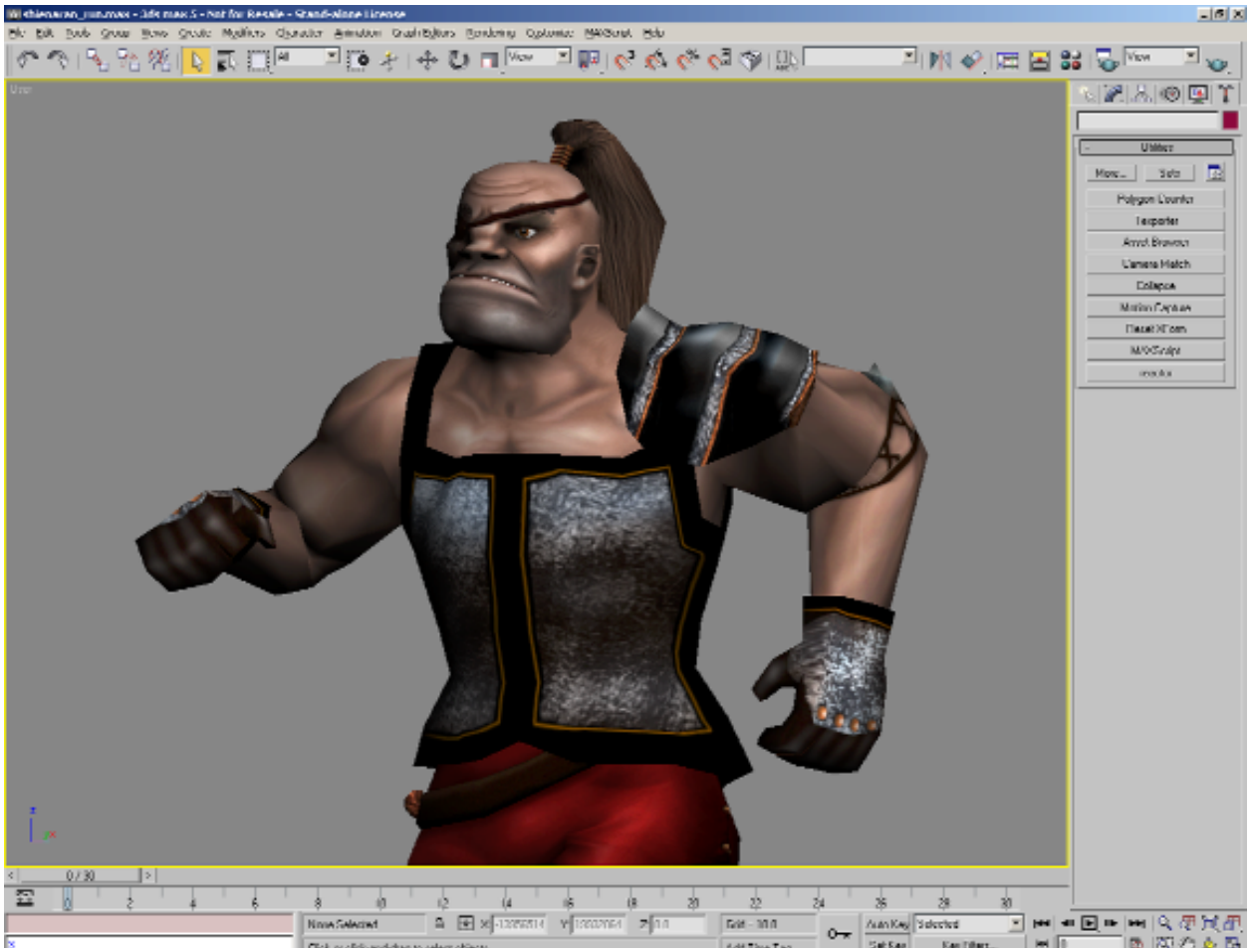
3d studio max for Games

Ryan Greene



MESMER

3.1 Course Overview



Course Synopsis

In this course, you will learn the uses of 3d Studio Max for game production. You will get acquainted with the interface, and learn the proper workflow for game art production in Max. The course will teach you how to create high quality content for games quickly and efficiently.

Various topics will be covered, including such technical aspects as modeling, texturing, animation, and more. Much of the course will be focused on these types of topics, and the more technical aspects of game art creation. An artist must understand his tools, and be able to use them efficiently to be effective in a production environment.

In addition, some theory will be introduced, which will be helpful when you must focus on designing, planning, and implementing smart solutions for content creation. Simply put, the artist must always have a plan of attack when assigned with a duty. Determining the correct approach and methodology can be a huge time saver when a task must be completed on a deadline.

OVERVIEW

COURSE PREREQUISITES

Students should have some Photoshop skills.

COURSE OUTCOMES

After completion of course content, students will be able to use:

- Good planing
- User Interface
- Selection and Transformations
- Polygon Modeling
- Modeling Strategies
- Scene Management
- Materials
- Mapping Coordinates/UV Assignments
- Lighting and Light Baking
- Animation and Keyframing
- Linked Hierarchies
- Character Animation
- Character Studio
- Exporting to a Game Engine

COURSE CONTACT HOURS

Instructor Led: 40 hours
Lab and Project Time: 20 hours

SOFTWARE REQUIRED FOR COURSE

Each student must have a PC workstation with a current version of 3DS Max installed.

MEDIA ACCOMPANYING CLASS

Scene files for demonstration and student use are provided.

REQUIRED BOOKS FOR CLASS

3ds max Illuminated: Foundation, Ryan Greene, Mesmer Press, 2002, ISBN: 0970753020

BIBLIOGRAPHY

Fundamentals and Beyond Character Studio, Pia Maffei, Autodesk Inc., 2003, ISBN: 1-5644-970-X

Animating Real-Time Game Characters, Paul Steed, Charles River Media, 2002, ISBN: 1584502703

Modeling a Character in 3DS Max, Paul Steed, Wordware Publishing, 2001, 1556228155

COURSE TOPIC SHORT LIST

1. Planning
2. UI Overview
3. Selection and Transformation
4. Polygon Modeling Basics
5. Advanced Modeling Toolsets
6. Modeling Strategies
7. Scene Management
8. Materials
9. Mapping Coordinates – UVW Assignments
10. Lighting Basics/Light Baking
11. Animation Basics
12. Linked Hierarchies
13. Character Animation
14. Character Studio
15. Exporting to a Game Engine

PROJECT LIST

1. Selection and Transformation tutorial: Chessboard - 3ds max Illuminated: Foundation, p. 15

2. Advanced Modeling tutorial: WW2 Fighter –3ds max Illuminated: Foundation, p. 55 (graded)

3. Modeling and Smoothing tutorial: Modeling an Orc Head (graded)

4. UVW Unwrap/Texture Baking tutorial: Mayan Statue (graded)

5. Animation Tutorial: Double Dribble – 3ds max Illuminated: Foundation, p. 144

6. Hierarchies, Bones, and IK Tutorial: Leg Rig (graded)

ASSESSMENT LIST

Planning/UI overview quiz: 10 points

Selection and Transformation quiz: 10 points

Polygon Modeling Basics quiz: 10 points

Advanced Modeling tutorial: WW2 Fighter –3ds max Illuminated: Foundation, p. 55: 10 points

1 point: object is named
2 points: it is an Editable Poly object
2 points: has wings
1 point: has tail
1 point: has cockpit
1 point: clean edges
2 points: Smoothing Groups Applied

Advanced Modeling Toolsets quiz: 10 points

Modeling and Smoothing tutorial: Modeling an Orc Head: 10 points

1 point: object is named
2 points: Profile is defined
2 points: facial features are defined
2 points: has ears
3 points: Good edge flow

Scene Management quiz: 10 points

Materials quiz: 10 points

UVW Unwrap/Texture Baking tutorial: Mayan Statue: 10 points

1 point: object is named
2 points: Unwrap UVW applied
1 point: UVs laid out well
1 point: Material is named
2 points: Material is applied and shown in viewport
1 point: Lighting applied to model
2 points: new baked texture created

Animation Basics quiz: 10 points

Hierarchies, Bones, and IK tutorial: Leg Rig: 10 points

1 point: bones are named
1 point: helper objects are named
1 point: correct bones in hierarchy
1 point: thigh bone has fins
1 point: HI IK solver thigh to heel
1 point: HI solver heel to toe
1 point: control splines in place
1 point: foot dummy in place
1 point: swivel dummy in place
1 point: rig functions correctly.

Hierarchies/Character Animation quiz: 10 points

3.2 Course Outline

Planning

- Know the Subject
- Design and Research
- Re-Purposing Existing Assets
- Visibility/Importance of object

UI Overview

- Menu Bar
- Main Toolbar
- Viewports
- Viewport Controls
- Command Panel
- Track Bar/Time Slider
- Status Area
- Animation Controls

Selection and Transformation

- Selection Tools
- Transforms
- Transform Gizmo
- Snaps and Type-in controls

Polygon Modeling Basics

- Modeling at the proper scale for the game engine
- Modifying Primitive Objects-Parametric Modifiers
- Using the Modifier Stack
- Edit Mesh Modifier
- Normals

Advanced Modeling Toolsets

- Editable Poly Basics
- Editable Poly Sub-Object Controls
- Splines and Patches

Modeling Strategies

- Modeling in Halves
- Putting detail where you need it
- Define the Silhouette
- Contiguous and Non-Contiguous/Segmented Meshes
- Open-Edged Models and Airtight Meshes
- Keeping the Mesh Clean
- Smoothing

Scene Management

- Naming Conventions
- Hiding Objects
- Freezing Objects

Merging Objects
Grouping Objects
Managing Polygon Counts and LODs
Cleanup and Troubleshooting a Model

MATERIALS

Material Editor basics – Material Slots
Texture Maps - Bitmaps
Navigation
Multi/Sub-Object Materials
Rendering
Configuring Paths

MAPPING COORDINATES - UVW ASSIGNMENTS

UVW Map Modifier
Unwrap UVW Modifier
Edit UVW Dialog
Working with UVs
UV Exporting-Texporter Plug-in

LIGHTING BASICS/LIGHT BAKING

Max Lighting Basics
Global Illumination
Render To Texture
Vertex Lighting

ANIMATION BASICS

Keyframing and In-Betweening concepts
Creating Keyframes
Viewing Trajectories
Keyframe editing
Expanded TrackBar (Mini Curve Editor)
Track View editing: Curve Editor
Track View: Dope Sheet
Time Configuration

LINKED HIERARCHIES

Linking vs. Grouping
Creating a Linked Hierarchies
Managing Hierarchies
Animating in Forward Kinematics
Segmented models vs. Contiguous models
Bones
Bone Tools Editing
Using Inverse Kinematics Solutions
Using Control Objects to animate IK rigs
Deforming a Mesh using the Skin Modifier

CHARACTER ANIMATION

Morph animation
Looped Cycles

CHARACTER STUDIO BASICS

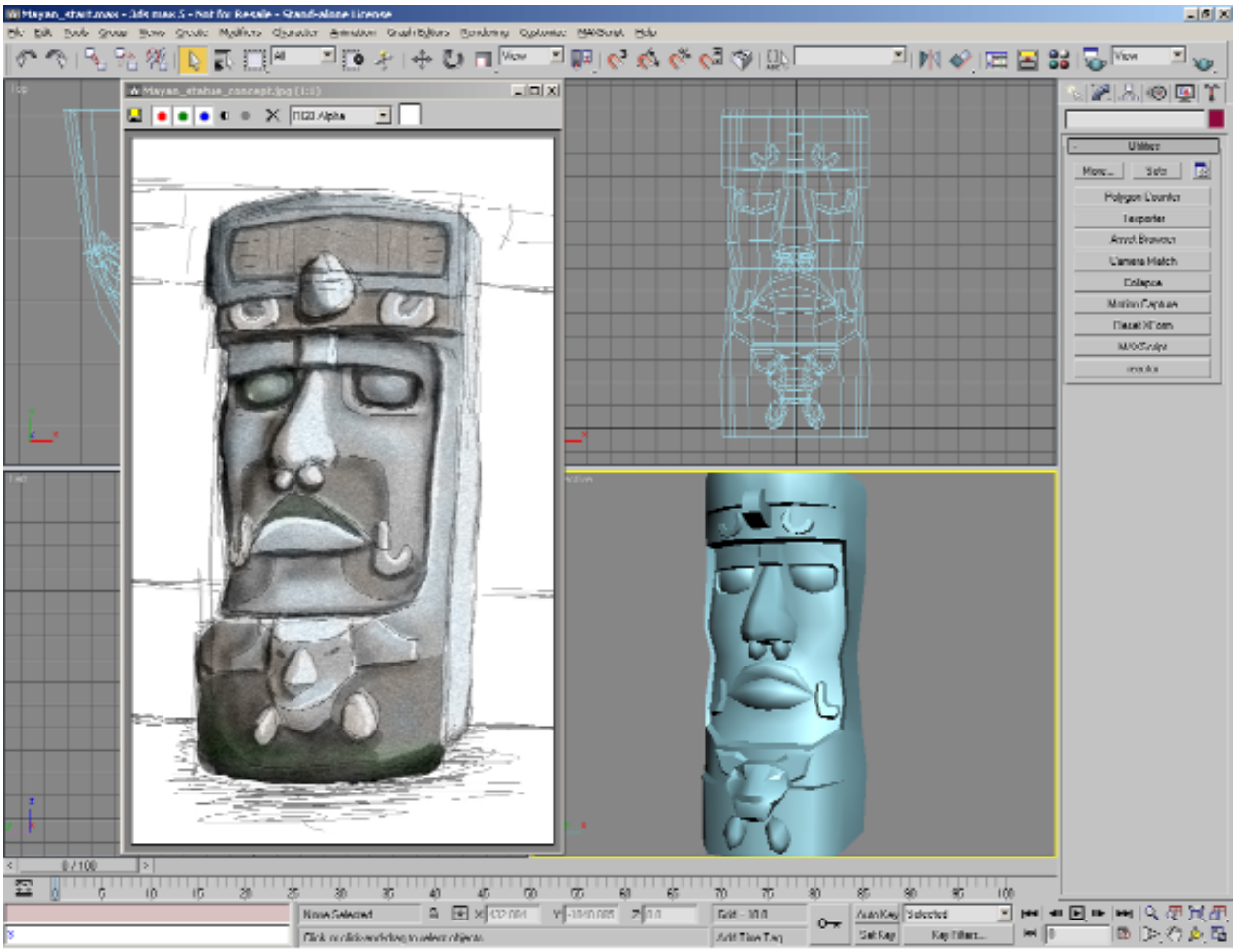
- Biped
- Footstep Animation
- Pure Freeform Animation
- Layers
- Biped Tools
- Using Motion Capture
- Mixer
- The Physique Modifier

EXPORTING TO A GAME ENGINE

- Proper naming and scale
- Scenery objects - Static Meshes
- Characters – Animated Meshes

CONCLUSION

3.3 Course Content and Lectures



Planning makes perfect

Planning

Before you do any type of job, you should first assess exactly what needs to be done, and what will be the proper approach for doing it. This is true for game artwork as well. You must remember that you will be working as part of a team, and that you must understand how your work contributes to the greater whole. Simply put, you should understand the context that your artwork will be seen in, and must plan for it accordingly.

Know the Subject

The first thing that an artist must do is to understand what it is that they need to make. This is the research and design portion of the creative process, and the importance of this aspect cannot be understated. In the case of a 3d model, you absolutely must know what the object or character is supposed to look like before they even to begin working on it in 3d. You must take into account the style of the game as a whole, and take into consideration the proportions of the model, if it will have exaggerated dimensions/features, or whether it will be more realistically built. Similarly, color palettes should be taken into account when texturing.

DESIGN AND RESEARCH

Once these aspects are taken into consideration, you should begin researching the subject. This may entail doing original conceptual design sketches, or examining concept art that is already provided. Photos can also be used as reference, for either the model as a whole, or for smaller portions of the model. The web is a great source for images, and most search engines allow image searches for specific topics. Magazines can be another great source of reference. If you need to model a muscular character for instance, you may want to use the photos in a fitness magazine for anatomical reference when creating the model. Using reference will drastically speed up the process of modeling, and is an absolute necessity if it is some real-world object that is being re-created in 3d.

RE-PURPOSING EXISTING ASSETS

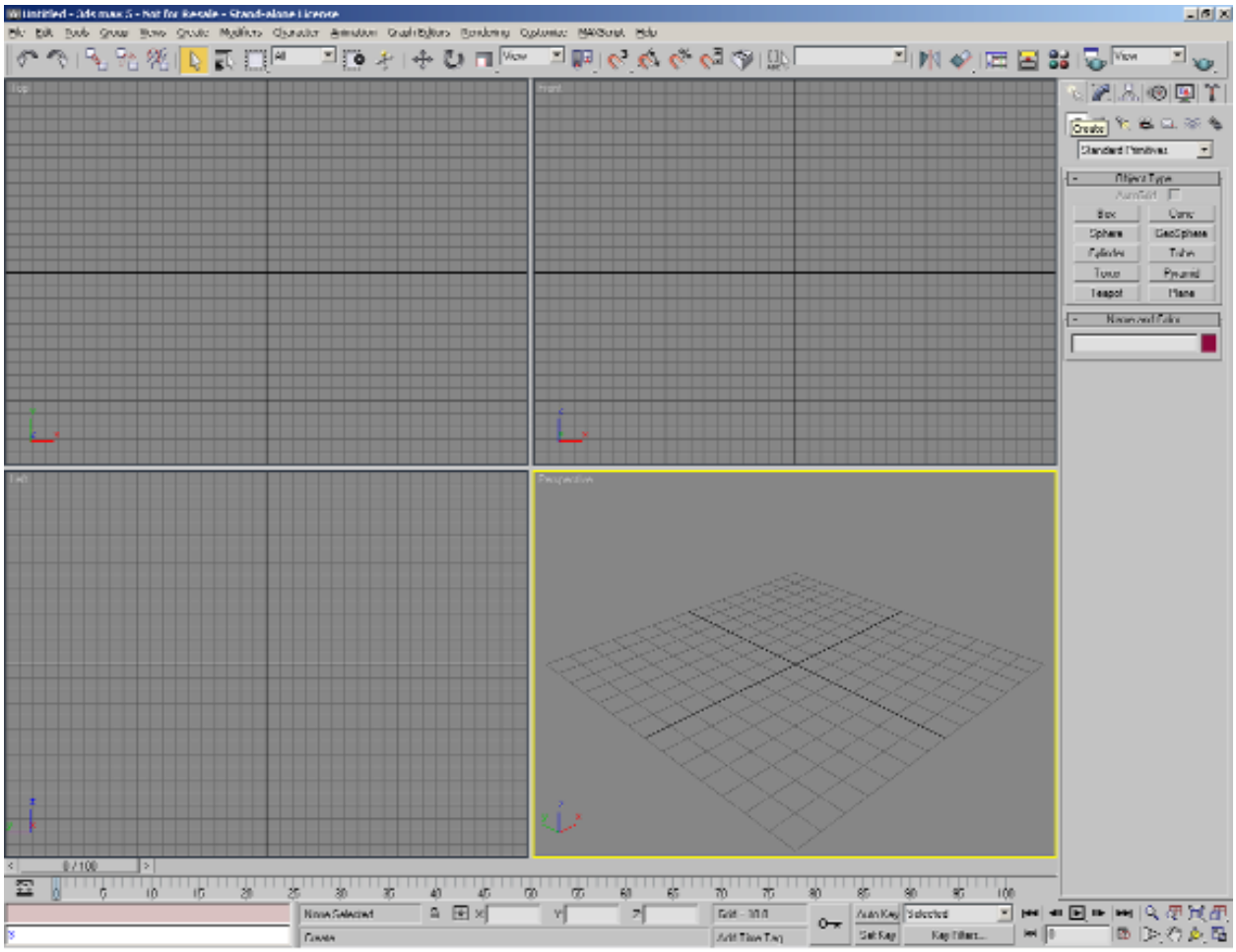
Often times, artists, or the game company itself will have a library of existing assets, which may be re-used for the project at hand. There is no point in doing redundant work when it is not called for. Even if an object or texture in the library is not exactly what is called for in the current project, it still may be customized to suit current needs, and re-saved. Sometimes altering existing assets may take a lot less time than creating art from scratch. If previously created art will be re-purposed, be aware of the fact that the use of copyrighted material can cause a lot of trouble, and should always be avoided.

VISIBILITY/IMPORTANCE OF OBJECT

You must also find out what the overall importance of the object is, and how it will be viewed. Knowing this will help to determine where detail for a model should be concentrated. Will the model be seen in the game up close, or from a distance? What angle will it usually be seen from? These all help in determining how the model should be constructed, and how textures should be applied.

Frequently, questions of polygon count, number of textures, texture sizes, etc. will be determined by an art lead. But if they are not, the artist should set a reasonable budget, taking into account the role that the model will play in the game.

UI OVERVIEW



The Interface

Suggested reading: 3ds max Illuminated: Foundation Chapter 1, pp. 1-15

You must know your tools, whether he is using paint, ink, or Max. This may not be the most glamorous part of the course, but being aware of the options available is important for efficient production.

MENU BAR

From the Menu Bar at the top of the screen, you can access nearly every command available in Max. Sometimes this is redundant, but it keeps artists from getting lost.

From the File Menu, you can open and save files, which is a standard type of function. You can also choose to create a new scene, or you can Reset the scene. This second option ensures that all options and settings in the interface are reset to their defaults. You can also Merge objects from other scenes into the one that you are currently working on. This is particularly useful when working in a team environment, when you are planning on re-purposing existing assets.

The Help Menu is actually quite good in Max. It includes brief tutorials, and a searchable index. It includes graphics, which makes sense, as Max is a graphics program.

MAIN TOOLBAR

The Main Toolbar is where you go to choose tools that allow you to interact with the scene. Selection Tools, Transforms for Move, Rotate, and Scale, and other interactive tools are available here. There are also options for Undo, Redo, Rendering, and editing tools that are available on the Main Toolbar. Many of the tools on the Main Toolbar have hotkeys assigned to the same functionality. If ever you want to know the name of a specific tool in Max, just hold your mouse over the icon for it without moving for a second, and a text description will appear.

VIEWPORTS

The Viewports are where you go to look at the objects in your scene. One viewport is always active, and is indicated by a yellow border. There are two overall types of viewports, the first being an orthographic view, the second being a Perspective view. In the Perspective view, objects that are further away appear smaller. In orthographic views such as the left, front, and top viewports, this is not the case. There is no perspective to these viewports, and objects are represented as if they were a technical drawing instead, with no indication of depth.

A grid is present in each view, which may be useful for monitoring size and proportions of a model. Whenever an object is created, it will be planted on the grid in whichever viewport is active. You will notice that there is a darker centerline down each grid, and where these meet is X=0,Y=0, Z=0 in Max world space. This is the center of the Max universe, and objects will be described in relation to this point. If an object is at X=0, Y=0, Z=10, then that object is floating 10 units above the grid, as Z corresponds to vertical height, whereas X means side lateral position, and Y is the depth.

You can resize viewports by moving your mouse between the borders, and clicking and dragging. To reset the layout, right-click on the viewport border. Other viewport options are available by right-clicking on the text label in the upper left corner of each viewport. You can change display mode from wireframe to Smooth and Highlights, the hotkey for which is F3 on the keyboard. F4 is the hotkey for Edged Faces. You can also change a viewport to a different angle by choosing Views from the viewport options. It often happens that you want to switch the left viewport to a right viewport, or something similar.

VIEWPORT CONTROLS

The viewport controls are located in the lower right corner of the interface. Some controls are only available for certain viewports. Specifically, Arc Rotate and Field of View are only available when the Perspective view is active. Also note that if there is a little mark in the lower right corner of a viewport control icon, then there are more options available if you click and hold on the button. These are known as flyouts.

For a quick run-down on what the viewport controls do, Arc-Rotate allows you to rotate around the scene, and is generally done in the perspective view. If this control is used in an orthographic view, it will switch that viewport to a User view, which is orthographic, but at an off angle. Zoom moves the virtual camera closer into the scene, or farther out. In the Perspective view, there is a similar tool, called Field of View, which functions like a telephoto lens on a stationary camera. Be careful of using this, as it can distort the perspective of the virtual camera. The Zoom Extents controls attempt to frame the scene in the viewport(s). Pan moves the virtual camera side to side, and up and down.

There are also viewport hotkeys, which can really save the artist a lot of time. These all hinge on the middle mouse button. Used alone, the middle mouse button activates Pan when depressed. Ctrl+Alt middle mouse, or the wheel roll activates Zoom. Alt+Middle mouse activates Arc Rotate. In addition, Z on the keyboard is Zoom Extents.

COMMAND PANEL

The Command Panel is a set of tabbed panels located at the right side of the screen. This is where you can go to create objects, modify them, control their motion, access utilities, and more. The Command Panel is the generic term for the entire set of command options, but each tab may be referred to as its own panel, such as the Create Panel, the Modify Panel, etc.

In any of the panels within the Command Panel, you always have the option to name the object that is currently selected. It is always a good idea to name the objects in your scene, but just remember that they must be selected to see any information about them whatsoever. Some of the commands available in the Command Panel may also be accessed from the Menu Bar, but not for naming objects.

TRACK BAR/TIME SLIDER

Just below the viewports, is an area that will be used when animating. There is a Time Slider, and a hashed and numbered TrackBar area, which keeps track of keyframe information. From the TrackBar, you can edit keyframe information, either within the visible hashed area, or by clicking the small button to the right of this area. Clicking that button will expand the TrackBar, so that animation function curves are visible. These allow more control over editing animation.

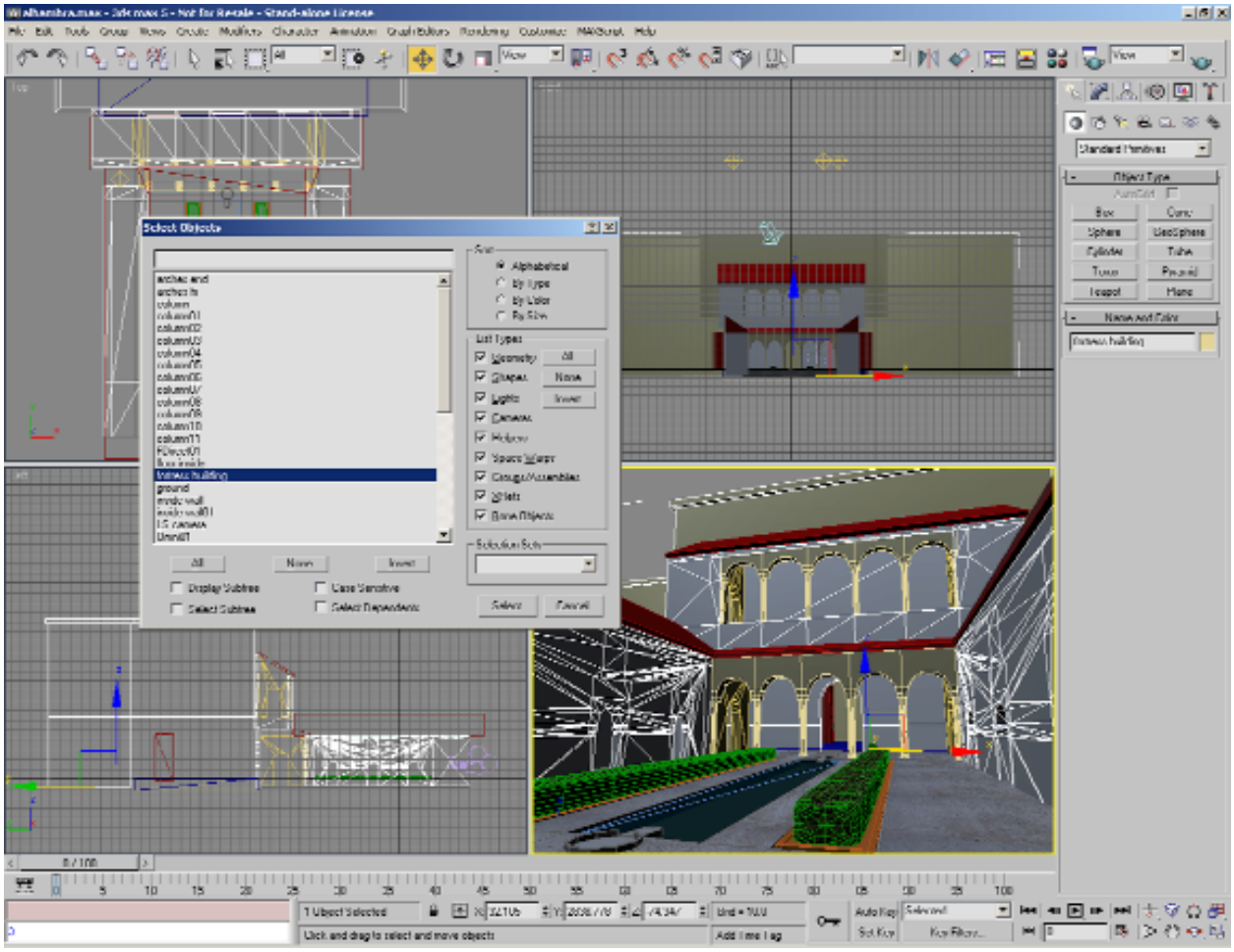
STATUS AREA

The Status area is located directly below the TrackBar, and provides you with various statistics for objects in your scene. When an object is selected, it is possible to monitor its XYZ Position, rotation, and scale information in a numeric description here.

ANIMATION CONTROLS

There are several buttons located just to the right of the status area, which allow for animation controls. There are several keyframing buttons, which allow artists to set start points and end points to animation. In addition, there are playback controls that look similar to a VCR, which allow you to play your animation, rewind, etc. There is also a control for Time Configuration, which sets the length of the animation, the playback rate, and similar options.

SELECTION AND TRANSFORMATION



Selections and Transformations

Selection skills are important any time an you need to interact with the scene that you are working on. Various tools are available on the Main Toolbar, and also from the Edit Menu which allow you to quickly select the object or objects that you want. These tools and options are useful when just selecting objects and naming them, but become even more important when you want to Transform (move, rotate or scale) them in some way. Also, when you want to modify an object, selecting the correct components is very important.

SELECTION TOOLS

The Select Tool, located on the Main Toolbar, is the primary tool that you will use to quickly select objects in your scene. To select objects, have this tool active, and click on an object in the viewports. A selected object will be turn white in a wireframe view, or will be bounded by brackets in a shaded view. You will also see the name of the object somewhere in the Command Panel. You can select multiple objects at once by holding down Ctrl and clicking on them, or may release an object from a selection by holding down Alt or Ctrl and clicking on it again.

You can also select multiple objects by clicking and dragging a selection around them. There are several buttons that may can assist you with the selection region on the Main Toolbar. The Selection Regions flyout lets you change the shape of the selection region. There is also a Window/Crossing mode toggle, which can be useful as well.

Filters are available, which let you restrict selections to only certain types of objects in your scene. There is also a Select by Name tool that brings up a dialog with a list of the objects in your scene. From this list you may choose to select multiple objects. This option is very helpful when working with a large scene, but requires that you name your objects properly, so that they can be easily located from the list. Naming your objects logically is always a good idea.

TRANSFORMS

Once you have an object selected, you may want to actually do something with it, such as move, rotate, or scale it. This is where the Transforms come in. These Transforms are available from the Main Toolbar, and when active, allow you to re-position, re-orient, or re-size objects in your scene. Also note that the name of each of the Transforms is preceded by "Select and…", such as Select and Move. Each of the Transforms may be used to select an object, as well as enact its functionality. You should be careful when using these tools to select objects however, because it is easy to accidentally move, rotate, or scale the object, when you only meant to select it.

TRANSFORM GIZMO

Whenever you have an object selected, and a Transform is active, you will see a Transform Gizmo appear on the object. If the Gizmo is not visible, ensure that the object is selected, and a Transform is chosen. If it is still not visible, hit X on your keyboard, as this toggles the Gizmo on and off. This Gizmo will indicate the different axes that are available for controlling the transformation of the object. These will be color coded and labeled on the gizmo, so that it is obvious which way the object may be transformed. By positioning you mouse over one of these axes on the Gizmo, you can restrict the resulting transformation to only that axis.

In the case of the Select and Move Transform, you will see that the Gizmo indicates the axis restriction by a colored arrow. If you move your mouse over the arrow and click and drag, the movement will be restricted to that axis. There is also a small junction area between any two axes. If you move your mouse over this junction, it will highlight in yellow, indicating that you can click and drag with a 2-axes restriction.

For Select and Rotate, the Transform Gizmo is circular. The inner colored circles restrict rotation to one particular axis. The outer gray circle allows the restriction to be based off of the screen. Note that the active restriction axis is highlighted in yellow, just as it was for movement.

With Select and Scale, you have outer handles, which control single axis scaling. The outer junction area controls two axes scaling. The inner junction area, at the core of the Gizmo, allows for uniform scaling of the object as a whole. Note that there are several scaling options available on a flyout if you click and hold on the Scale button.

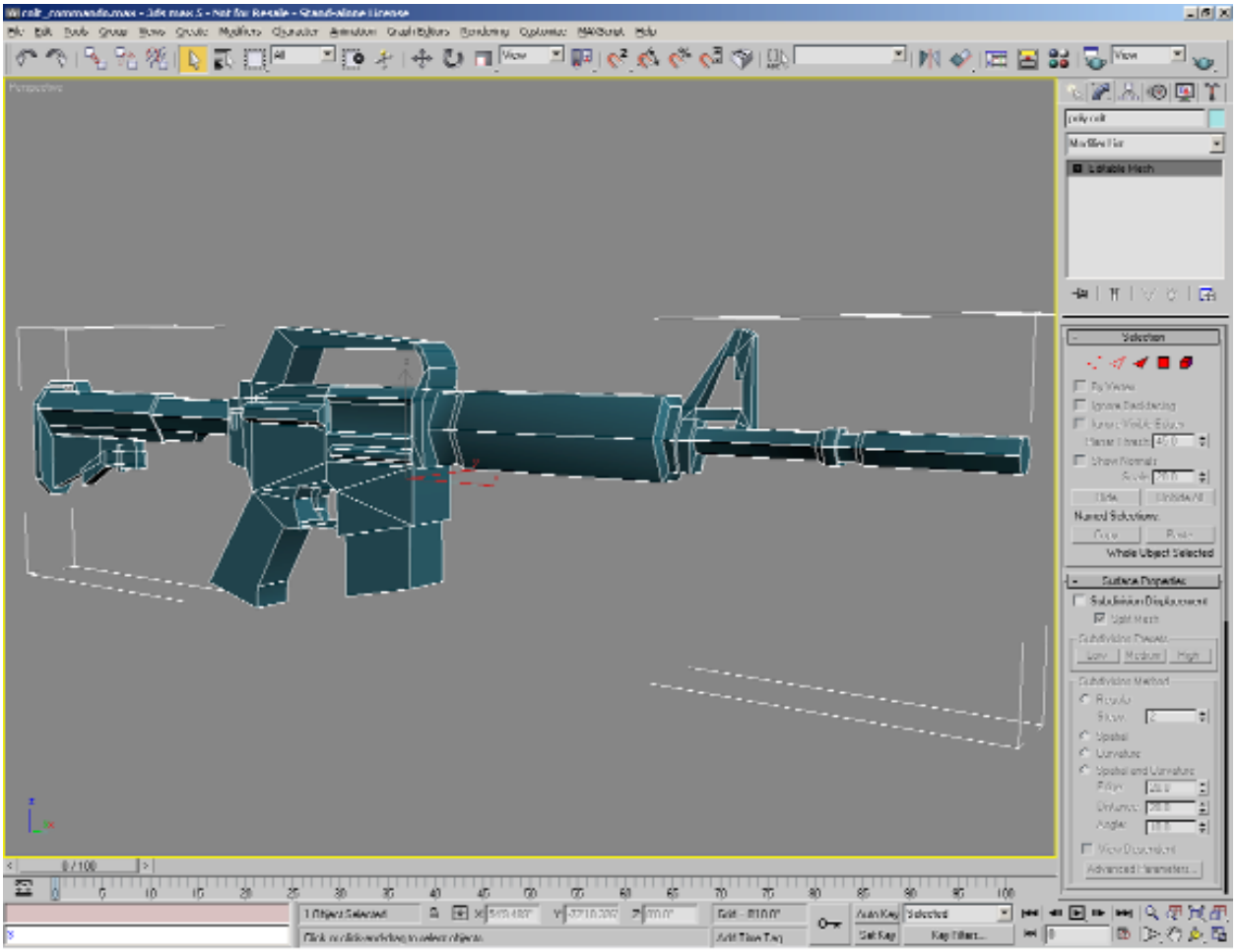
You will use Transforms so much that it is a good idea to become acquainted with the several shortcuts that may be used for accessing them. The first method is to use the Quad Menu. Just right-click in a viewport (not on the label), and from the lower right quad, choose move, rotate or scale. The second method is to use the W, E, and R keys on your keyboard, which allow you to activate the transforms as well.

SNAPS AND TYPE-IN CONTROLS

For precise work with Transforms, it is sometimes a good idea to use additional help, rather than just eyeballing the transformation with the Gizmo. There are several snap toggles available on the Main Toolbar, which can aid you in using the transforms. The snap toggle allows you to snap your movement to different points on the grid, or other points in Max. The Angle Snap toggle restricts your rotation to 5-degree increments. The Percent Snap restricts scaling to ten-degree increments. By right-clicking on each of the snaps, you can alter the restriction that the snap dictates.

In addition to using snaps, you can also use type-in controls to affect transformation. You can do this by either activating the transform, and typing in values in the Status Area, or by right-clicking on the Transform's button, and bringing up a type-in floater. In both cases, you will also see that there are spinners next to the value fields. Spinning these spinners does the same thing as typing in values. Spinner controls can be controlled with more precision by using the Spinner Snap Toggle as well.

POLYGON MODELING BASICS



Polygon Modeling

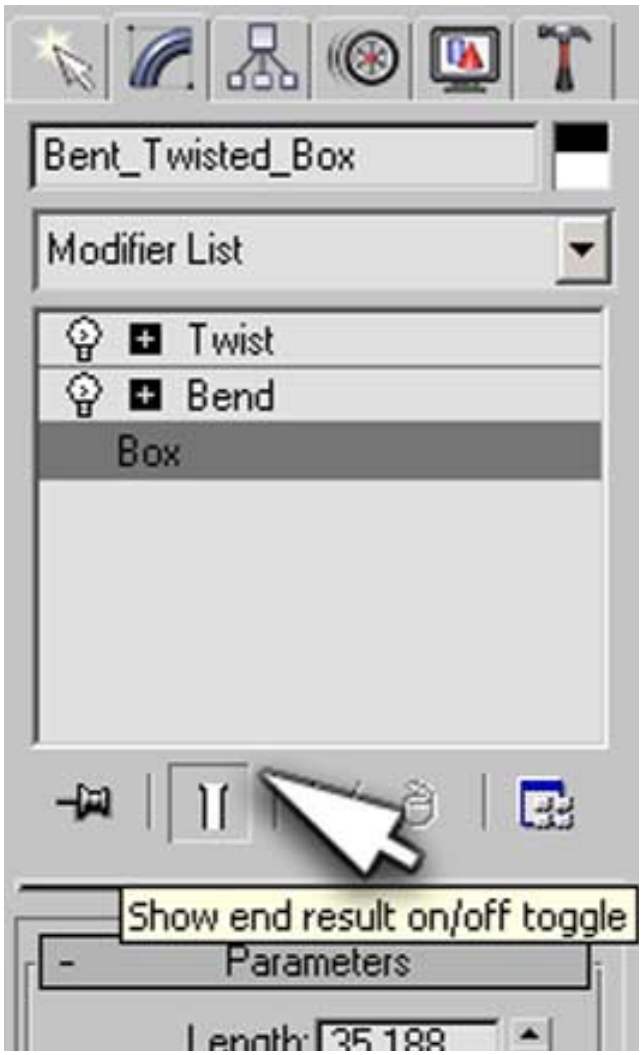
Suggested reading: 3ds max Illuminated: Foundation Chapter 2, pp. 25-55

Polygon modeling is the bread and butter method for creating most geometry for games. Mastering polygon modeling gives the artist total freedom to create whatever they want. Although some people have thought of polygon modeling as a low-tech method for creating artwork, the related tools have become so sophisticated that they now even used by artists working in film as well as games.

MODELING AT THE PROPER SCALE FOR THE GAME ENGINE

Before making anything for a game, you need to take into account the overall size or scale for the object in question. Figuring this out ahead of time will allow you to export the object to the engine at the proper size. If this is not determined ahead of time, you will end up re-adjusting objects' sizes when they come into the game being much too big, or too small

Similarly, you must consider the object's relative scale to other objects in the game. Is it a big important object, or a smaller object that is just meant to fill in the scene.



Show End Result

A good thing to do is to maintain consistency when modeling objects is to use a consistent units setup. You can do this in Max by going to the Customize Menu→Units Setup, and choosing to use generic units, US Standard feet and inches, or a Metric setup. Once chosen, you can then easily gauge how large an object is.

From the Customize menu you can also alter the System Units Setup, which controls how big your scene as a whole is. You generally don't need to adjust this unless the game you are working on requires some specific scale here.

MODIFYING PRIMITIVE OBJECTS-

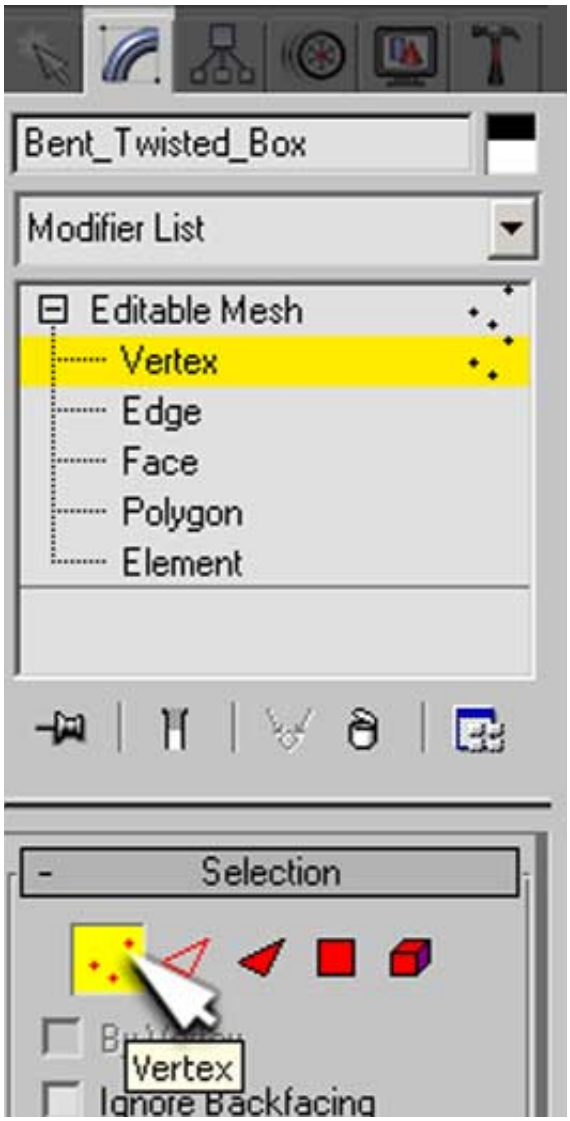
PARAMETRIC MODIFIERS

Generally, in Max you will start a model by creating a Primitive object, and modifying/restructuring it into a new object. A Primitive is a polygonal object, which has various dimensions that may be altered by the use of spinners. These spinner-controlled dimensions are also known as parameters, so Primitives may be referred to as parametric objects.

You can create Primitive objects by going to the Create Panel→Geometry, clicking one of the buttons for the Primitive, then clicking and dragging to begin establishing the objects shape. After it is created, you can select the object and go to the Modify Panel, and further alter the object's parameters to re-size it. All Primitives have different parametric controls, as they have different shapes.

One thing that all Primitives will have in common is a control for segmentation. Basically, this is the amount of sub-division for that object. It is a good idea to take a look at this closely in the Perspective view by right-clicking on the Perspective viewport label and turning on Edged Face display mode (F4). The more segmentation there is on an object, the smoother the curves will look, the better it can be deformed, and the higher the polygon count. So, for games, only apply sufficient segmentation so that the object's profile is well defined. Any excess segmentation beyond this serves no real purpose, unless it will be further deformed.

Any time you are planning on modifying or deforming the Primitive object further, you should make sure that there is sufficient segmentation to support this deformation. To quickly illustrate this example, create two boxes of a similar size. One box should have 1 segment for length, height, and width, and the other should have twenty. From the Modify Panel, click on the drop-down arrow at the top of the Modifier Stack, and apply a Bend Modifier to each of the boxes. This is a parametrically controlled Modifier, with spinners determining its overall effect. Turn up the bend Angle to 90 for each of the boxes. Note that the segmentation for the first box is not sufficient to be deformed, but that the second box bends quite well.



Sub-Object Controls for Edit Mesh

Using the Modifier Stack

The Modifier Stack is a hierarchical list of all modifications that you have made to your model. In the last example of the bent boxes, it is actually possible to change the segmentation on the boxes after the Bend Modifier has been applied. Just click on the box level in the Modifier Stack, adjust the segmentation, then go back up to the Bend level to adjust it more. Since Bend is a parametric modifier that just performs a generic, overall effect, it will attempt to deform any structure that is below it in the Modifier Stack.

If you are working at a lower level in the Modifier Stack, you must make sure that the Show End Result Toggle is on to preview the Modifiers that are higher up. For many Modifiers, this toggle will be depressed by default. If it is not, you will not see the cumulative effect.

All Modifiers have some controls that are specific to them, such as Angle and Direction for Bend. Depending on the type of Modifier, these controls may be a Parametric spinner, or something simpler. In addition, Modifiers also have Sub-Object Controls available for them. These may be accessed by clicking on the plus button next to the Modifier in the Modifier Stack to make them visible, then clicking on the text for the Sub-Object controls. The Sub-Object text will be highlighted in yellow.

Sub-Object Controls are manual controls for adjusting some aspect of the modification. In the case of a parametric Modifier such as Bend, there will be a Gizmo control available, which you can use your Transforms on. Sub-Object controls can always be adjusted by using Transforms on them. For bend, this will move the gizmo, so that the overall effect is relocated. To exit Sub-Object mode, simply click on the Sub-Object text that is highlighted in yellow again. It will no longer be highlighted. In the case of the Bend Modifier, this type of control isn't always very useful, but for some other types of Modifiers, Sub-Object controls can be incredibly powerful.

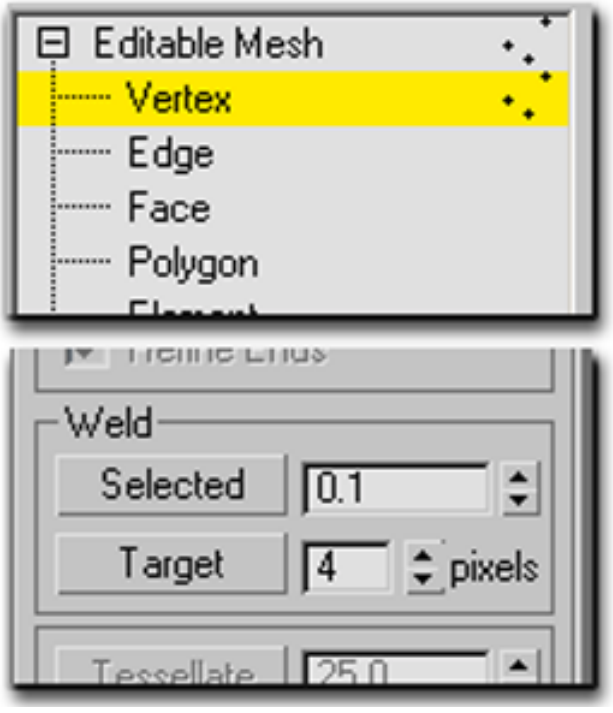
Edit Mesh Modifier

The Edit Mesh Modifier is not a parametric Modifier with a Spinner that dictates an overall effect. It provides a mesh modeling toolset for the modification of Primitive or other objects, the controls for which are almost completely dictated at the Sub-Object level. Once applied, several Sub-Object Modes are accessible by clicking on their text, or the shortcut icons that are just below the Modifier Stack. Once active, you can use your Transforms on the various portions of the object to deform it. While in Sub-Object mode, your Transforms will only affect the components of the mesh, and not the object as a whole. Once again, you can exit Sub-Object mode by clicking on whatever is yellow in the Modify Panel, so that it is no longer highlighted.

In Vertex Sub-Object Mode, you have the option to Move, Rotate, or Scale vertices, so that the points are re-arranged. You can also condense several vertices together by welding them. This is good for getting rid of excess geometry and reducing polygon counts. You can Target Weld Vertices by activating this button, then moving one vertex on top of the one that it should be welded to. You can also Weld a mass of vertices together at once by selecting them, adjusting the threshold (distance apart) spinner, then clicking on the Weld button. If the threshold is too low, the vertices will not be welded, as they were to far apart for the weld to be effective. Adjust the threshold and repeat.

In Edge Mode, in addition to transforming the edges, you have options for chamfering edges, which splits one edge into two with a new polygon in between. In addition, you should note that all the edges in the geometry are not visible by default. Although Edit Mesh is a triangle (face) based system of polygon modeling, you will only see squares (polygons) by default. This is because the interior edge inside the middle of each polygon is hidden by default. If you click and drag around a chunk of geometry, you can see dashed lines indicating invisible interior edges. If you want to make them visible, you can hit the Visible button under the Surface Properties rollout of Edit Mesh. In addition, when you want to re-orient an edge, you can use the Turn button to do so.

The controls for Face and Polygon Sub-Object Modes are almost identical, with Face working on each triangle, and Polygon working on each Square. Some of the key controls for these Sub-Object modes include Extrusion and Beveling, which are fundamental and commonly used modeling techniques. It is also possible to re-build geometry by using the Create button.

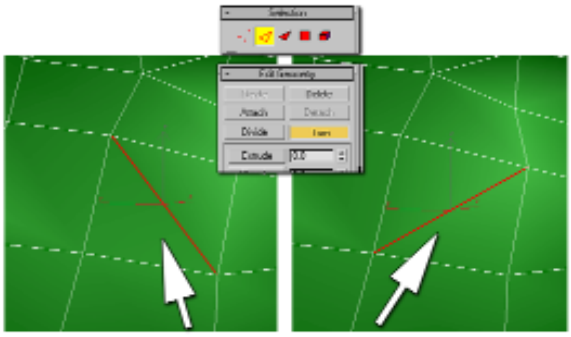


Welding Controls

NORMALS

Sometimes a part of the mesh gets deleted intentionally or unintentionally, so that there is a gaping hole in it. If this happens, notice that the interior of the model is invisible, as it is usually wasteful for Max to calculate the display of this interior portion of the mesh. When this happens, you will need to re-build the geometry, starting in Polygon or Face Sub-Object mode. Click on the Create button, and the vertices will appear. Click on the first vertex that you want to start with, then work counter-clockwise around the hole, clicking on the vertices, until you finish where you started. A new polygon or face will be built, facing the correct direction. This display direction of the polygon is called a Normal. If you had worked clockwise in creating the geometry, the polygon would be facing backward, and would seem invisible, as it is displaying in the wrong direction. The geometry does exist, but its Normals need to be flipped. If this happens, you can flip a normal by selecting the polygon, the clicking the Flip button under the Surface Properties rollout.

Element Sub-Object Mode allows you to quickly select any portions of a mesh that are not physically connected by geometry. This is known as a non-contiguous or segmented mesh. This type of mesh structure occurs when you have deleted portions of the mesh, or when the mesh was constructed from various separate objects, which were attached. To attach an object to a mesh, exit any Sub-Object Mode, and then from the Edit Geometry rollout, use one of the Attach options to select the object to be attached. This will be a new Element in the mesh. For many game engines, it is perfectly appropriate to use this modeling method, as it often results in a model with a lower polygon count, than would a heavily subdivided and extruded model.

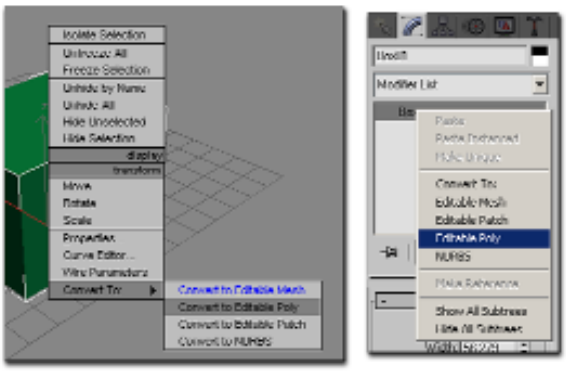


Turning Edges

ADVANCED MODELING TOOLSETS

Suggested reading: 3ds max Illuminated: Foundation Chapter 3, pp. 71-82

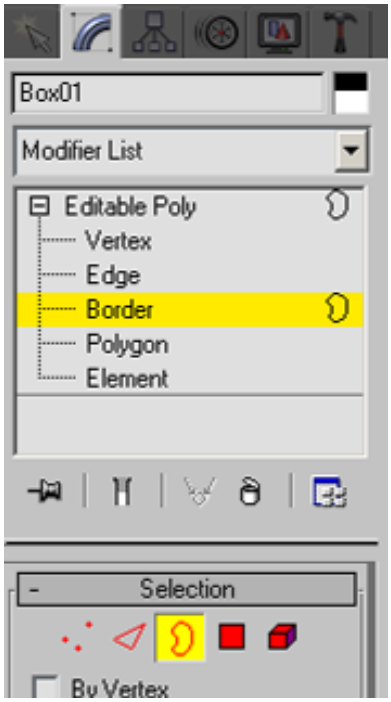
Although the Edit Mesh Modifier is a good tool for polygon modeling, it does not have the power and versatility available in the newer Editable Poly toolset. Editable Poly provides many more options than Edit Mesh, and tries to be alleviate some of the problems that can present themselves in Edit Mesh. So, in addition to its more robust toolset, it also tries to prevent modeling mistakes, which can allow you to work quicker, and without fear of error. In addition to Editable Poly, there are other modeling options available to you which can be faster to use in some situations. These options include Spline and Patch modeling.



Converting and Collapsing to Editable Poly

EDITABLE POLY BASICS

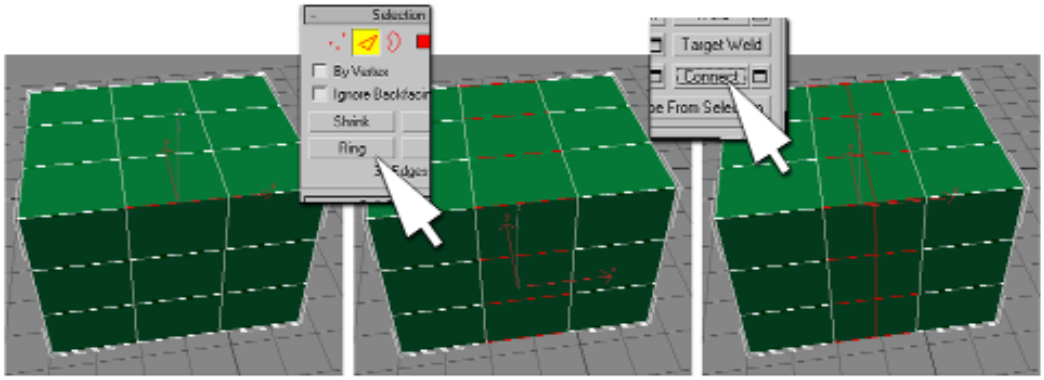
Unlike Edit Mesh, there is no Editable Poly modifier. When doing polygon modeling, as you do with the Edit Mesh Modifier, the editing that you do with these tools completely hinges on the underlying structure of the object. So, in the case of Edit Mesh, if you were working on an object that was originally a box, and performed extensive modifications at the Edit Mesh Level, then attempted to go down in the Modifier Stack and change the segmentation of the box, you would see some problems. All of the Edit Mesh modifications hinge on the structure of the box below it. If you change that underlying structure, the Edit Mesh modifications will become inaccurate, and things will look bad. This should be avoided.



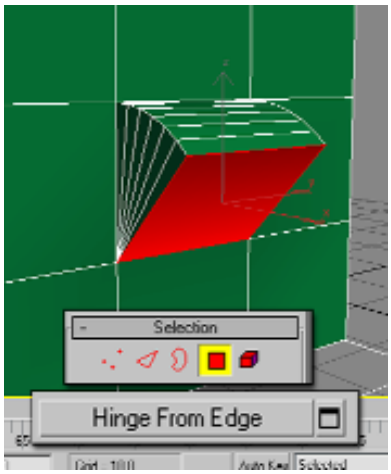
Editable Poly Sub-Object controls

Editable Poly avoids this whole situation because it is not a Modifier. Rather, it is an Editable State that an object may be converted or collapsed to. To modify a box using Editable Poly, you must either Right-click on the Object itself, or its Modifier Stack, and choose Convert To→Editable Poly. The object is condensed to an Editable base state, which can be modified in the Modifier Panel. Note however, that the Box level in the stack is no longer present.

The first difference between Editable Poly and Edit Mesh is that it is a Quad-based Polygon toolset vs. Triangle-based Edit Mesh. This means that all geometry is based on polygons (squares) vs. faces (triangles) in Edit Mesh. Also, Edit Poly has automated management of interior edges, so that you usually don't have to worry about interior edges. This prevents some modeling errors, such as accidentally moving a vertex so that it crosses a hidden edge. In Edit Mesh, this would be a fold with poor shading, and would need to be fixed by turning the problematic edge. Editable Poly will automatically manage this for you, turning edges as you go so that you can't get crossing edges.



Using Ring and Connect



Hinge from Edge

EDITABLE POLY SUB-OBJECT CONTROLS

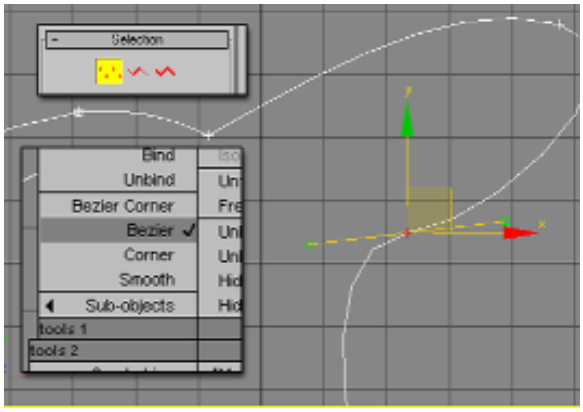
Vertex Sub-Object Mode in Editable Poly is pretty similar to that of Edit Mesh. You will notice that the Target Welding functionality is a little different. Rather than having to move the vertices, you can just click on them, a dashed line will appear, then you can click on the target to weld to.

There are some unique tools for Edge Mode however. If you start with a converted Primitive object, there are a few tools for quickly selecting multiple edges. Loop allows you to select edges, which run end to end. Just select one of the edges, then click Loop, and all edges that loop around the model are selected. This makes for easy chamfering of edges. Ring selects edges that run parallel to the selected edge. This may be used in conjunction with the Connect option that will draw new edges perpendicular to those selected, splitting the old geometry down the middle.

Border Sub-Object Mode is unique to Editable Poly. A border is the open edges adjacent to any hole in the mesh. This can be used to quickly locate holes in your mesh. The Cap option will automatically create new polygons to fill this border.

Polygon Mode is a little different as well. The Cut tool allows you to manually subdivide portions of the geometry, just as you could with Edit Mesh's Cut. In Editable Poly though, the Cut is previewed before you finish it, and it allows for easy snapping to edges or vertices. Inset allows you to do a zero extrusion/scale that lets you create a smaller polygon inside of a larger one. Hinge from Edge gives you the ability to do a swivel extrusion, and turning edges is also handled in Polygon Sub-Object mode, by using Edit Triangulation.

Many of the functions in Editable Poly also have Settings buttons. These bring up a little type-in floater for precision control of the command. In addition, they preview the overall effect of the command, so that you know what result you are going to get ahead of time. This helps to avoid the guesswork of adjusting a threshold, attempting to apply, then having to do it again if it was wrong. The Settings dialogs let you either hit OK, which registers the result, and closes the dialog, or hit Apply, which registers the result, and allows you to perform the command again.



Using vertex handles to control curvature

Quickslice is another command available at for both selected geometry at the Polygon Sub-Object level, or for the object as a whole at the object level. This command allows you to quickly slice a line of edges/vertices through the model, adding geometry where you need

SPLINES AND PATCHES

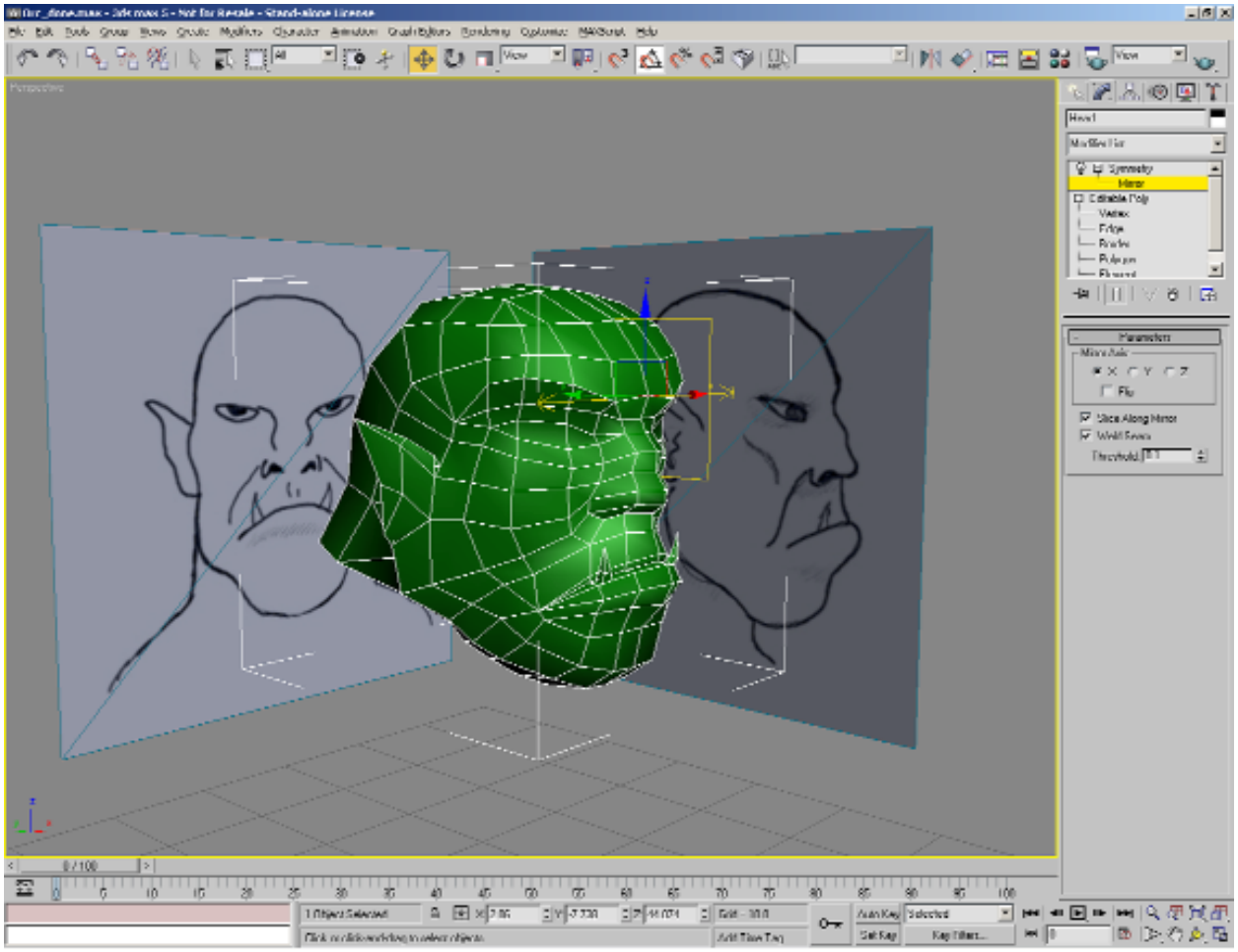
Other options for starting a model may entail the use of other shapes such as Splines. Splines are simply line shapes, which may be modified with the use of Modifiers such as Extrude, and Lathe, or may be combined into Compound objects such as lofts. Sometimes the Sub-Object controls that Splines provide you can make it easier to create some objects, particularly those with complex curvature.

Patches are another option, and are also good for getting subtle curvature in geometry. Patches models come in two general varieties. The first is a Spline cage with a Surface Modifier applied to it. This is a Spline/Patch hybrid technique, which used to be a plug-in for Max called Surface tools. The second option is a Patch Surface with an Edit Patch Modifier applied to it. This is a pure patch model, not based on Splines.

The power of both Patch and Spline Models lies in the editing of their vertices. By right-clicking on their vertices at the Sub-Object level, you can change the corner type of the vertices. Both Patches and Splines allow for you to access vertex handles, depending on the corner type chosen. These handles control the curvature of the surface into and out of that vertex. By turning the handles, you can add subtle or drastic curvature to the overall surface of the model.

In addition, with both of these tools, you have controls for how much interpolation/steps/segmentation that the model has as a whole. This determines the detail, density, and overall polygon count of the model. Some game artists are now choosing to create a high detail model that could be used for promotional material and cut-scenes, and then turning down the segmentation/steps to create lower Level of Detail models that will be the in-game models. This is a perfectly appropriate way to work if it is called for, and Spline/Patch models lend them to this workflow quite well.

MODELING STRATEGIES



Good strategy means less time

It's not only important to know the tools in Max when modeling, it's also good to think about how you should use them. Once again, this is planning. Proper planning before modeling can save time when constructing the geometry, and can help to avoid problems down the road.

MODELING IN HALVES

If you are constructing a model that looks the same on both sides, why do twice the work? Just create half of the model, and apply the Symmetry Modifier, which will mirror the geometry, attach it, and weld it down the middle. There are axis controls for the Symmetry Modifier controlling which axis the model will be mirrored from. You can actually apply this Modifier, go back down in the Modifier Stack and turn on Show End Result to preview the cumulative effect as you continue to make modifications.

If you would prefer to do things the manual way, you could use the Mirror tool from the Main Toolbar to get similar results. Mirror the model, Attach it, then weld the vertices down the middle.

PUTTING DETAIL WHERE YOU NEED IT

When creating geometry for a model, think about how it will be seen in relationship to camera. There is no point to modeling the treads on a character's boots if he's never going to kick his heels up.

In addition, you must be careful to plan for deformation of the model when it is going to be animated. Remember, if a mesh is going to be deformed, it must have enough structure to support that deformation. This is most evident in character models.

When creating character models, you must make sure that their joints have enough segmentation to be deformed well. Consider the animation that the character will need to perform, and plan for that appropriately. Will the character need to run? If so, it needs segmentation at the knees, hips, and some at the ankles, so that they can deform properly. Throw a baseball? It'll need segmentation at the shoulder, elbow, and wrist. So, always take into account how the model will need to be animated, and plan accordingly.

DEFINE THE SILHOUETTE

Once you have determined how the mesh will need to deform, you should think about where you can keep the polygon count to a minimum. Most low polygon game artists will emphasize the importance of the silhouette of the model. What this means is that textures can fake a lot of the surface detail on a model, but cannot fake the overall profile shape or silhouette of the model. Taking this into account, you should avoid adding excessive geometry to a portion of the model, unless it contributes to the silhouette. For example, there is no point in modeling in the veins on a character's arms, as this could be faked by textures. However, adding in enough segmentation so that the character's biceps have a nice smooth bulge might be worth it.

Generally, you will be trying to model something within a specific polygon budget. Take into account where the extra polygons will give you the biggest impact. Generally deformation and silhouette take precedence. Also take into account the angle that the model will generally be seen from, and put the detail into those areas.

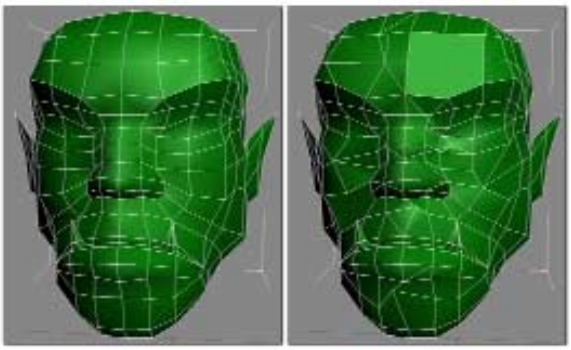
CONTIGUOUS AND NON-CONTIGUOUS/SEGMENTED MESHES

When modeling, you should also consider if the game engine prefers contiguous or non-non segmented meshes. A contiguous mesh is a model with that is all physically connected, and is a single element. The model is not composed of separate pieces/elements. This generally makes for a fairly clean mesh, which is a little higher in polygon counts. The increased polygon count occurs because geometry is frequently cut or subdivided, and then extruded. These cuts and subdivisions increase the polygon count on the main bulk on the mesh.

Many game engines do not require contiguous meshes, and allow for a mesh to consist of separate elements. If this is the case, additional geometry may be modeled separately, then attached to the main bulk of the mesh, with no concern for geometry bridging the gap and connecting the pieces. Since you can avoid the necessity of the geometry connecting the pieces, the overall polygon count will be lower.

OPEN-EDGED MODELS AND AIRTIGHT MESHES

Yet another polygon saving technique you can use is to delete portions of the mesh that will not be seen by the player of the game. This is only appropriate if the game engine can tolerate open edges on models. Some engines see this as an error, and prefer that models are sealed up, airtight models. If open-edged models are acceptable, it can be very easy to save on polygon budgets by deleting unnecessary geometry and just leaving a hole in the mesh. This is most frequently done with scenery object, where the polygons on the underside of the model are deleted. There is no reason to have this geometry on the model if it will always be planted on the ground, and hidden from view.



There is a difference between good edge flow and poorly planned geometry.

KEEPING THE MESH CLEAN

In general, you should do your best to avoid common modeling errors such as double faces, isolated vertices, and crossing edges. You should always make sure to check you work for just these types of errors, even though you will avoid most of them if you are careful when modeling. Nevertheless, you should always check your work, and can use tools such as the STL Check modifier to help locate erroneous geometry.

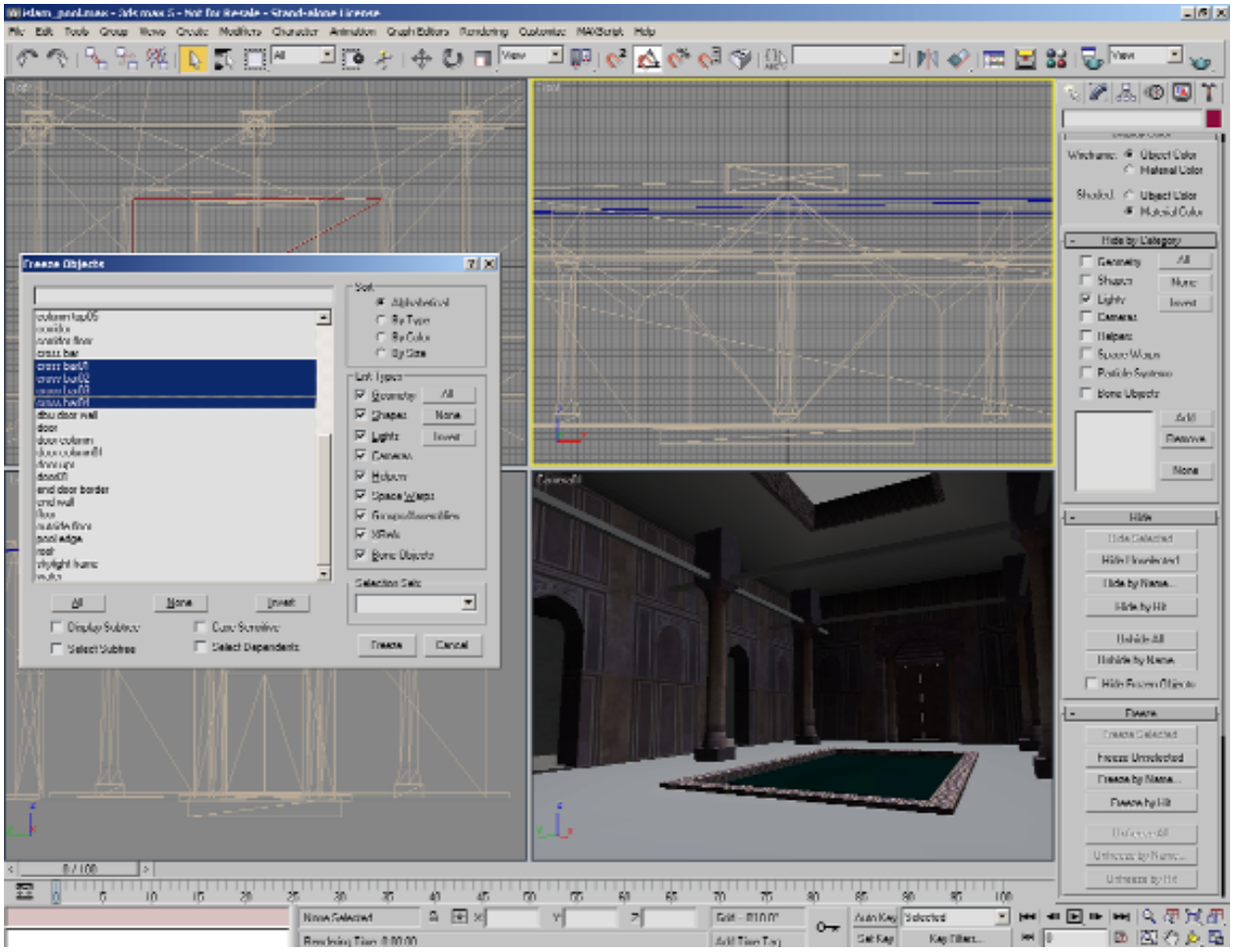
In addition to these common problems, you should try to ensure good edge flow throughout your geometry, particularly in the case of character models. For organic geometry, edges should flow in loops around your model. Try to avoid abrupt edge turns whenever possible. This will make the model look more fluid and lifelike.

SMOOTHING

When the model is complete, you will see that it may look very faceted. This is because the light is not hitting the edges of polygons properly. To fix this, select all of the polygons in Sub-Object mode, go to Polygon Properties, and type in a value next to the AutoSmooth button. This is essentially an angle threshold. Next, click on the AutoSmooth button to register the value that you just set. Any polygons that are next to each other and are turned less than the angle threshold value apart will be assigned to the same Smoothing Group. They will blend together, with light gently gradiating over their surface.

Now when you select a polygon, you will see that it is assigned to a specific Smoothing Group, and this group is indicated by one of the small buttons being depressed. A polygon can be assigned to Multiple Smoothing groups by depressing more than one button at a time. Always remember that if two adjacent polygons are assigned to separate Smoothing Groups, a hard edge will be visible between them. Some games will take into account smoothing information that you set in Max. Others will apply a generic smoothing solution to the model in the game itself.

SCENE MANAGEMENT



Keeping Scenes organized

If you need to create a more sophisticated segmented model that is composed of various separate pieces, keeping the scene organized can be an important concern. Effective management of files and all of the content within a file will make it easier to get the job done, especially if several artists will be working from that same file and/or directory.

NAMING CONVENTIONS

Proper naming conventions should be established and followed religiously. Names of objects and files should be brief, abbreviated descriptions of what they are. Spaces in the name should generally be avoided, as this can make it difficult when the programmers need to write code for the asset. Naming objects appropriately is an absolute necessity, and allows you to use such functionality as Select by Name, Hide/Unhide (or Freeze) by Name, Exporting, and Merging.

HIDING OBJECTS

Hiding Objects can be a good idea if you are trying to work on pieces of a larger scene in Max. Hiding object make them invisible, even though they are still present in your scene and may be unhidden at any time. Hiding can be done from the Display Panel, or from the Quad Menu.

FREEZING OBJECTS

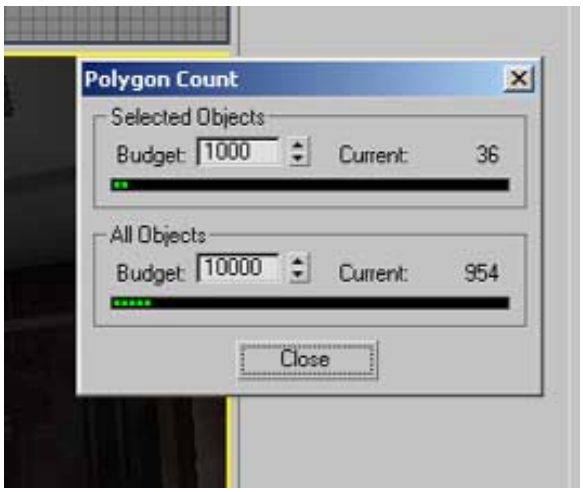
Freezing objects serves a similar to purpose, but makes the object un-selectable but still visible. This allows you to use them as visual reference, but prevents you from accidentally selecting and transforming them. Objects can be frozen and unfrozen from the Display Panel, or from the Quad Menu.

MERGING OBJECTS

The Merge command is available from the File Menu, and allows you to bring in certain or all of the objects from one scene into the one that you are currently working on. Many artists and studios save generic, useful objects to a model library, so that you can later merge them into your scene if need be. Merge prompts a select by name type dialog, which further illustrates the need for proper naming.

GROUPING OBJECTS

Generally, grouping objects from the Group Menu is only a temporary way to help you manage multiple objects in your scene. Most game engines will not interpret grouping in any way, so the Group command is only useful in an organizational sense. It allows you to quickly select a set of objects, and to transform them all at the same time. The Ungroup or Explode commands should be used when the file is completed, so that neither programmers, nor other artists will be hindered by groups that they will need to destroy.

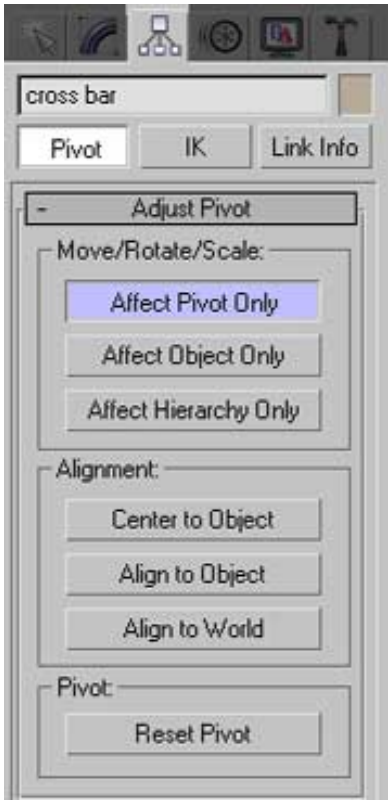


Using the Polygon Counter Utility

MANAGING POLYGON COUNTS AND LODS

Some artists prefer to monitor their polygon counts as they work, and there are several ways to do so. The Polygon Counter –is available in the Utilities Panel under More, and prompts a floater with a running triangle count. This utility currently does not function properly with Editable Poly Models, as they are quad-based geometry. This may be overcome by applying a Turn to Mesh Modifier to the top of the Modifier Stack. You could also find out the face count by going to the File Menu→Summary Info.

Some games need varying Level of Detail models (LODs) for each piece of content. These are simply lower polygon count variations of the original model, which will be swapped out in the game as the model recedes in the distance. Doing LODs for a model requires polygon reduction, and usually some UV adjustments to the model as well. Polygon reduction can be done by hand by simply welding vertices and deleting geometry. This gives you the ultimate in control, but can be rather time consuming. You may wish to experiment with the MultiRes Modifier, which is a polygon reduction Modifier that can be fairly effective.



Affect Pivot Only

CLEANUP AND TROUBLESHOOTING A MODEL

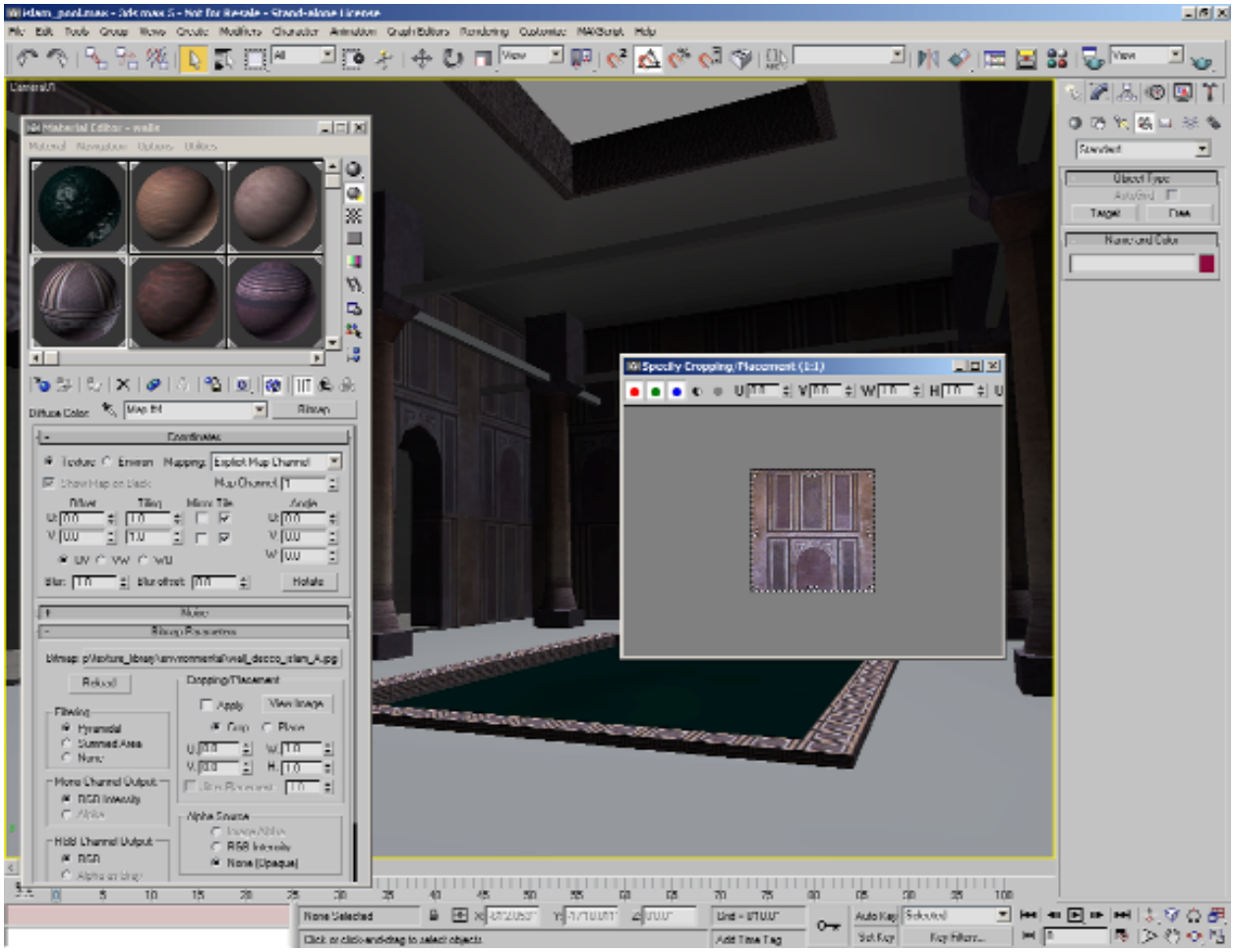
If the model that you are working on may be used by another artist, you may want to collapse the Modifier Stack so that they will have a clean slate to work from. This is generally good policy, unless there is a particular Modifier that will need to be accessed again. Collapse the Stack by right-clicking on the top of the Stack, or by right-clicking on the object and choosing Convert To: Editable Mesh/Poly.

The center point of the object is called its Pivot Point. This is the point that the Transform Gizmo is generally centered on, and around which the model rotates. If the model has been heavily modified, this Pivot Point may need some adjustment. This can be done from the Hierarchy Panel→Affect Pivot Only. With this button on, you can either manually move the Pivot where you want, use type-in Transforms to do so, or use the default controls in the Hierarchy Panel to align and center the pivot to the object itself, or the world.

Also available in the Hierarchy Panel are the Reset Transform and Scale option. These wipe the slate clean with regards to transformation information. It is good policy to use these options, particularly Reset Scale. Many game engines will have difficulty with objects scaled to anything but 100%.

The Reset X-Form Utility from the Utility Panel is another way to Reset all of the transforms, and force Max to interpret the geometry correctly. This Utility puts a special type of Modifier on the top of the stack. This Modifier may be collapsed, so that the object is just a mesh. This Utility is very powerful, and will force Max to re-interpret the geometry of the mesh more accurately, and will fix any erroneous calculations.

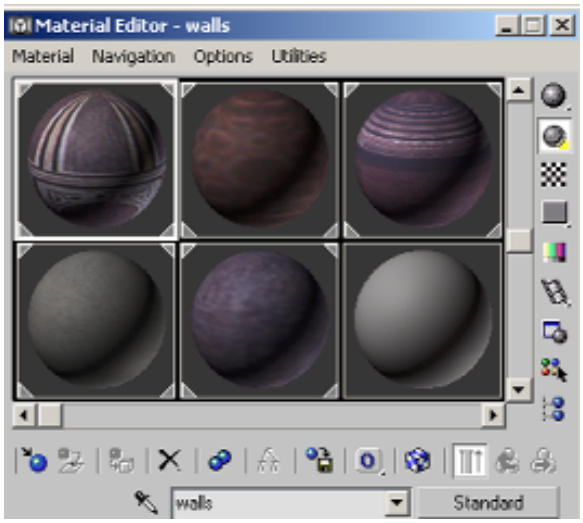
MATERIALS



The Material Editor

Suggested reading: 3ds max Illuminated: Foundation Chapter 4, pp. 93-110

Materials, and the texture maps that they are composed of, play a very important role in making game models look good. Detail, which may be lacking in the low polygon model itself, can be faked or at least enhanced with the use of good texture maps. Current games are now supporting multiple texture maps in many cases, allowing the artist to control exactly how the surface of the model appears. The main Diffuse color will almost always be determined by a texture map, but so too might the bumpiness and transparency. In Max, these attributes can be determined within Materials. You will use the Material Editor to create these materials. In addition, you will need to ensure that the texture maps are properly aligned to the model by manually supplying mapping coordinates to the model.



Active and assigned Materials

MATERIAL EDITOR BASICS — MATERIAL SLOTS

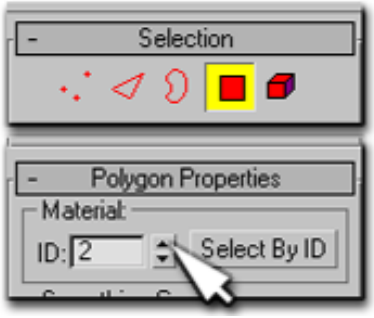
Think of the Material Editor as you would a painter's palette. You can access it from the Main Toolbar, by hitting M on the keyboard, or from the Rendering Menu. It is where you go to prepare paints or Materials that are to be applied to an object. Just because the Material is on your palette (in the Material Editor), does not mean that it is on your canvas (applied to an object). Similarly, just because a Material is applied to an object in the scene, does not mean that it is currently in the Material Editor.

In the Material Editor, various Material Slots will be available, indicated by sample spheres. More slots are accessible by moving your mouse between the borders of the slots and clicking and dragging. You can also set the number of viewable slots my clicking on the Material Editor Options button, or by right-clicking on a slot itself. Only one slot can be active at a time, and is indicated by a white border. Similarly, if a Material is applied to an object in the scene, it is flagged by triangles in the corners of the slot.

You can assign Materials to objects in your scene by using one of two methods. First, you can click and drag the sample sphere onto an object in your scene. Second, you can click on the Assign Material to Selection button. This is a better option for crowded scenes, where object may be selected by using Select by Name.

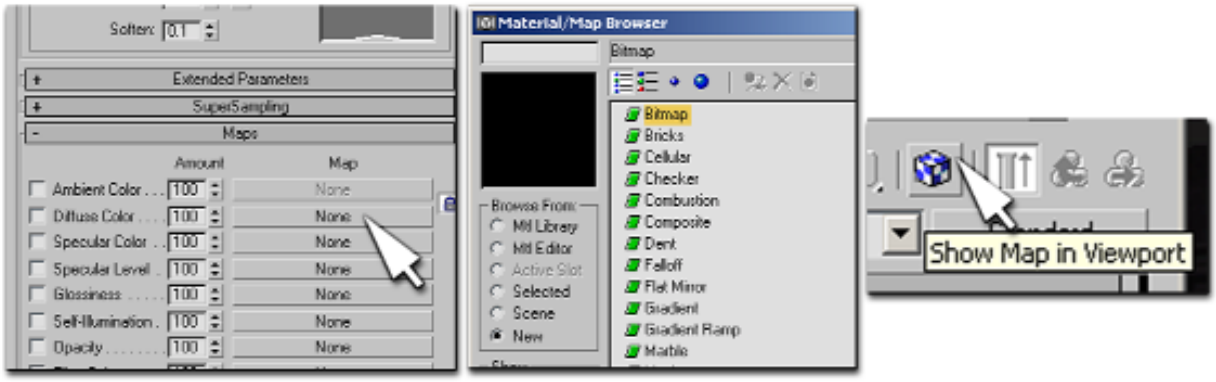
TEXTURE MAPS - BITMAPS

In games, almost all of the surface appearance of a model is controlled by various texture maps that are usually in some specific image format such as BMP, TGA, PSD, etc. In Max, any image such as these is referred to as a Bitmap, and is incorporated into a 3d Material. Adding texture maps to a Material can be done by expanding the Maps rollout in the Material Editor, then clicking on the None button. At this point, the Material/Map Browser will appear, and you will choose Browse From→New, choose Bitmap, then locate the image that you want to use. By default, the Material will only show its default color on the object to which it is applied. To see the texture itself, you must click on the Show Map in Viewport button in the Material Editor. If the map is not being displayed properly, you may need to supply Mapping Coordinates, which will be covered shortly.

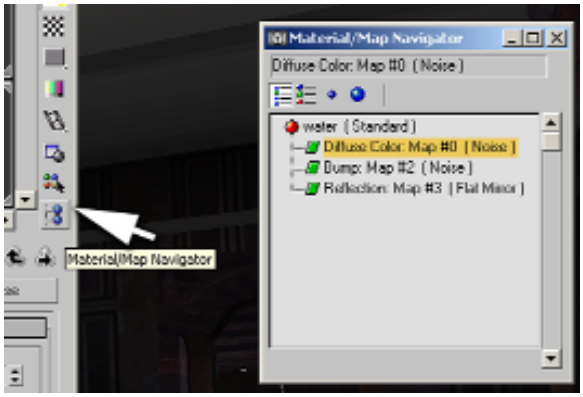


Setting Material IDs

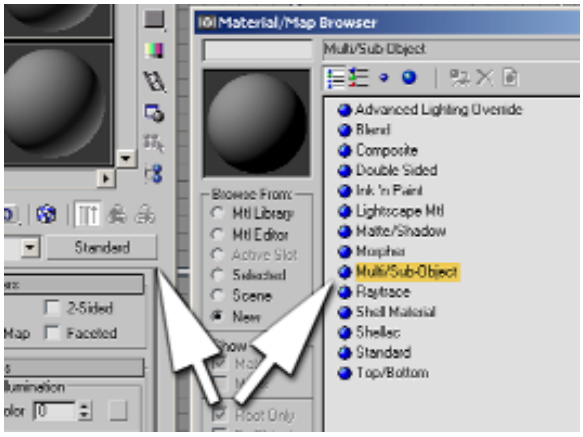
There are various attributes that a Material can control, but three of them are the most commonly used in games. Diffuse is the main color, which will almost always be used. Bump controls the appearance of surface irregularity, or bumpiness, and is usually indicated by a black and white image, where white areas make the surface of the object appear to pop out more. Black and white images may also be used as Opacity maps, which control transparency. In the case of Opacity, white areas will be visible, black invisible.



Click on the None button next to the attribute, and choose a New Bitmpap. Be sure to choose Show Map in Viewport.



The Material/Map Navigator



Using Multi/Sub-Object Materials

Once you have added a Bitmap to your Material, you will see that the controls in the Material Editor look a little different. This is because you are one step below the root in the overall Material Hierarchy, at the Bitmap level. Here you will find a control under the Bitmap Parameters rollout for View Image. Most of the other controls for tiling and alignment that you will also see are not always applicable to games.

If you want to get rid of a texture map, at the Bitmap level, click on the large Bitmap button, Browse from→New, and choose None from the list. You can also achieve this same effect by clicking and dragging the text that says None from one inactive attribute into the one that currently has a texture active.

NAVIGATION

If you need to get back to the root of your Material to add more texture maps, you can do it by hitting the Go to Parent button. If you choose, you can also bring up the Material/Map Navigator, which is a floater controlling, and indicating your position in a Material's hierarchy. This tool can really help to keep you from getting lost in the Material Editor.

In addition, sometimes you lose a Material from the Material Editor itself, even though you still see it on an object in your viewports. It can be re-called to the Material Editor by activating a Material Slot, then clicking the eyedropper tool and clicking on an object in your scene. The active slot will now indicate the Material in question. Note that this does not erase the Material that was previously in the slot, if that Material is currently applied in the scene.

MULTI/SUB-OBJECT MATERIALS

Especially with newer games, there is the possibility that a single model may require multiple Materials. Commonly, this might be a Bitmap for a character's head, and another for its body. To be able to assign different Bitmaps to different portions of a model, you must prepare the mesh itself first. Do this by collapsing the stack, then going to Polygon Sub-Object Mode, and select the polygons for the first portion of the mesh. Go to Polygon Propertied, and set the Material ID to 1. Next, select the rest of the mesh (Edit Menu→Select Invert), and set its ID to 2.

Open the Material Editor, choose a Material Slot, then click on the Material Type button, and change it from Standard to Multi-Sub Object. Set the number of materials to two. So, you have a new master Material that now consists of two sub-materials, and each one will correspond to the IDs that you set the mesh to.

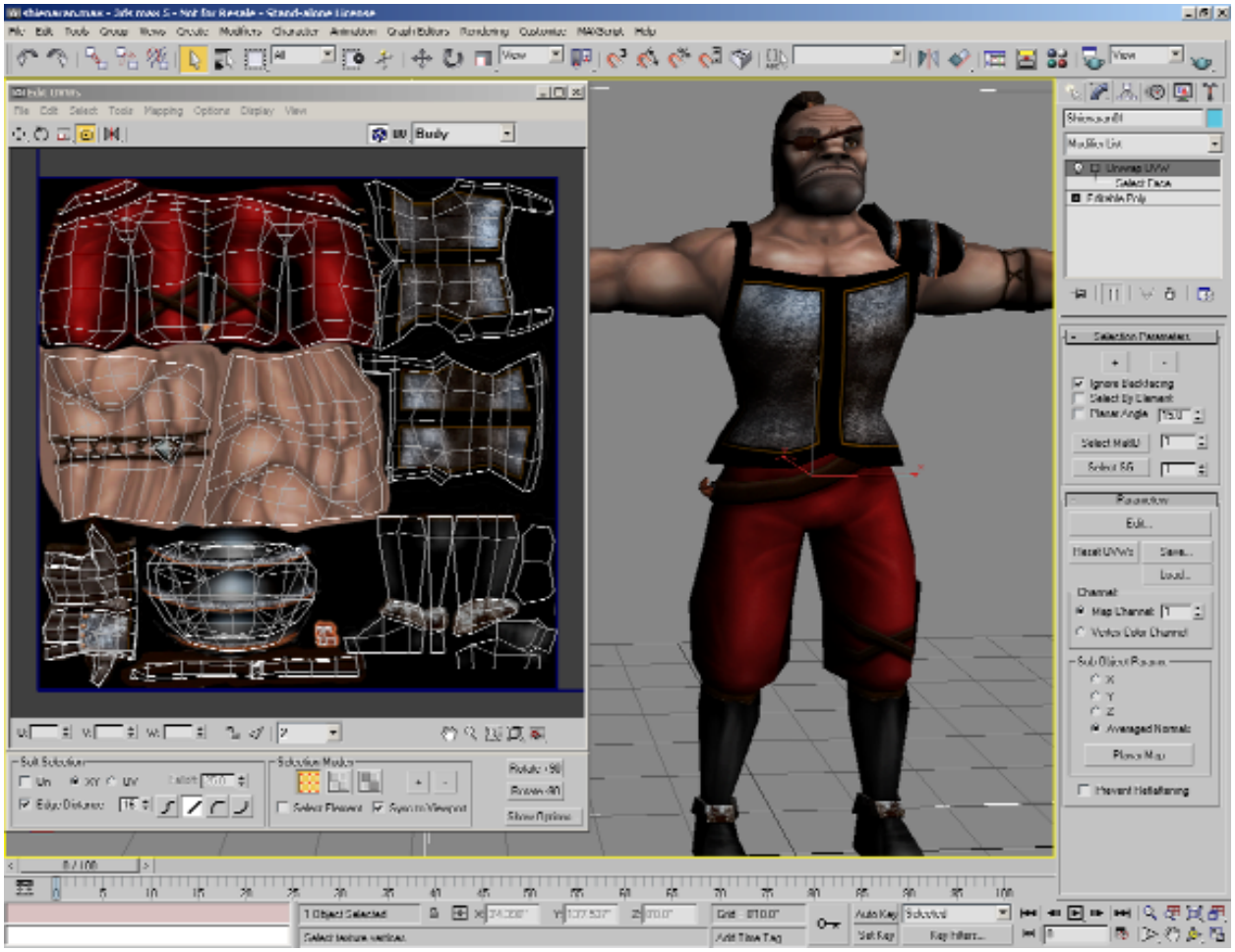
RENDERING

Rendering is not something that you will be doing for a real-time game in Max. Even so, it can be a good way to double check your work as you progress, especially with textures. Materials are not shown flawlessly in Max's viewports, so it may be a good idea to check your work by doing test renders. You can do this by clicking on the Quick Render button, so that the active viewport will be rendered. You may also want to bring up the ActiveShade floater from the Quick Render flyout. This floater is an auto-updating draft renderer that will give you feedback as you make adjustments to your Materials.

CONFIGURING PATHS

Sometimes Max can lose track of exactly where a Bitmap is located on your harddrive or network. This is usually due to operator error, especially when a file is being shared between several artists. This can be avoided if Bitmaps are always stored and assigned from the same location on the network. But accidents do happen, and you may need to manually direct Max where to look for a texture folder. You can do this from the Customize Menu→Configure Paths→Bitmaps dialog.

MAPPING COORDINATES - UVW ASSIGNMENTS



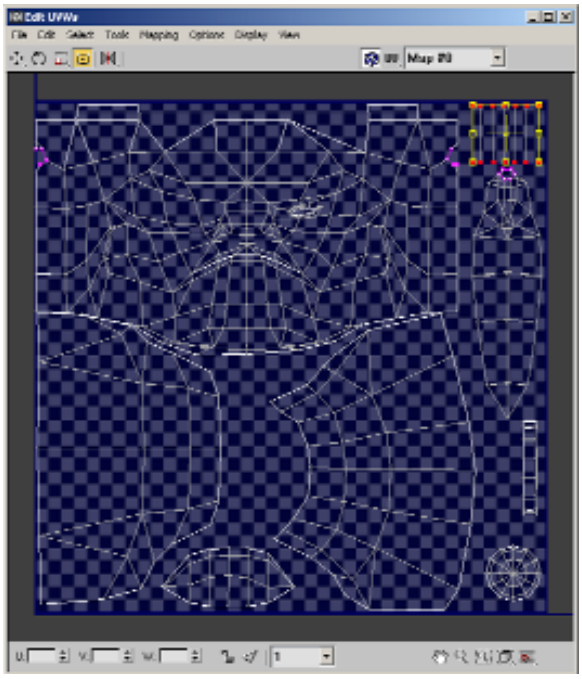
Assigning mapping coordinates

Mapping coordinates determine how textures will be projected onto the model. Primitive objects have mapping coordinates built into them by default. If you have heavily modified a primitive object, and have constructed new geometry, you will need to manually assign mapping coordinates to the model. This can be done by applying a UVW Map Modifier. In addition, you can also use the Unwrap UVW Modifier to both assign new mapping coordinates, and to adjust existing mapping coordinates.

UVW MAP MODIFIER

The UVW Map Modifier allows for quick alignment of mapping coordinates for an object. The term UVW refers to mapping coordinates for the various axes, X (U), Y (V), and Z (W). This Modifier allows you to set the shape of the projection onto the object, by choosing a mapping type such as Planar, Box, Cylindrical, etc. It also provides controls for the overall size and tiling of the map projection, and some automated alignment controls. In addition, you can activate Sub-Object Mode to manually realign the Gizmo. Once applied, the Modifier Stack can be collapsed, and mapping coordinates will be baked into the mesh.

Using Using Unwrap UVW



Setting up a UV layout

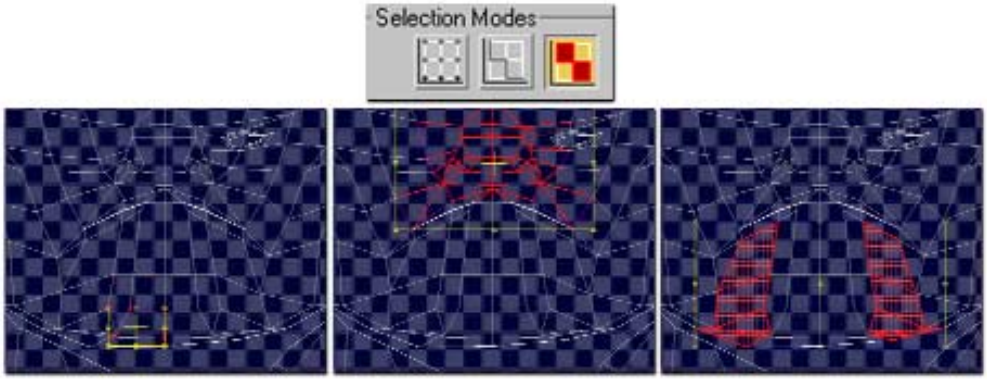
UNWRAP UVW MODIFIER

The Unwrap UVW Modifier is a vital tool for game work, and can work in two overall ways. The first is for the adjustment of existing mapping coordinates/UVW Map Modifiers. If there is a UVW Map Modifier directly below it in the Modifier Stack, the Unwrap works on just those UVW coordinates. If the stack has been collapsed, the Unwrap will work on all mapping coordinates baked into the mesh. Once it is assigned, you can hit the Edit button to begin editing the layout of the UVW map projection.

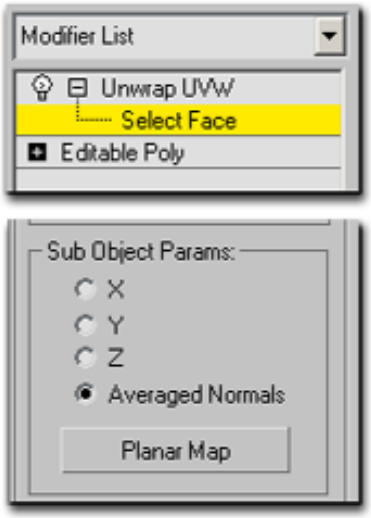
The second method in which Unwrap UVW can work is to assign mapping coordinates. If the model does not have any mapping coordinates, Unwrap will assign Planar mapping by default. If you hit the Edit button, you will see that this is the case initially. You can then turn on Select Face Sub-Object mode to select other portions of the mesh, and use the Planar Map function to re-map portions of the mesh. There are axis controls to dictate the angle from which the polygons are re-mapped from.

EDIT UVW DIALOG

Hitting the Edit button brings up the Edit UVW Dialog. From this dialog, you can use some of the various tools to move the mapping coordinates around, so that they are organized well. If you have an existing texture map, you can move UV elements around to line up with the correct portions of the image. In games, textures will generally be laid out like a decal sheet, similar to those used for plastic models that come with a sheet of stickers that need to be applied. If you do not already have a texture prepared, you can simply do your own UV layout, then export this as a template to be painted over in PhotoShop, or some other paint program. Generally, you want to keep this layout square, as game textures usually need to be in this shape.



Using different selection modes



Assigning Planar Mapping in Unwrap

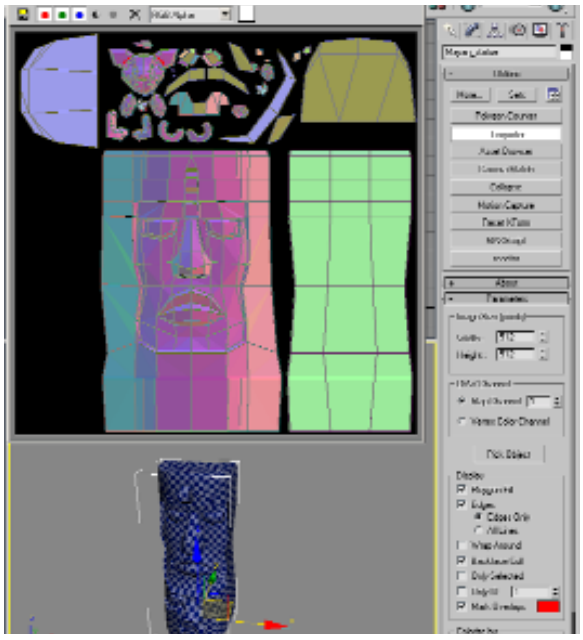
At the top of the dialog, you can choose which map to preview in the Edit window. Just below the window, there is an option to Show all Material IDs, or only one of them if the object has a Multi-Sub Object material applied to it. There are selection modes for Vertex, Edge, and Face modes at the bottom of the dialog. These selections will sync up with the selection in the viewports if Select Face Sub-Object mode is active. In the dialog there are some Options available for what is displayed in the Edit window. You can turn off the tiling and increase the brightness if it is helpful.

WORKING WITH UVS

If you are doing UV assignments, and the model does not already have textures painted for it, it can be helpful to apply a Material with checkerboard texture to the model as you work. This just helps to give you a visual cue, and allows you to look for any bad mapping projections, which will cause the texture streaking known as distortion.

Hiding Seams

With the checkerboard applied, use Unwrap to re-assign mapping coordinates when necessary, then use the Move, Rotate, Scale, Freeform, and Mirroring tools to get a good UV layout. Be conscious of the fact that any time you re-map a certain portion of the mesh, you will get seams around its borders. The edges of this UV element will be at a different location than those of the geometry next to it. This means that the geometry will be a different color, and that a seam will show. Seams are a necessary evil, but should be hidden in less visible portions of the model when possible. For example with a character, you may want to apply a cylindrical UVW Map to the arm, making sure that the seam is on the less visible underside of the arm. These UVs may then be adjusted via Unwrap. Remember that you can use Unwrap in conjunction with a UVW Map modifier, but just be sure to collapse your stack down and apply a new Unwrap if you want to work on the rest of the model.



Texporter

Some additional tools that you will want to explore in the Edit UVW dialog are available under the Tools menu of the dialog. These include the Weld options, which allow you to weld UV points together (alleviating seams). You might also need to break up portions of the UV mesh, and can do this by selecting the portion that you want to break free and choose Detach Edge Vertices. Once you have gotten a UV element aligned, you can also choose to freeze or hide the UVs via the Quad Menu.

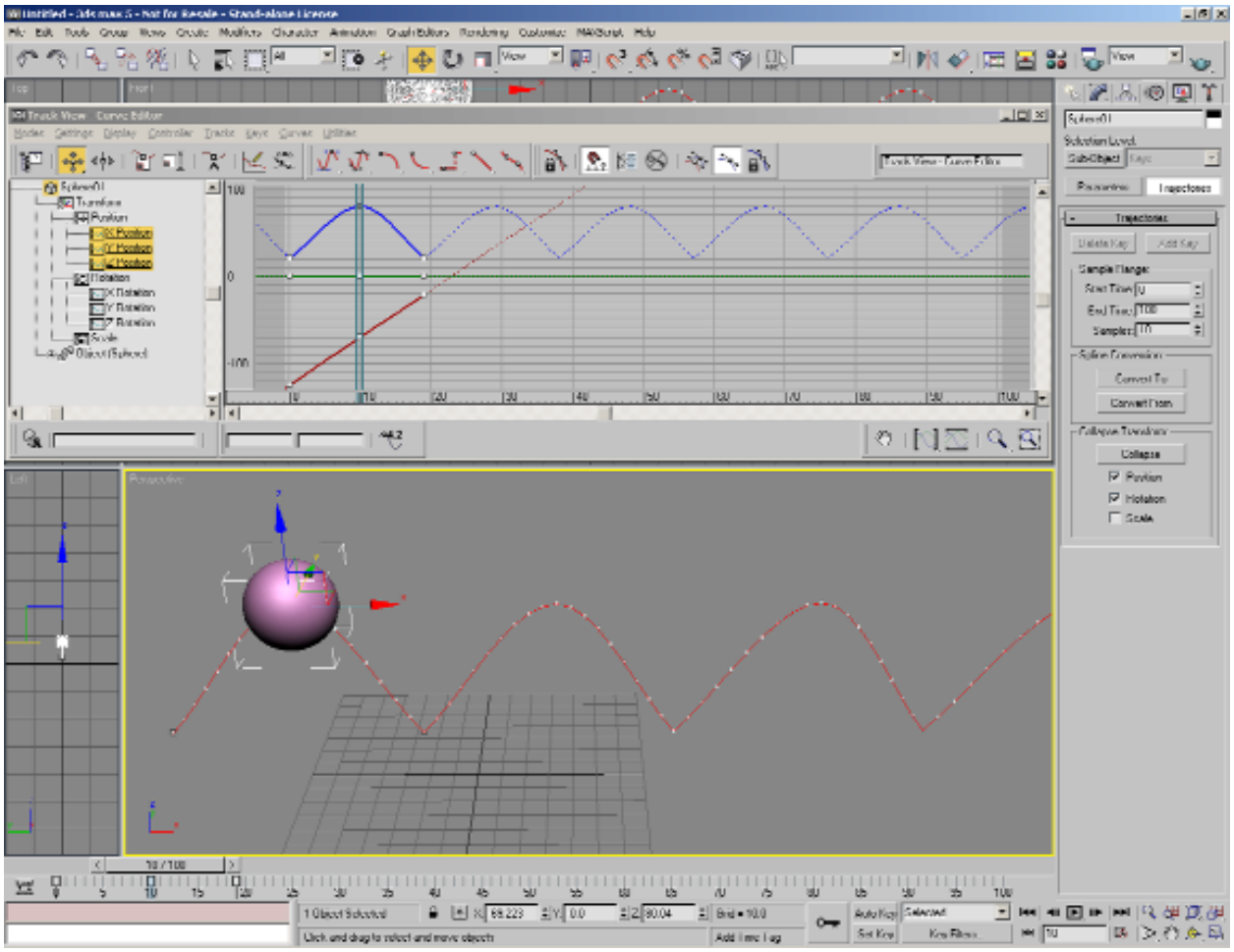
You can also utilize some of the automated unwrapping features by selecting the UVs you want re-mapped, go to the Mapping menu in the Edit UVW dialog. From here, there are three options for automated unwrapping. Sometimes these will give you almost exactly what you want if you play with their settings. Often times though, you will end up doing some portion of the work manually.

UV EXPORTING-TEXPORTER PLUG-IN

If you are establishing a new UV layout for a model with no textures for it, you will want to use this layout as a template for painting over in a paint program. You can do this by using Print Screen, or you can use a 3rd party plug-in to help you. Texporter is a free plug-in that allow you to save the UV template in a variety of formats. It also gives you extensive control over exactly how the geometry will look in the template image.

Use one of these two methods for creating the UV image template. Paint over it in PhotoShop, save the new image, then use it as part of a Material which you can re-apply to the model. The paint strokes that you applied to the template's geometry will line up to the geometry of the model in viewport. Note that PSD files will auto-update in Max as you alter them and re-save them in PhotoShop.

LIGHTING BASICS/LIGHT BAKING



Animation Basics

Suggested reading: 3ds max Illuminated: Foundation Chapter 6, pp. 155-165

Although lighting in a game is generally handled by the game engine itself, it is a good idea to know how to setup basic lighting in Max. The reason for this is that Max's lighting is much better than most game engine's, and that you can bake this lighting into your models. So, when you bring the model into the game, it looks a little better lit than anything that the game engine could do on its own.

MAX LIGHTING BASICS

There is default lighting in every Max scene, just so that you can see what you are working on. This lighting is very poor however, and you can create your own custom lights, which will force the default lighting to be turned off. You can create lights from the Create Panel, and depressing the Lights button. Several light types are available.

Adjust the light's intensity by altering the Multiplier value. Adjust the color of the light by clicking on the small color swatch, and choosing a new one with the Color Picker. If desired, you can turn on shadows by placing a check in this checkbox. Usually strong shadowing should be avoided however, as it may be going in the wrong direction when placed in the game level. It is usually a good idea to use several lights to illuminate your model, each with a lower intensity to them.

GLOBAL ILLUMINATION

Global Illumination is the new rage in pre-rendered 3d. It is a more natural light that takes into account light bouncing around the scene, color bleed, and soft shadowing. This is possible to do in Max as well, and can be easy to setup adding a Skylight light type to your scene, and then turning on an Advanced Lighting solution known as Light Tracer. Light Tracer can be activated through the Rendering Menu→Advanced Lighting→Light Tracer.

The Skylight light type tries to simulate natural outdoor lighting. It is meant to work in conjunction with an Advanced Lighting solution. One thing to note is that Skylight's light source comes from every direction, as it would from the atmosphere on a sunny day. To get it to look right, you must make sure that there is a floor below the object in your scene, so that light doesn't emanate from below the object.

RENDER TO TEXTURE

Once a lighting solution is set up in your scene, you will need to bake that lighting information into the model. This may be done several ways, but a quick way to achieve this is to bake the lighting into the object's textures. From the Rendering Menu, you can access Render To Texture, which does this. When active, you can add a Complete Map, set the image size, ensure that the Map Channel is set to 1. Under the General Settings, ensure that Automatic Unwrap Mapping is off, as this will not use the UV layout that you established. With this all set, click the render button at the bottom of the dialog, and a new texture map will be created with the lighting information baked into the map. Save this map by hitting the save button in the corner.

When using this technique, you should avoid overlapping UV coordinates, as they can have inaccurate lighting on the render to texture. Also note that this texture will automatically be applied, and if you pick this material in the Material Editor, you will see that it is a unique material type. I generally re-create my material, using the new baked texture map.

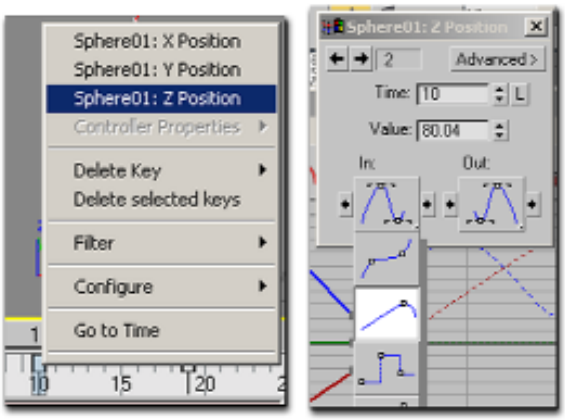
VERTEX LIGHTING

Instead of baking lighting into the texture, you can bake the lighting information into the vertices of the model itself. Most game engines can read this vertex lighting information, and will display it in the game. This technique works quite well with standard Max lights set up in the scene, then by using the Assign Vertex Color utility. Once vertex colors have been assigned, right-click on the object itself, go to Properties, and under Display Properties, turn off By Layer, then check Vertex Colors, so that they can be seen in viewport.

The benefits of vertex lighting are that you don't need to create more custom textures for each object in order to get better lighting. The drawbacks to this technique are that you need to have a denser mesh in order for the vertex coloration to gradiate well. You can also use the Vertex Paint Modifier to achieve a similar effect manually.

ANIMATION BASICS

Suggested reading: 3ds max Illuminated: Foundation Chapter 5, pp. 121-144



Keyframe Editing

Many things in a game need animation. This may include character animation, mechanical animation, or ambient movement of props, such as the branches of a tree swaying in the wind. To be an effective animator, you need to understand how animation functions, and the tools that are available to you.

KEYFRAMING AND IN-BETWEENING CONCEPTS

In traditional animation, a master animator would hand-draw the most important positions called keyframes of an animation. After that an assistant animator would fill in the gaps in between with more drawings. This is called tweening, and is something that Max creates, or interpolates for us in 3d. In both cases, the animation was simulated by creating a series of images, then showing them in rapid succession. This is essentially what a game engine does with content.

CREATING KEYFRAMES

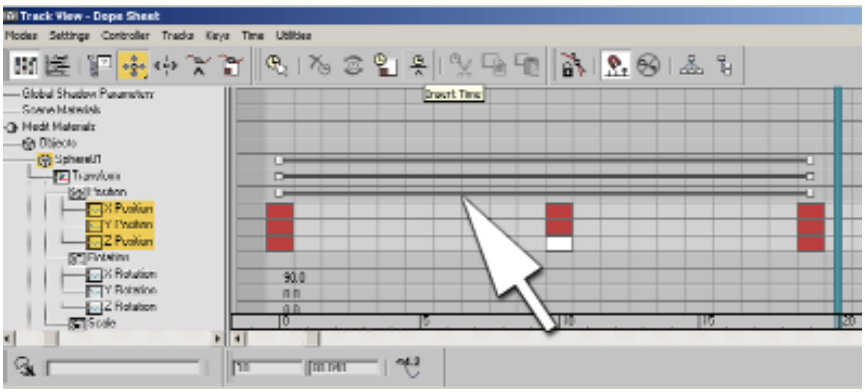
There are two toggles that allow you to create animation. The first is the Auto Key toggle, which is a hot toggle. When activated, any Transformation that you do to an object using Move, Rotate, or Scale will be keyframed. You can also use it to animate any spinner in Max. To use it, turn it on, move the time slider to the time when you want the animation to end, then use a transform to alter the object's position or rotation. A start keyframe is created automatically at frame zero in the TrackBar, end the ending keyframe is at the frame that you determined it. You can scrub through the animation by dragging the Time Slider, or by using the animation playback controls.

The second one is a warm toggle, called the Set Key toggle. When it is on, you can use the Transforms to alter the position or rotation of an object, but no keyframe will be set until you click on the large square Set Key button. No keyframe will be automatically set at zero, as you need to set each keyframe individually with Set Key animation mode. To determine which Transform you are setting keyframes for, click on the Key Filters button.

Viewing Trajectories

VIEWING TRAJECTORIES

It can be helpful to see the arc that your object is moving in, to better understand how it gets from point A to point B. This path is called a trajectory and can be viewed by selecting the object, going to the Motion Panel, and choosing the Trajectories option. A dashed line will indicate where the object is going in its animation. This is just a graphical cue, and cannot be selected or manipulated. This trajectory indicates the keyframes by little boxes, and Max's tweening calculations by the dashed line.



Ranges bars in Dope Sheet

KEYFRAME EDITING

Usually any animation that you set up may need some editing, which can be done in the Track Bar. You can move keyframes in the TrackBar to alter start times, end times, or the overall length of the animation. Just click and drag on a keyframe to move it, or click and drag around several keyframes to affect them all at the same time. If you hold down Shift and Move a keyframe, then that key will be duplicated. Selected keyframes will be white in the TrackBar. They can be deleted by hitting delete on the keyboard.

Once you've got the keyframes themselves correct, you may need to adjust the in-betweening that Max had interpolated. Right-click on keyframes to access the animated Position, Rotation, or Scale information. The Edit Key Info dialog will appear, and at the bottom will be a graphical button showing the In/Out tweening that Max is currently using coming into and going out of that keyframe. Click and hold on the buttons to alter the tweening. This type of editing can also be done from the Motion Panel→Parameters→Key Info when you have moved the Time Slider to the keyframe.

EXPANDED TRACKBAR (MINI CURVE EDITOR)

Another quick option to use for adjusting keyframes and their tweening is to click on the small button at the left side of the TrackBar. This expanded TrackBar shows the interpolation/function curves, and the keyframes for the animated object. You can click on the keyframes, and use the Bezier handles to adjust the interpolation between the keyframes, or you can change out the interpolation to a different solution by selecting the keyframes, and choosing one of the Set Tangent options.

TRACK VIEW EDITING: CURVE EDITOR

If you are feeling a little cramped in the Mini Curve Editor, you may want to use the full sized Track View: Curve Editor. You can access this from the Main Toolbar, or from the Graph Editors menu. Just as in the Track Bar, you can change Tangents, right-click on keyframes, move them to different points in time, Shift-Move copy them, etc.

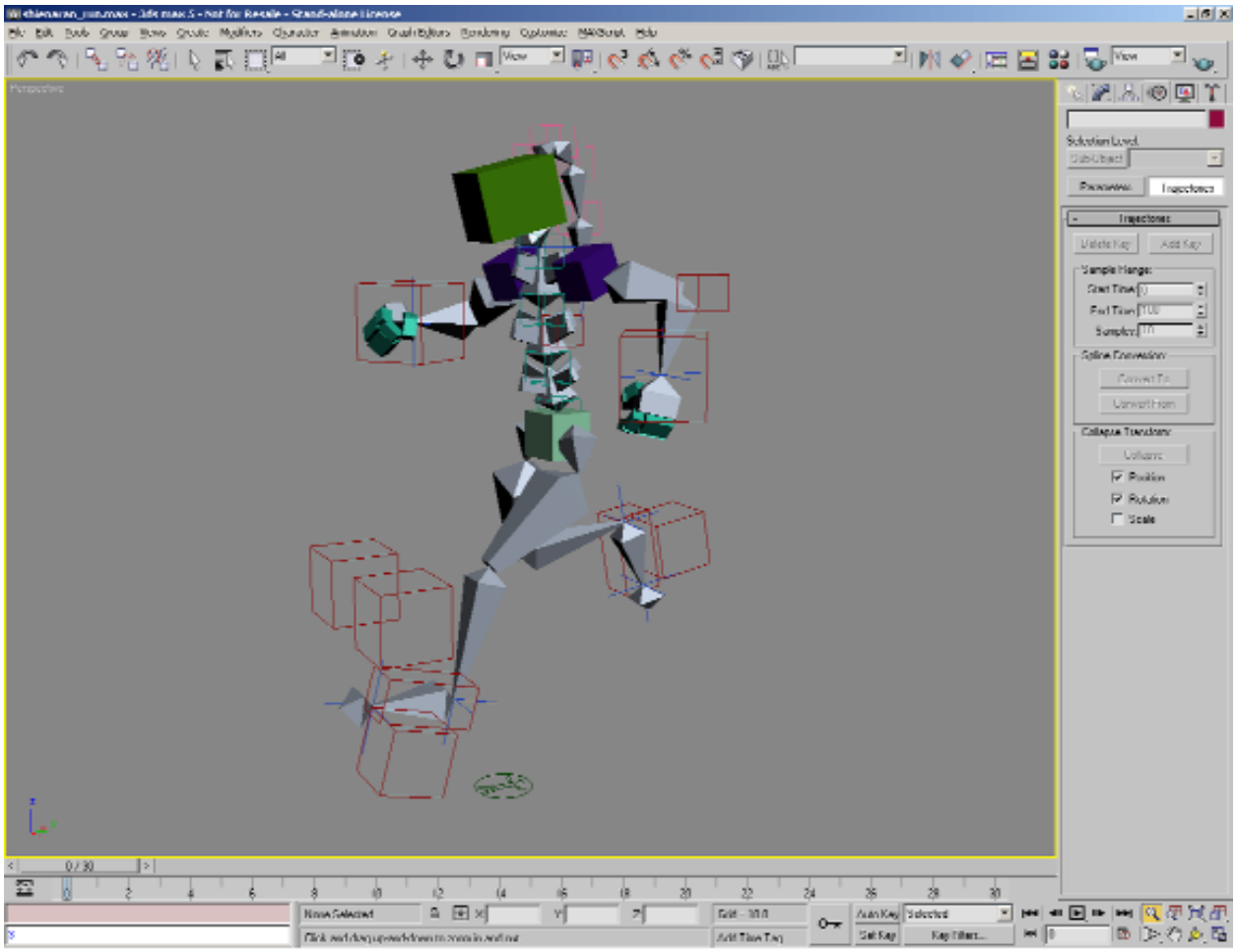
TRACK VIEW: DOPE SHEET

There is another type of Track View editing mode available to you if you bring up the Curve Editor, then go to Modes→Dope Sheet. One of the most important aspects to the Dope Sheet is access to the range of animation. In Dope Sheet, the keyframes of the animation are still visible, and look somewhat like they do in the Track Bar. Just above these keyframes is a black bar that indicates the range or length of the animation. If you want to change when the animation takes place, you can move your mouse over the middle of the range bar, then click and drag. If you need to shorten or lengthen the timing of the animation, you can move your mouse over the box at the start or end of the range bar, and move it.

TIME CONFIGURATION

Just next to the viewport controls near the animation playback options is a button for Time Configuration. Clicking on this brings up a dialog for adjusting how much time is available for an animation, the framerate, etc. The range of the animation can be affected here as well, by re-scaling the time.

LINKED HIERARCHIES



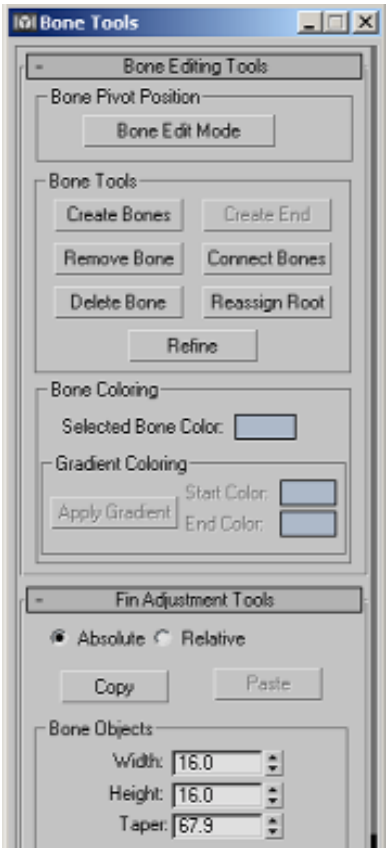
Linked Hierarchies

Suggested reading: 3ds max Illuminated: Foundation Chapter 8, pp. 213-228

Linking objects is a way to get one object to follow along with another. This creates a hierarchy, in which there is a parent-child relationships. The reason for doing this is so that you can have more sophisticated animation control over the objects. For instance if you want to animate a plane flying around, then you would want the propellers to go wherever the body of the plane went, but should be able to spin independently from the body of the aircraft. This could be achieved by linking.

LINKING VS. GROUPING

Grouping objects together is not the same as linking, and is for temporary organization of the scene only. Linking establishes Parent-Child hierarchical relationship, in which the child will inherit any transformation from their parents. Unlike a group however, these children may have additional animation applied to them, which will not affect their parents.



Editing Bones with Bone Tools

CREATING A LINKED HIERARCHIES

Creating a Linked Hierarchy can be done by using Select and Link from the Main Toolbar. Click and drag from the object that is to be the child to the parent object, and the objects should flash in white to indicate a successful link. Note that the objects must be independent of one another initially, and should not be part of the same mesh. When linking, be careful to move your mouse over the wires of the objects if linking in a wireframe view. Objects can be linked to only one parent. If you make a mistake, you can click on Unlink Selection to destroy the link. Also note that if Select and Link is the active tool on the Main Toolbar, the Select by Name dialog becomes Select Parent, allowing you to link via selection from the list.

MANAGING HIERARCHIES

If there are linked objects in your scene, you can see the hierarchy by activating the Select Tool, then choosing Select By Name. When the dialog appears, place a check next to Display Subtree. Objects in hierarchies will be indented below their parents.

Another option is to use the Schematic View from the Main Toolbar. From this view, you can see the hierarchy, and have the ability to link and unlink hierarchies within the dialog. To make it easier to keep track of what you are viewing, you might want to use the filters to hide some information when it is not needed.

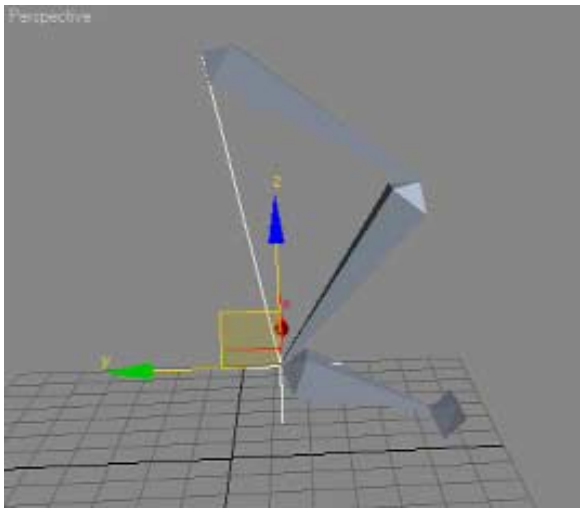
ANIMATING IN FORWARD KINEMATICS

Forward Kinematics, or FK, is a method for animating hierarchies that starts with the parent, and works down the hierarchy. In the case of the plane example, you would link the propellers to the body of the plane, so that the body is the parent. You would then use Transforms to animate the movement of the body of the plane, which causes the propellers to follow. Then you could animate rotation of the propellers, so that they spin. This is a very straight-forward way to animate, and is very effective in many cases. The only catch might be that objects Pivot Points might be out of position, making for poor rotation. This can be fixed by using Affect Pivot Only from the Hierarchy Panel.

Using Envelopes and the Weight Table to assign proper influence

SEGMENTED MODELS VS. CONTIGUOUS MODELS

If a model is made up of various different component objects (segmented), you can establish your hierarchy, then directly animate the different pieces. If however, the model is one contiguous mesh, you will be unable to transform the separate pieces. In this case, the model will need to be deformed. For newer games, character animation is usually handled using this type of deformation.



Solving using IK

BONES

To deform a contiguous mesh, it is common to use an underlying hierarchy that will serve as a skeleton that drives the animation, and resulting deformation. You would just work on the animation of the skeleton, which will in turn cause the mesh to deform. You can use a hierarchy or linked primitive objects, other objects, or a special type of object called Bones.

Bones can be created by going to Create Panel→Systems→Bones. Bones are a special type of linked hierarchy that has some unique capabilities. First and foremost is the fact that they are already linked up for you. You can also affect their size, and give them fins that can serve as visual cues when animating.

BONE TOOLS EDITING

Bones are a special type of geometry, and must be edited a little differently than other objects in Max. You should avoid using the Scale Transform on Bone objects, as it can have a bad effect on the overall Bone hierarchy. If you need to resize bones, you can go to the Character Menu, and choose Bone Tools. From this dialog, you can activate Bone Edit Mode and then move the Bones to re-scale them. Be sure to turn off Bone Edit Mode when you are done. In addition, you can alter the width, fin adjustments, etc., of multiple Bones at the same time from the dialog.

USING INVERSE KINEMATICS SOLUTIONS

Inverse Kinematics or IK is another animation method, which allows you to move the child in a hierarchy and causes the parents to get dragged along. This method works similar to a puppeteer working a marionette. Pull the string attached to the hand, and the whole arm moves. In Max, the animator needs to set up the IK "strings" by establishing an IK solution for the limbs.

To create an IK solution for a limb such as a leg hierarchy, you will select the thigh, choose Animation Menu→IK Solvers→HI Solver. At this point, a dashed line will be running to your mouse, indicating that it is ready to solve. Move your mouse down past the shin, and click on the foot in the hierarchy. A point helper (the goal)will be created at the heel of the foot, and you can move the entire leg by moving this IK chain goal. Also notice that if you now move the thigh bone, the heel tries to stay pinned down to the goal, but the shin bone and foot bone cannot be moved, as they are controlled by the goal.

You can use several different IK solutions within the skeleton to get the type of control that you want over the hierarchy. To adjust the rotation of the IK goal, and its influence, you will need to select the goal, and go to the Motion Panel. Under the IK Solver rollout, there is a control for the Swivel Angle. This may be affected by using the spinner, or by adding another object the scene and picking that object for the goal to point at.

USING CONTROL OBJECTS TO ANIMATE IK RIGS

Sometimes the IK chain goals can be a little difficult to see and deal with. You may want to link the goals to other objects, so that the IK rig is easier to select and visually understand. The most commonly used control objects that artist will use are helper object, and Spline objects. Both can be found in the create panel.

DEFORMING A MESH USING THE SKIN MODIFIER

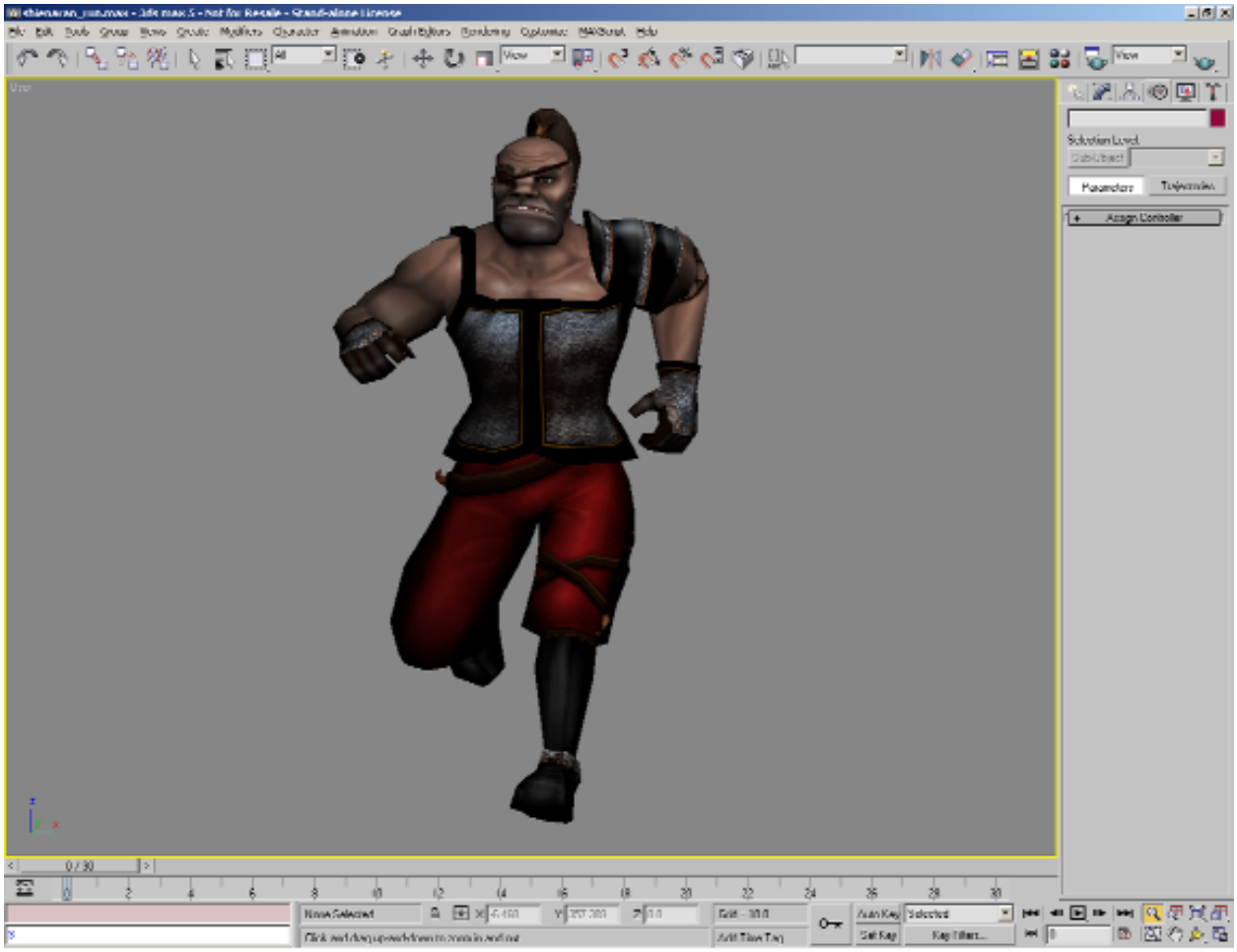
To get a model to deform properly, you will generally make a Bones skeleton that fits snugly inside the mesh. After that, you will apply the Skin Modifier to the mesh, and Add the bones to the influence of the model. Once this is done, the bones will deform the mesh, but there may be some inaccuracies in their influence.

You may need to alter the influence of the bones over the vertices in the mesh. If you click on Edit Envelopes, you can see the influence that each bone has over the mesh. Use the control points to re-size the envelope. If necessary, Add Cross-Sections so that you have more control. There is also an option for Paint Weights, which allows you to add influence to a specific portion of the mesh, and there is a Paint Weights setting buttons that let you fine-tune the settings of this tool

You can also choose to select vertices within the Skin Modifier. This allows you to easily find them within the Weight Table. This spreadsheet-like table allows you to manually enter the influence of bones over the vertices of the mesh.

Depending on the game engine, you may need to make sure that the vertices for the mesh are set to Rigid, meaning that they are only affected by one bone. Do this by selecting the vertices, and checking the Rigid checkbox.

CHARACTER ANIMATION



Character animation

Character Animation really is an art unto itself. To be able to animate well, you must have a solid understanding of motion, and how living beings operate. Once you understand how things work in the real world, you may want to exaggerate some of these aspects to make your character animation look more dynamic. In addition to these skills, you should learn some tricks such as loopable animation, which is more specific to character animation for games.

Characters should appear as if they have real weight to them. They should not float across the ground if they have some bulk to them. As a character moves, it settled downward, planting its feet. The timing of the animation is obviously important as well, as animation may not look as good if it is too fast or two slow. The viewer should also be able to see what is going to happen a second ahead of time. This can be thought of as wind-up, or anticipation. At the other end of the animation would be follow through. Characters animation should not just stop abruptly. This is unnatural, as people in the real world don't move like robots. There is a little bit of drift or follow through to their motion.

MORPH ANIMATION

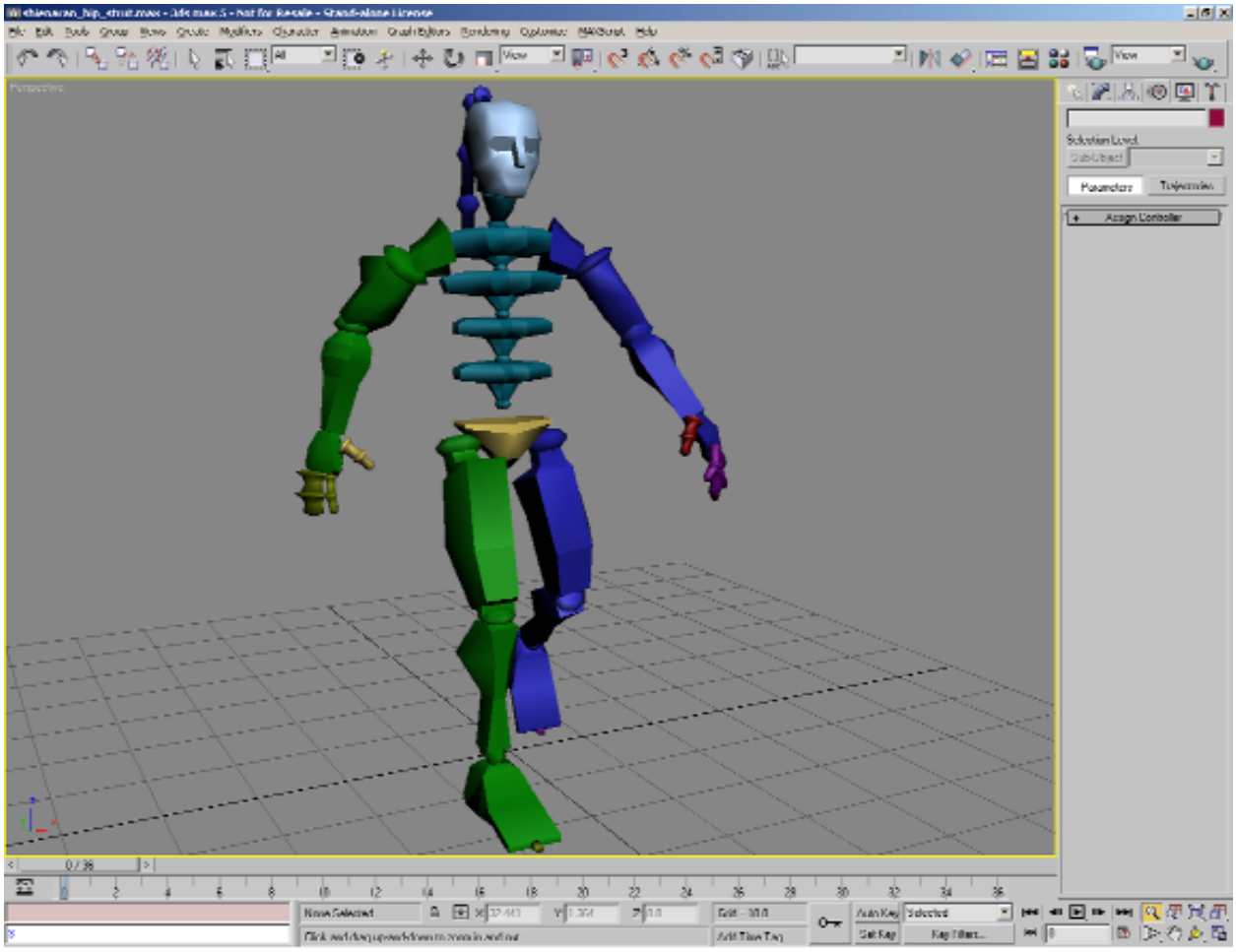
Most games utilize some sort of Bones driven animation for objects, but some support another method called morphing. Morphing is the change in one object's shape to that of another. In Max, both objects must have the same number of vertices to morph properly. Usually this means creating at least two variations of the model. The first is the model in the original state, and the second is the deformed variation. You can then apply a Morpher Modifier, and load up the deformed variation as a morph target. A spinner controls the amount of influence that the target has on the shape of the original.

LOOPED CYCLES

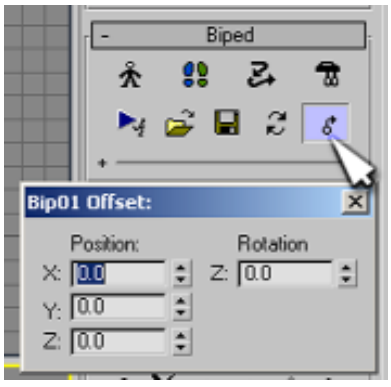
In games, it is unusual to perform some long and complicated animation, as the number of keyframes could slow down the game when it is running. Rather, artists break animations down to small sections that can be repeated back to back. These are known as looped cycles. A common looped cycle might be a character's two-step walk cycle, which can be repeated by the game engine.

Whenever your are creating an animation that needs to be loopable, the key thing to remember is that the beginning and end of the animation need to be exactly the same. Generally you will want to select all of the keyframes for the object or hierarchy at the start frame, and Shift-Move them to the end of the animation, which will copy them. This ensures that the beginning and the end of the animation are the same, so that there will be no pops in the motion when it is repeated.

CHARACTER STUDIO BASICS



Character Studio



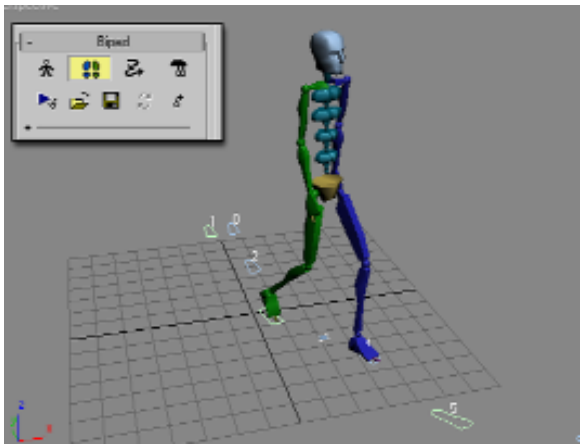
Move All Mode

Character Studio is a piece of add-on software known as a plug-in for 3d Studio Max. What it does is supplements Max's character animation toolset. Although it is possible to animate a character without the use of Character Studio, the plug-in can be a real time saver. Character Studio consists of 2 parts, the first being the Biped skeleton, and the second being the Physique Modifier, which is similar to Skin.

BIPED

Biped is a pre-rigged skeleton that that allows you to use both Forward and Inverse Kinematics on it. You do not need to establish IK solutions manually, as this is done for you. You can create a Biped by going to the Create Panel→Systems→Biped.

Once a Biped is in your scene, you will want to select it and go to the Motion Panel. Biped's controls are accessed via this Panel. You will need to make sure that the Biped fits in your character mesh, and you must be in Figure Mode to do this. Once in Figure Mode, you will see a Structure rollout appear, which allows you to alter the number of bones in the skeleton. You can also use the Scale Transform directly on the parts of the Biped to get them sized correctly.



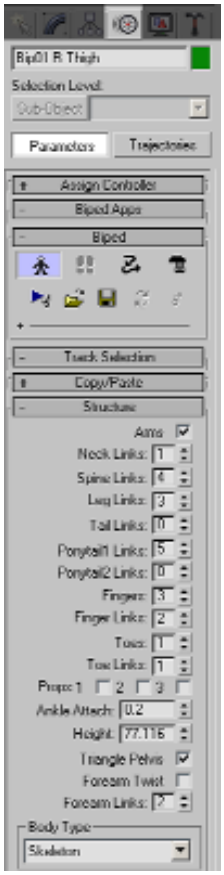
Footstep Animation

FOOTSTEP ANIMATION

Biped allows for a unique method of animation known as footsteps. This method allows you to place markers that the character will follow. You can do this by activating Footstep Mode, and Creating Multiple Footsteps. After this, you only need to Activate the Inactive footsteps. Generally footstep animation is a nice start, but you will need to supplement it with some old fashioned keyframing. When you create keyframed motion by hand in Character Studio, it is known as freeform animation.

PURE FREEFORM ANIMATION

Freeform animation may be used in addition to footsteps, or may be done on its own with no footsteps. You can set keyframes for the various parts of the skeleton by activating AutoKey, or by expanding the Key Info rollout in the Motion Panel, and clicking on the Set Key button. Make sure that you are not in Figure or Footstep mode when you do this. You will also see that there are options for Set Planted Key, and several others. These are different types of IK keys, which will keep the selected extremity pinned down via IK as you animate the rest of the Biped's body. These are useful keys for doing things like walk cycles.



Setting a Biped's structure in Figure Mode

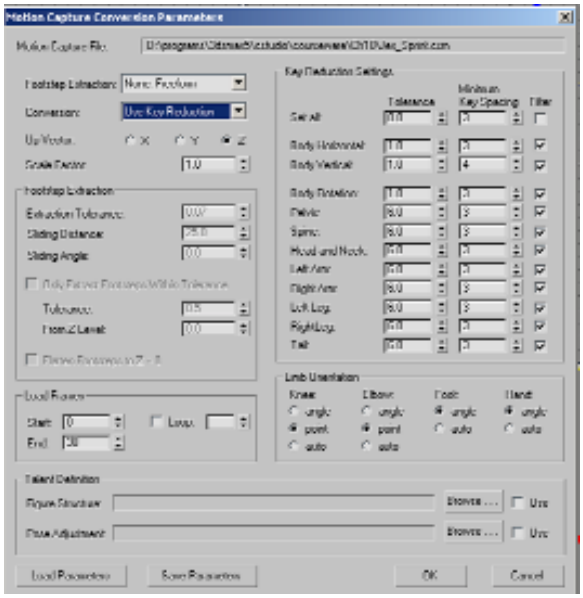
There are a number of other useful controls under the Keyframing Tools rollout. Here you will find controls for locking Hands/Feet into their current position. These will not set keyframes, but will temporarily keep them in their current position, allowing you to work on the rest of the body.

LAYERS

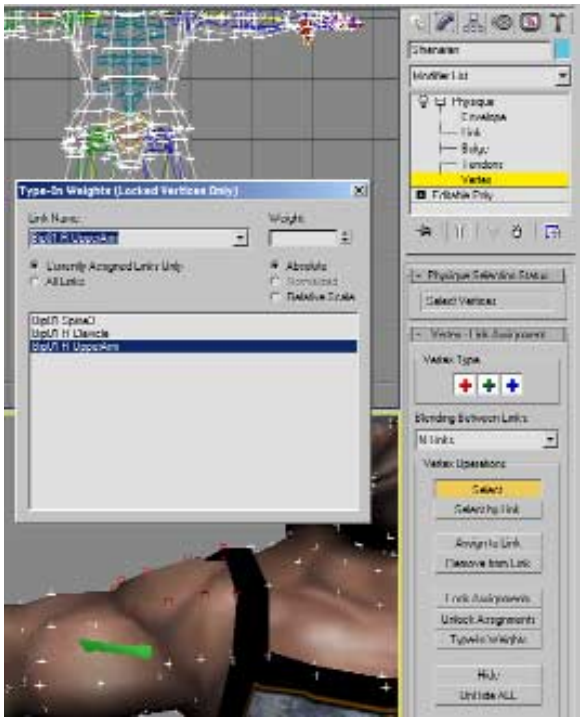
Layers is a method for supplementing animation without destroying the underlying animation. If you have an animation of a character walking, and then decide that his arms need to swing wider like a gunfighter, you could do this through Layers. Add a new layer, and rotate the arms outward. Set a keyframe at the start of the animation for this arm rotation. Now collapse the layers to condense this new customization into the original animation. As the arms swing, they will be kept wide throughout the animation.

BIPED TOOLS

Under the Biped rollout, there are several useful tools. From here, you can save animation that may be loaded onto another Biped. You can also load animation from this rollout. You can convert footstep animation to freeform and vice versa. There is also a control called Move All Mode, which allows you to re-position and orient a character that has already been animated to a new location.



Importing Motion Capture



Physique

USING MOTION CAPTURE

Motion capture is a method of recording the movement of a real person, and then applying the digital recording to a Bones skeleton in 3d. Biped allows you to import motion capture through the Motion Capture rollout. When you bring this animation into Max, it will bring in a keyframe for every frame of animation. This is virtually impossible to edit, and should be avoided. Luckily, Character Studio allows you to turn on an automated keyframe reduction when importing motion capture data, making the resulting animation much more manageable.

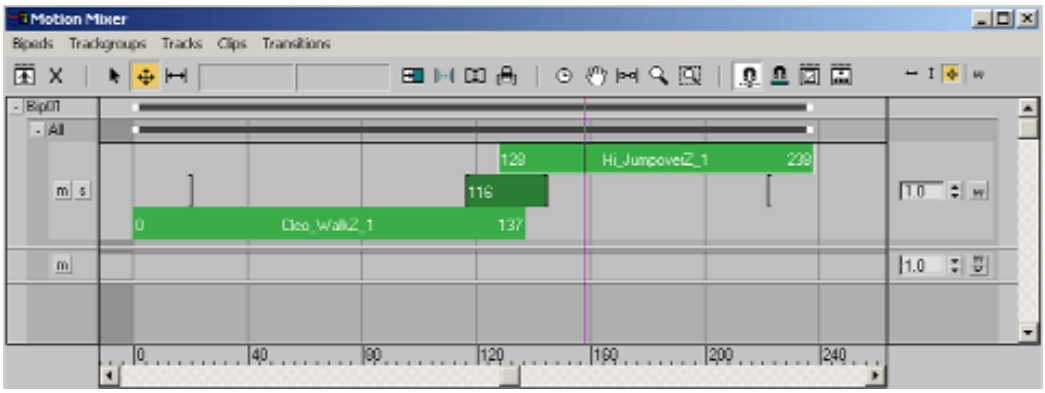
MIXER

The Mixer can be accessed under the Biped Apps rollout, and allows you to combine multiple clips into a new animation. It works similarly to a non-linear video or audio editor in that you can add clips, control the transitioning between clips, etc. You can also add Trackgroups with animation for just certain portions of the skeleton, so that one part of the Biped will be calling one animation, and the other portion of Biped will be using a different animation. Once you have the animation arranged as you like, you must compute the Mixdown (compile the animation) and apply it to the Biped.

THE PHYSIQUE MODIFIER

The Physique Modifier is quite similar to the Skin Modifier that is native to Max. Both allow you to attach a mesh to a Bones or Biped hierarchy, so that it is deformed. Physique allows you to control envelopes' radial scale, parent and child overlap, and cross sections. You can also move individual Control Points, which will not affect the rest of the Cross Section.

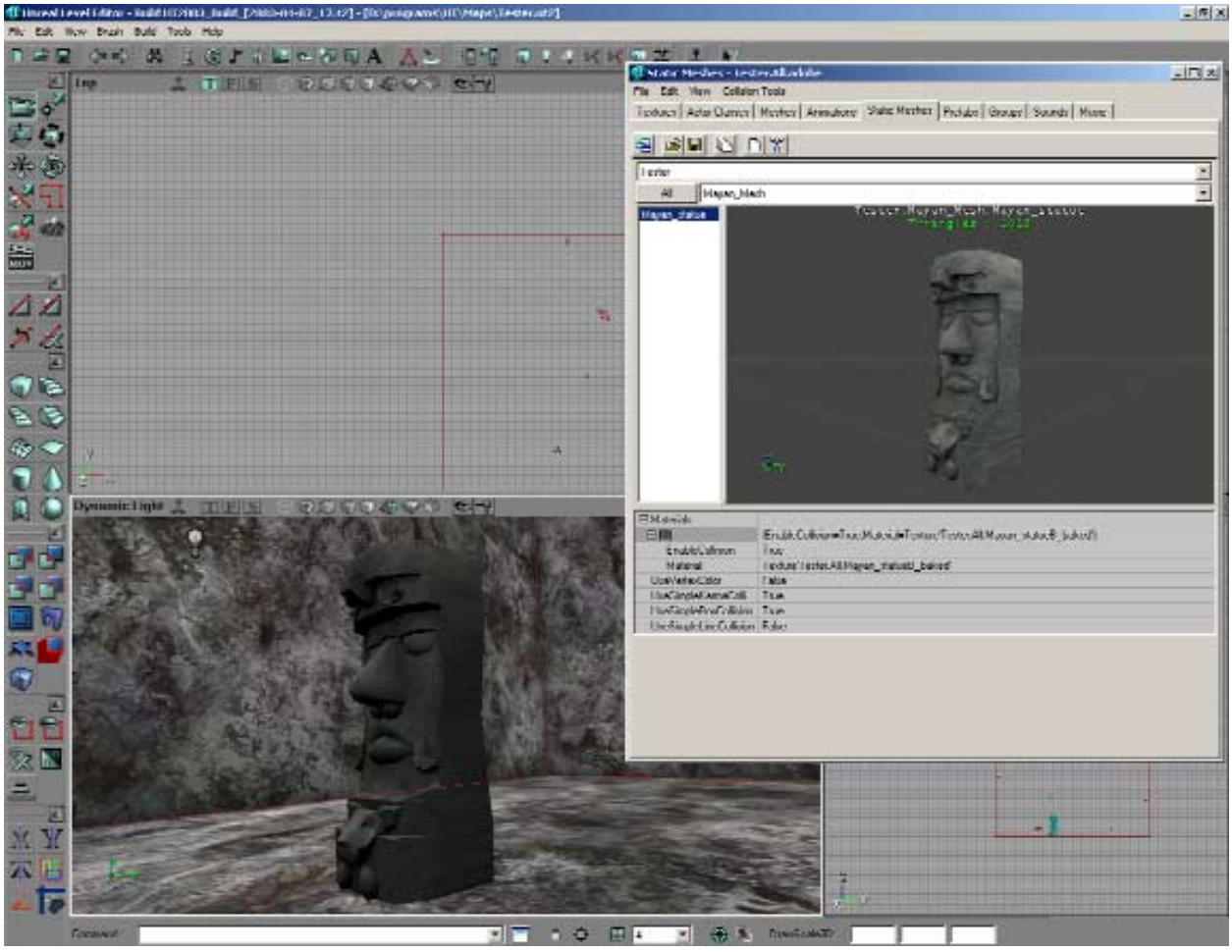
Vertex Sub-Object controls are also available, allowing you to select, lock, or manually assign vertex weights, so that the proper bone influences them.



Using the Mixer to combine animation clips

Set Key options for Biped

EXPORTING TO A GAME ENGINE



Exporting to a Game Engine

Each game makes use of Max data a little differently, and you should be aware of the game engine's requirements before hand. You must know the type of model that the game can accept, how it should be named, how big it should be, etc. In this section topics will be covered in relationship to Unreal technology, but such information will vary if you are using a different engine.

PROPER NAMING AND SCALE

In Unreal Tournament 2003, the average UT Characters = 96 Max (generic) units. Use this as a frame of reference when modeling. Ninety six units in Max is equal to about six and a half feet in Unreal Units. The requirements for naming are flexible for scenery objects, but can be more specific for Bones/Biped hierarchies. Consult the Unreal technical documentation from this book, or from Unreal Developer Network, or from other internet based Unreal sites.

SCENERY OBJECTS - STATIC MESHES

For scenery objects, scale should be taken into account, the model should be clean, and the Modifier Stack should be collapse. Airtight meshes are preferable, but don't need to be contiguous. To export, go to the File Menu→Export, and for Unreal, choose the .ase (ASCII) format. Name the file without spaces, click save, and check for any error warnings.

The .ase file can then be brought into unreal Ed via the Static Mesh Browser. You will generally want to bring it into a new package. Place the object in your scene. The object will be placed with no textures. Next, bring up the Texture Browser, create a new package, and bring in the texture that you want to use. Assign the texture to the object, and note that the texture will be aligned in accordance with the mapping coordinates that you established in Max. Save your packages, and re-build the scene.

CHARACTERS — ANIMATED MESHES

As stated earlier, Bones Hierarchies will need to be named appropriately ahead of time. Exporting out of Max may require a plug-in that will be able to export the information properly. Scale of the mesh should be in accordance with Unreal Standards, and the polygon count can be in the range of 2000-4000. Generally, one or two texture maps should be applied to the character, with possible team color variations.

CONCLUSION

In this course, the students learned about the tools that are available to for game production in 3d Studio Max. They learned about the interface, planning, modeling, texturing and UV mapping, light baking, animation, and exporting to a game engine. The tools that were introduced are some of the tried and true methods for game art creation. They are now acquainted with these tools and know how to use them effectively. In addition they should now understand some of the reasons why game artists work in the manner that they do. They should be able to apply this knowledge to creating high quality game artwork quickly in Max.

3.4 Quizzes and Projects

Quiz Answers

Quiz 1 Answers

1. A. You can click and hold on a flyout button to access more options.
2. C. Most commands can be accessed from the Menu Bar, even though they may be accessible elsewhere in the interface as well.
3. C. Transform gizmo is the correct term for this useful tool.
4. B. The Status Area at the bottom of the screen allows for type-in control.
5. A. Proper naming is very important when working in a team environment.
6. C. You can access Transforms from the Quad Menu
7. A. The Gizmo will look different for each Transform.
8. A. You can Scale an object on one, two, or three axes.
9. C. It is a set of tabbed panels at the right side of the screen.

C. Snaps can help with precision work.

Quiz 2 Answers

1. A. Primitives are parametric objects, whose dimensions are controlled by spinners.
2. B. Primitives can be created from the Create Panel.
3. C. Objects must have sufficient segmentation so that they can be deformed.
4. B. Edit Mesh is a modifier that allows you specific Sub-Object controls which need to be executed manually.
5. C. Edge is a Sub-Object mode in Edit Mesh.
6. A. Show End Result allows you to preview Modifiers higher up in the stack.
7. B. Creating polygons in a Counter-Clockwise direction ensures that their Normals are facing forward.
8. A. You can force edges to be visible or invisible.
9. C. Normals indicate a polygon's display direction.
10. A. Elements are the correct term, and are also a Sub-Object mode in Edit Mesh.

Quiz 3 Answers

1. B. An object must be converted to the Editable Poly state.
2. C. You can choose Convert To: Editable Poly from the right-click Quad Menu.
3. A. Border is the correct term, and also a Sub-Object mode.
4. B. Editable Poly is a quad-based modeling toolset.
5. B. Ring and Loop add to your selection of edges.
6. C. The Cut tool cuts, slices, and dices.
7. C. Apply is an option for registering the command.
8. B. They are a type of line.
9. A. Right-click on the vertex, and from the upper left Quad, choose a new corner type.
10. A. They can be use to create high or low polygon count models.

Quiz 4 Answers

1. A. The Symmetry Modifier mirrors and welds a duplicate of your model, allowing you to just work on half of it.
2. A. Joints must have sufficient segmentation to deform.
3. B. The silhouette, or profile of the character must be defined by geometry.
4. A. A contiguous mesh consists of one connected element.
5. A. If the game engine accepts this type of mesh, you can save on polygon counts by deleting portions of the model that will not be seen.
6. C. STL check checks for modeling errors
7. B. Smoothing controls how polygons interact to light cast on them.
8. B. Autosmooth automatically assigns Smoothing Groups.
9. A. They will not blend together, and a crisp, hard edge will be visible between them.
10. A. A polygon can blend to several different Smoothing Groups.

QUIZ 5 ANSWERS

1. B. Hiding is a way to make an object invisible while you work on the rest of the scene.
2. C. Freezing an object prevents you from selecting an object.
3. C. You can hide and Freeze objects from the Display Panel.
4. B. A group is the correct term.
5. C. Polygon Counter is a Utility.
6. C. Levels of Detail is the correct term for less detailed variations of a model.
7. A. Both achieve the same thing.
8. A. The Pivot Point controls the rotation of the object.
9. B. You should Reset the Scale in particular, so that the scaling of the object is set to 100%, which is what is preferred by many game engines.
10. A. It is a Utility.

QUIZ 6 ANSWERS

1. A. The Material Editor is like a painter's palette, where you prepare the Materials.
2. C. Diffuse is the main color of a Material.
3. A. They may be composed of combinations of diffuse maps, bump maps, opacity maps, etc.
4. B. Bitmap is a picture, and the image may be in a variety of formats.
5. A. A Material Slot that is bordered in white is the one that you are currently working with.
6. A. You can drag and drop the sample sphere in the Material Slot onto an object in your scene.
7. B. The polygons corresponding to the different Materials in the Multi/Sub must have their IDs set.
8. A. It is a type of Material, and is different from a Standard Material.
9. B. It auto-updates as you make changes to your materials.
10. A. It can direct Max where to look for images.

QUIZ 7 ANSWERS

1. B. AutoKey is another animation toggle.
2. B. You can click and drag the Time slider to check your animation.
3. A. True, and this will change the timing of the animation.
4. B. AutoKey is the toggle that can automatically create keyframes.
5. A. You can animate any Transform.
6. C. The Motion Panel allows you to edit keyframes. Trajectories are just a visual cue.
7. A. The curves are representing the interpolation, which can be edited.
8. A. Shift Moving objects duplicates them in Max.
9. A. The range bar can be adjusted in Dope Sheet.
10. C. The Time Configuration dialog controls the length of time available.

QUIZ 8 ANSWERS

1. B. Linking establishes a Parent to Child relationship.
2. B. Children should be linked to Parents. The relationship is reversed.
3. B. The Pivots should be put in the correct location using Affect Pivot Only
4. B. You need to use an IK Solver, and animate the goal in IK.
5. B. Inverse Kinematics is the correct term.
6. B. It is an Inverse Kinematics Solver
7. B. The Skin Modifier will make the mesh stick.
8. C. A Morph Target is necessary for the process, and should have the same vertext count as the original mesh.
9. A. Both are effective methods for animating in Max.
10. C. A looped cycle is the proper common term for this type of animation

QUIZ 1: PLANNING/UI OVERVIEW

1. FLYOUT BUTTONS INDICATE THAT THERE ARE _____.
   a. more options available
   b. Modifiers present
   c. exporting options available

2. YOU CAN ACCESS NEARLY EVERY COMMAND IN MAX FROM THE _____.
   a. viewports
   b. Track Bar
   c. Menu Bar

3. THE GRAPHICAL INDICATOR THAT ALLOWS FOR PRECISE TRANSFORMATION OF A SELECTED OBJECT IS CALLED THE _____.
   a. Select Tool
   b. Main Toolbar
   c. Transform Gizmo

4. YOU CAN TRANSFORM AN OBJECT WITH TYPE IN CONTROLS BY EITHER RIGHT-CLICKING ON THE TRANSFORM, OR ACTIVATING IT AND TYPING IN THE _____.
   a. Command Panel
   b. Status Area
   c. Edit Menu

5. OBJECTS IN YOUR SCENE SHOULD ALWAYS BE NAMED PROPERLY.
   a. True
   b. False

6. TRANSFORMS CAN BE ACCESSED VIA THE MAIN TOOLBAR, THE KEYBOARD, AND THE _____.
   a. Command Panel
   b. Viewport Controls
   c. Quad Menu

7. THE TRANSFORM GIZMO CAN LOOK DIFFERENT DEPENDING ON WHICH TRANSFORM IS CURRENTLY ACTIVE.
   a. True
   b. False

8. SCALE TRANSFORMS ALLOW FOR SCALING ON MULTIPLE AXES.
   a. True
   b. False

9. THE COMMAND PANEL IS LOCATED AT THE _____ OF THE INTERFACE.
   a. bottom
   b. left
   c. right

10. YOU CAN USE _____ TO HELP AID YOU WITH MORE PRECISE TRANSFORMATION OF OBJECTS.
   a. edits
   b. merges
   c. snaps

QUIZ 2: SELECTION AND TRANSFORMATION

1. PRIMITIVE OBJECTS' DIMENSIONS MAY BE CONTROLLED BY _____.
   a. parametric spinners
   b. a control mesh
   c. Element Sub-Object mode

2. PRIMITIVES CAN BE CREATED FROM THE MODIFY PANEL
   a. True
   b. False

3. YOU CAN USE PARAMETRIC MODIFIERS SUCH AS BEND TO MODIFY A PRIMITIVE OBJECT IF IT HAS SUFFICIENT _____.
   a. Surface Properties
   b. Normals
   c. segmentation/mesh density

4. EDIT MESH IS A PARAMETRIC MODIFIER THAT GIVES YOU A GENERIC, OVERALL EFFECT.
   a. True
   b. False

5. EDIT MESH ALLOWS YOU TO ACCESS SUB-OBJECT MODES FOR VERTEX, _____, FACE, POLYGON, AND ELEMENT.
   a. point
   b. loop
   c. edge

6. THE _____ TOGGLE ALLOWS YOU TO PREVIEW MODIFIERS THAT ARE HIGHER UP IN THE MODIFIER STACK.
   a. Show End Result
   b. Vertex
   c. Preview

7. IN EDIT MESH, WHEN YOU CREATE POLYGONS MANUALLY, YOU SHOULD CREATE THEM IN A _____ DIRECTION SO THAT THEY ARE FACING FORWARD.
   a. Clockwise
   b. Counter-Clockwise
   c. Top to Bottom

8. YOU CAN MAKE EDGES IN A MESH VISIBLE OR INVISIBLE.
   a. True
   b. False

9. _____ DICTATE WHICH DIRECTION THAT A POLYGON IS VISIBLE FROM.
   a. Gizmos
   b. Elements
   c. Normals

10. IF THERE A MESH IS MADE UP OF VARIOUS COMPONENTS THAT ARE NOT PHYSICALLY CONNECTED, THESE PIECES ARE REFERRED TO AS _____ OF THE MESH.
   a. Elements
   b. Subdivisions
   c. Segments

QUIZ 3: POLYGON MODELING BASICS

1. EDITABLE POLY IS A MODIFIER.
    a. True
    b. False

2. YOU CAN CONVERT AN OBJECT TO EDITABLE POLY BY RIGHT-CLICKING ON THE MODIFIER STACK, OR FROM THE _____.
    a. Main Toolbar
    b. Create Panel
    c. Quad Menu

3. THE EDGES AROUND AN OPEN HOLE IN A MODEL CAN BE REFERRED TO AS A _____ IN EDITABLE POLY
    a. Border
    b. Gap
    c. Spline

4. EDITABLE POLY IS A TRIANGLE-BASED MODELING TOOLSET.
    a. True
    b. False

5. RING AND LOOP CAN AID IN YOUR SELECTION OF _____.
    a. vertices
    b. edges
    c. polygons

6. THE _____ ALLOWS YOU TO MANUALLY SLICE UP GEOMETRY.
    a. Extrude control
    b. Element Sub-Object mode
    c. Cut tool

7. IF YOU CLICK ON A SETTINGS BUTTON FOR A PARTICULAR FUNCTION AND BRING UP THE TYPE-IN FLOATER, YOU MUST HIT _____ OR OK TO REGISTER IT.
    a. Yes
    b. Perform
    c. Apply

8. SPLINES ARE A TYPE OF _____ IN MAX.
    a. Primitive
    b. line
    c. edge

9. YOU CAN CHANGE THE CORNER TYPE OF A SPLINE OR PATCH VERTEX BY RIGHT-CLICKING ON IT.
    a. True
    b. False

10. PATCHES CAN BE USED TO CREATE HIGH POLYGON MODELS.
    a. True
    b. False

QUIZ 4: ADVANCED MODELING TOOLSETS

1. THE _____ CAN AID YOU IN MIRRORING AND WELDING TWO HALVES OF YOUR MODEL.
   a. Symmetry Modifier
   b. Quad Menu
   c. Smoothing Groups

2. FOR CHARACTER MODELS THAT WILL BE ANIMATED, YOU MUST PUT IN SUFFICIENT SEGMENTATION AT THEIR JOINTS SO THAT THEY WILL DEFORM PROPERLY.
   a. True
   b. False

3. THE SILHOUETTE OF THE MODEL CAN BE FAKED WITH TEXTURE MAPS.
   a. True
   b. False

4. A CONTIGUOUS MESH CONSISTS OF _____.
   a. a single element
   b. multiple elements
   c. separate pieces

5. SOME GAME ENGINES WILL ACCEPT MESHES WITH OPEN EDGES, ALLOWING YOU TO DELETE PORTIONS OF THE MESH THAT WILL NOT BE SEEN, WHICH SAVES ON POLYGON COUNTS.
   a. True
   b. False

6. THE _____ MODIFIER CAN HELP YOU CHECK YOUR MODEL'S GEOMETRY FOR MISTAKES.
   a. Mesh Select
   b. Editable Poly
   c. STL check

7. SMOOTHING CONTROLS HOW _____ POLYGONS.
   a. to subdivide
   b. light affects
   c. to turn the edges for

8. _____ ALLOWS YOU TO SET SMOOTHING GROUPS, BASED OFF OF AN ANGLE THRESHOLD THAT YOU DETERMINE.
   a. STL check
   b. Autosmooth
   c. Retriangulate

9. IF TWO POLYGONS ARE ASSIGNED TO TWO DIFFERENT SMOOTHING GROUPS, YOU WILL SEE A _____ BETWEEN THEM.
   a. hard edge
   b. soft edge
   c. Spline

10. A POLYGON MAY BE ASSIGNED TO MULTIPLE SMOOTHING GROUPS.
   a. True
   b. False

QUIZ 5: SCENE MANAGEMENT

1. HIDING OBJECTS CAUSES _____.
   a. geometry to be deleted
   b. them to be invisible
   c. them to be un-selectable

2. FREEZING OBJECTS CAUSES
   a. geometry to be deleted
   b. them to be invisible
   c. them to be un-selectable

3. HIDING AND FREEZING CAN BE DONE VIA THE QUAD MENU AND THE _____.
   a. Modify Panel
   b. Utilities Panel
   c. Display Panel

4. A SET OF OBJECTS THAT CAN BE ALL BE SELECTED AND TRANSFORMED AT THE SAME TIME IS KNOWN AS A _____.
   a. Manipulation Set
   b. Group
   c. Basket

5. THE POLYGON COUNTER CAN BE ACCESSED FROM THE _____.
   a. Modifier Stack
   b. Select Menu
   c. Utilities Panel

6. VARIATION OF A MODEL AT PROGRESSIVELY LOWER RESOLUTION ARE KNOWN AS _____.
   a. Substitutes
   b. Large Overt Decoys
   c. Levels of Detail

7. CHOOSING CONVERT TO: EDITABLE MESH OR EDITABLE POLY, IS THE SAME THING AS COLLAPSING THE MODIFIER STACK.
   a. True
   b. False

8. BOTH THE TRANSFORM GIZMO'S POSITION, AND THE OBJECT'S ROTATION ARE BASED ON THE OBJECT'S _____.
   a. Pivot Point
   b. Rotator
   c. vertices

9. YOU SHOULD NEVER RESET AN OBJECT'S SCALE FROM THE HIERARCHY PANEL.
   a. True
   b. False

10. RESET X-FORM IS A UTILITY.
    a. True
    b. False

QUIZ 6: MATERIALS

1. YOU CAN PREPARE MATERIALS IN THE _____.
   a. Material Editor
   b. Material/Map Browser
   c. Material/Map Navigator
2. _____ IS THE TERM FOR THE MAIN COLOR OF A MATERIAL IN MAX.
   a. Bump
   b. Ambient
   c. Diffuse
3. MATERIALS MAY CONSIST OF MULTIPLE TEXTURE MAPS FOR VARIOUS ATTRIBUTES.
   a. True
   b. False
4. A _____ IS AN IMAGE THAT MAY BE USED AS A TEXTURE MAP WITHIN A MATERIAL.
   a. Procedural texture
   b. Bitmap
   c. map attribue
5. AN ACTIVE MATERIAL SLOT WILL BE FRAMED IN WHITE.
   a. True
   b. False
6. YOU CAN ASSIGN A MATERIAL TO AN OBJECT BY CLICKING ASSIGN MATERIAL TO SELECTION, OR BY _____.
   a. dragging and dropping
   b. adding texture maps
   c. using a Multi/Sub-Object
7. YOU MUST SET THE _____ BEFORE A MULTI/SUB-OBJECT MATERIAL CAN BE PROPERLY APPLIED TO AN OBJECT.
   a. Material Slot
   b. Material IDs of the mesh
   c. Bitmap's parameters
8. MULTI/SUB-OBJECT IS A _____.
   a. Material type
   b. Standard Material
   c. Texture Map
9. ACTIVESHADE IS _____ RENDERER.
   a. an animation
   b. an auto-updating
   c. texture map
10. BY CONFIGURING PATHS, YOU CAN TELL MAX _____.
    a. where to search for Bitmaps
    b. how to apply a Material
    c. to display textures in viewport.

QUIZ 7: ANIMATION BASICS

1. THE TWO TOGGLES THAT YOU CAN USE FOR KEYFRAMING ARE SET KEY MODE AND _____.
   a. Tangent
   b. AutoKey
   c. Animate
2. YOU CAN SCRUB THROUGH YOUR ANIMATION BY DRAGGING THE _____.
   a. keyframes
   b. Time Slider
   c. Dope Sheet
3. YOU CAN MOVE KEYFRAMES IN THE TRACKBAR.
   a. True
   b. False
4. THE SET KEY MODE TOGGLE AUTOMATICALLY CREATES KEYFRAMES FOR YOU.
   a. True
   b. False
5. YOU CAN ANIMATE BY USING THE ANIMATION TOGGLES IN CONJUNCTION WITH _____.
   a. the Transforms
   b. the Edit Menu
   c. hiding objects
6. YOU CAN EDIT KEYFRAME INFORMATION BY RIGHT-CLICKING ON KEYFRAMES, OR USING THE:
   a. Create Panel
   b. Trajectories
   c. Motion Panel controls
7. YOU CAN ADJUST THE INTERPOLATION OR IN/OUT TANGENTS BY USING THE MINI CURVE EDITOR.
   a. True
   b. False
8. YOU CAN HOLD DOWN SHIFT AND MOVE A KEY TO DUPLICATE IT.
   a. True
   b. False
9. DOPE SHEET ALLOWS YOU TO ADJUST THE RANGE OF AN ANIMATION.
   a. True
   b. False
10. YOU CAN CONTROL HOW MUCH TIME IS AVAILABLE IN THE TIMELINE BY USING THE _____ DIALOG.
    a. Edit Key
    b. Curve Editor
    c. Time Configuration

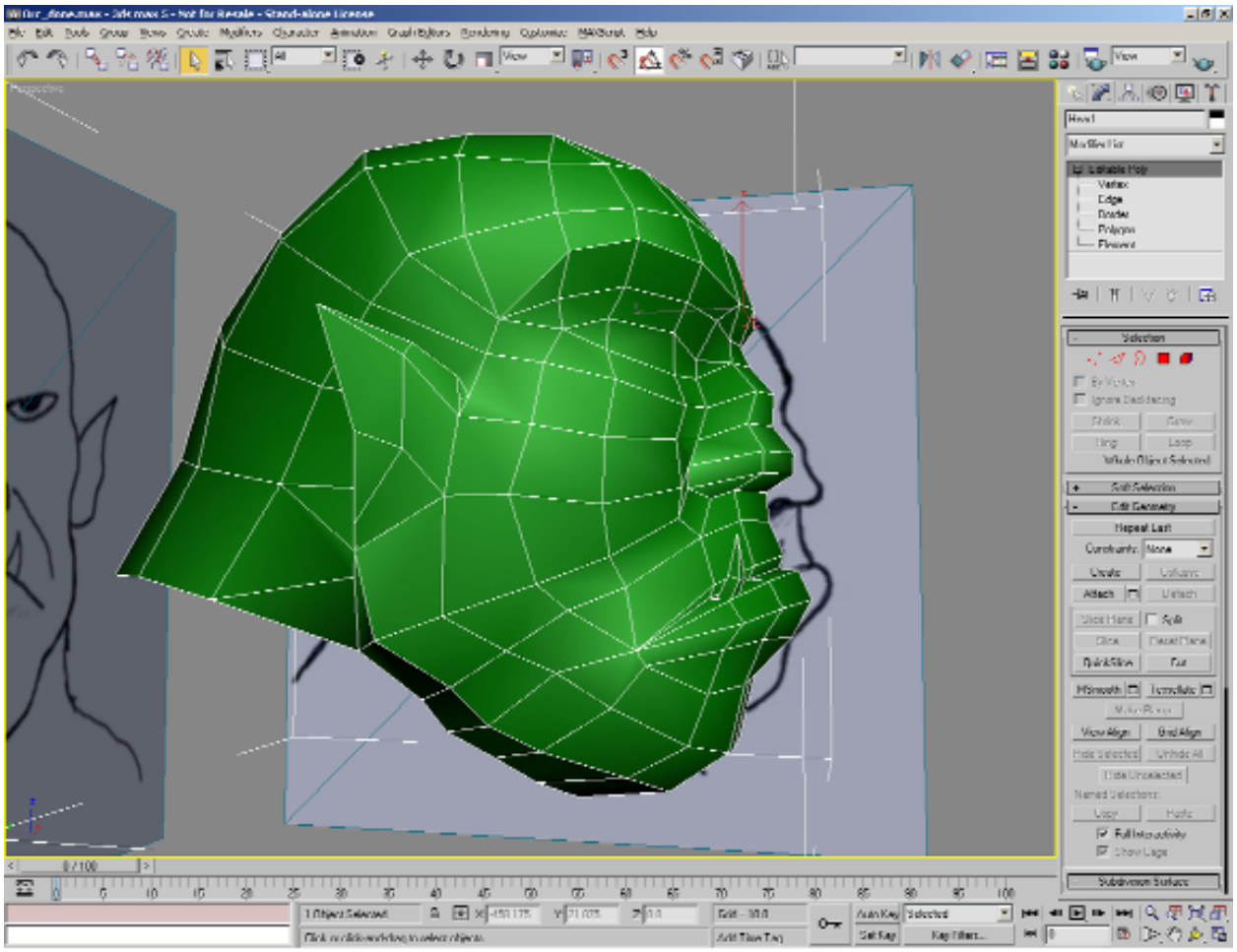
QUIZ 8: HIERARCHIES/CHARACTER ANIMATION

1. LINKING IS THE SAME AS GROUPING.
   a. True
   b. False
2. A PARENT OBJECT SHOULD BE LINKED TO THE CHILD USING SELECT AND LINK.
   a. True
   b. False
3. WHEN ANIMATING VARIOUS OBJECTS IN A LINKED HIERARCHY, YOU SHOULD MAKE SURE THAT:
   a. they are all the same size
   b. the Pivot Points are in the right spots c. you group them first
4. YOU USE THE SAME METHODS FOR ANIMATING WHEN USING FORWARD KINEMATICS AND IK.
   a. True
   b. False
5. IK STANDS FOR _____.
   a. Isolated Kinematics
   b. Inverse Kinematics
   c. Internal Kinematics
6. THE HI SOLVER IS USED TO CREATE FORWARD KINEMATICS
   a. True
   b. False
7. TO GET A MODEL TO STICK TO A BONES SKELETON AND BE DEFORMED BY IT, YOU CAN USE THE _____ MODIFIER.
   a. Mesh Select
   b. Skin
   c. Editable Poly
8. MORPHING ANIMATION REQUIRES THAT THE MESH HAVE A _____ TO REFERENCE.
   a. Material
   b. Skeleton
   c. Target
9. YOU CAN USE INVERSE KINEMATICS OR FORWARD KINEMATICS TO ANIMATE IN MAX.
   a. True
   b. False
10. AN ANIMATION THAT MAY BE REPEATED WITHOUT POPS OR STUTTERS IS KNOWN AS A:
    a. walk
    b. motion flow
    c. looped cycle

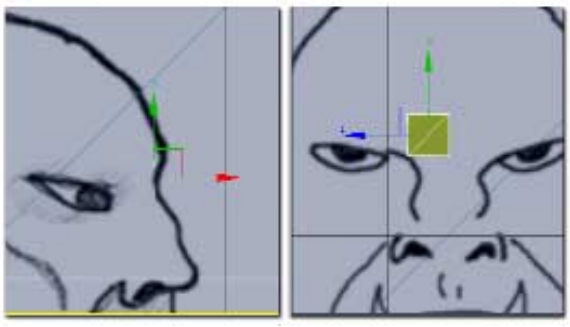
Project 1: Chessboard Tutorial, 3ds max Illuminated: Foundation p. 15

Project 2: Poly Modeling tutorial, 3ds max Illuminated: Foundation, p. 55

PROJECT 3: MODELING AND SMOOTHING - MODELING AN ORC HEAD



The finished head



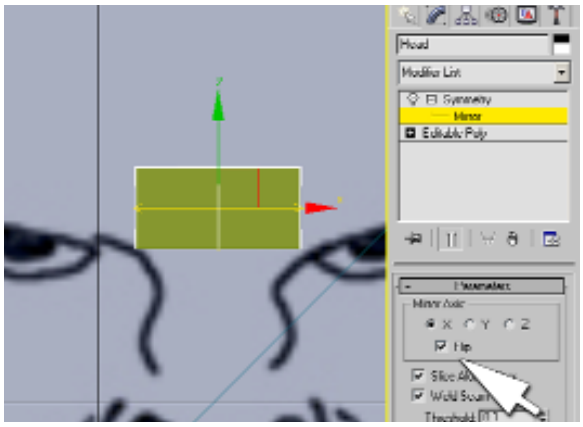
The plane object, in position

Files required: Orc_start.max scene file, Orc_front.jpg and Orc_side.jpg image files

INTRODUCTION

In this intermediate tutorial, you will be creating a low polygon count head model similar to the one shown above, using the Editable Poly toolset in Max. Topics covered will include: Edge Extrusion, Target Welding, Create Polygons, Cut tool, Hinge from Edge, Symmetry, and Smoothing Groups.

My method for doing head models is a little different from your standard box-modeled head. I feel that it is faster, less restrictive, and quickly lays out the geometry with very little excess which you would need to remove later. My method starts with reference imagery, which is extremely helpful when modeling. I define the profile of the model first, as this is one of the trickiest aspects to get right. I then define key features, fill in the gaps, and assign Smoothing Groups.



Adding Symmetry and adjusting the Mirror axis

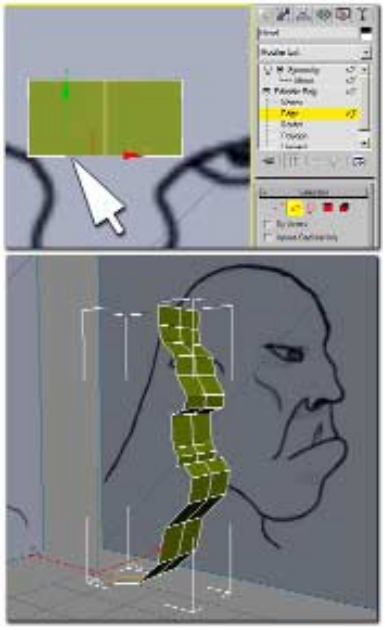
1. Open the file Orc_start.max. The scene consists of two plane objects with reference images mapped on them.

2. In the front viewport, create a small 1-segment plane object, and align it according to the following image. Name this object "Head".

3. Right-click on the Head plane object, and choose Convert To: Editable Poly. This will be the toolset that we will use for modeling.

4. Now add a Symmetry modifier to the object. This modifier mirrors the object and welds it down the middle, allowing us to just work on half of the model. If necessary, adjust the mirror axis, and check the flip checkbox to correct which direction it is mirroring. Click the plus button next to Symmetry, and turn on Mirror Sub-Object mode to move the mirror axis manually.

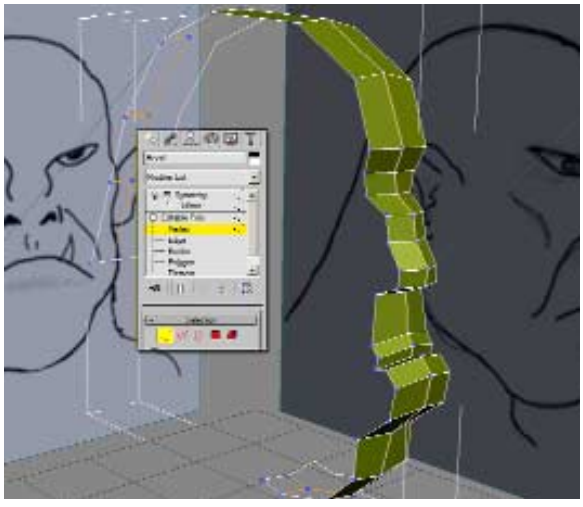
Show End Result toggle must be on to see the Symmetry.



Select the edge at the very bottom, and Shift+Move it to extrude it.

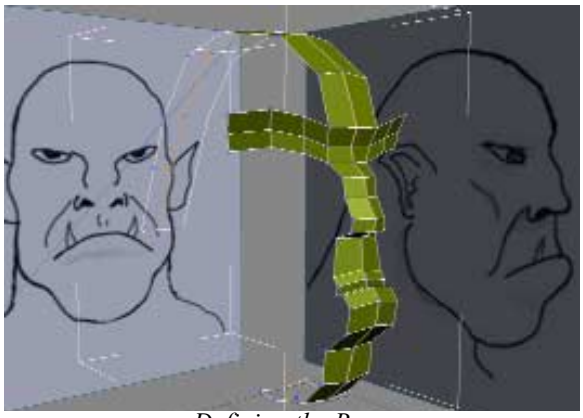
5. Exit Mirror Sub-Object mode, and click on the Editable Poly level in the Modifier Stack. Make sure that the Show End Result toggle is on, so that we can preview the results of the Symmetry.

6. Activate Edge Sub-Object Mode, and select the bottom edge of the Head Plane object. Right-click in the left viewport to activate it, then position the bottom edge so that it follows the profile of the reference image. Next, hold down Shift on the keyboard, and move the edge, which causes an extrusion. Position your mouse over the X,Y junction on the gizmo, and Shift+Move the edge repeatedly, tracing the reference image's brow, nose, lips, mouth, chin, and back around to where the jaw meets the neckline.

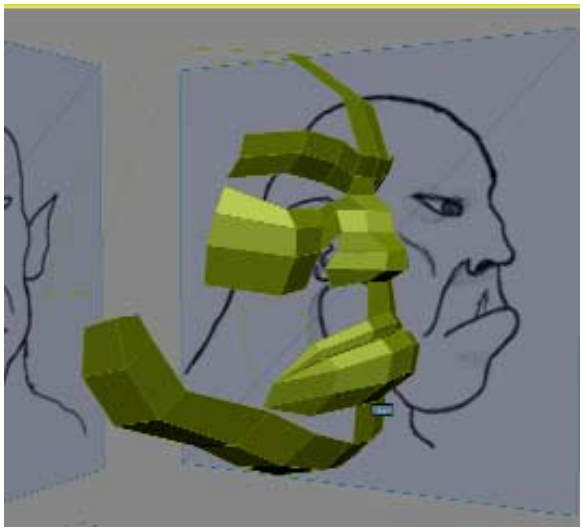


Move the vertices around, to give things some depth.

7. Select the edge at the top of the Head plane, and Shift+Move it over the top of the head, ending at the base of the cranium. What you will end up with is a clearly defined profile of the model.



Defining the Brow

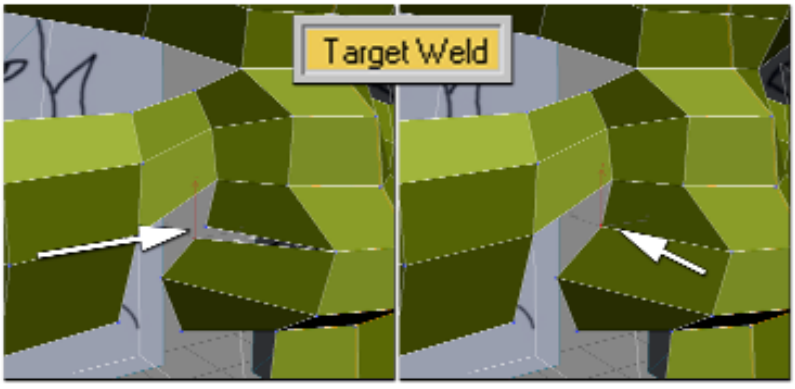


Defining key features with Edge extrusion

8. Now that you have defined the profile, switch to vertex Sub-Object Mode, and move the vertices around a little, so that they line up with the front reference image, and the nose has some depth to it. If desired, hit Alt/X on the keyboard to make the Head model see-through, and hit F4 to switch to Edged Face display mode. Also, be sure to check Ignore Backfacing so that you don't accidentally move vertices on the back-side of the head. Try to adjust the vertices so that they are positioned similarly to the following image. You MUST check your work in the Perspective view! Do not assume that things are in the right place if they are lined up in the two orthographic views. Arc Rotate (Alt/Middle mouse hotkey) around the object all of the time in the Perspective view, continually checking your work as you continue with this tutorial. Make adjustments whenever necessary.

9. Next, in the Front View, use the Shift+Move edge extrusion technique to define other key features, then switch back to vertex mode, and make adjustments to their position to put them at the right depth in the Perspective view. Try starting with the browline first.

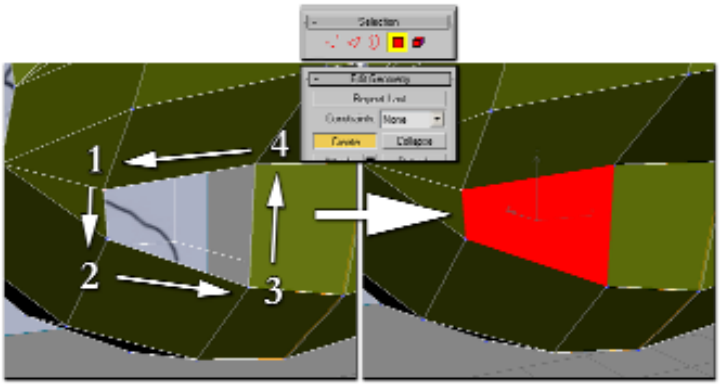
10. Ignore little details like his ears, eyes, and teeth, and continue using the Shift/Move edge extrusion, an the Create Polygon tools to fill in the rest of the gaps on the head. Try to make sure that the edges "flow" nicely. Avoid abrupt turns in your edges, and keep the lines clean.



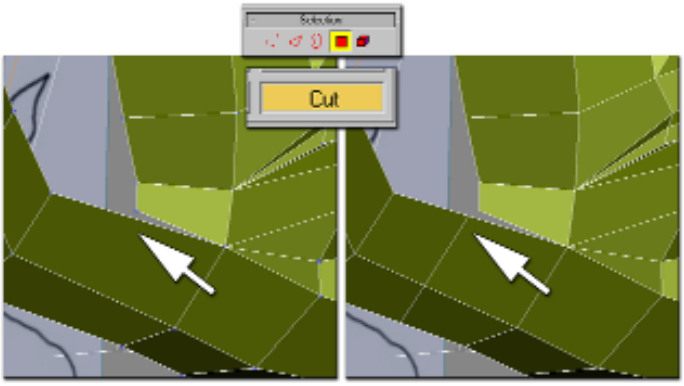
Welding vertices

11. Continue using the Shift/Move edge extrusion technique to define other key features, and make adjustments to their depth at the vertex level. Define more of the nose, cheekbones, mouth, and jawline. Note that you can extrude multiple edges at once. Don't worry about the gaps in between just yet, just define the features that'll give this guy character!

12. Activate Vertex Sub-Object Mode. Zoom in on any areas where several vertices are close together. Right-click, and from the Quad Menu choose Target Weld. Click and release on the vertex that you want to get rid of. Move your mouse to the neighboring vertex, and click on it. The first vertex is welded to the second.



Creating Polygons to fill in the gaps



Cutting new geometry to get things hooked up properly.

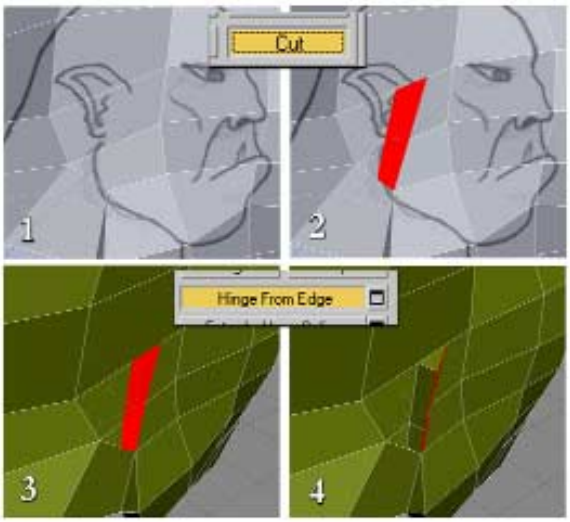
13. Now its time to start filling in the gaps. Use the Shift/Move technique to add any extra geometry, and target weld to close things up. Another option is to go to Polygon Sub-Object mode, and to choose Create from the Modify Panel or the Quad Menu. The vertices will become visible, and you can build polygons on them by clicking your way around the area you want to fill. Create 4-sided polygons whenever possible. Click on the vertex you want to start with, click on the second, the third, the fourth, then click on the first one that you started with. A polygon is created.

14. Ignore little details like his ears, eyes, and teeth, and continue using the Shift/Move edge extrusion, an the Create Polygon tools to fill in the rest of the gaps on the head. Try to make sure that the edges "flow" nicely. Avoid abrupt turns in your edges, and keep the lines clean.

15. Adding geometry to specific areas. You may find that as you continue to Edge Extrude and weld, that you come to some sticky areas where you need a little extra geometry. If you need to create extra geometry, there are two nice options to do so. First, you can go to polygon Sub-Object Mode, choose Cut, click and release on the edge where you want the cut to start, move your mouse, and click again to finish the cut. Right-click to de-activate the Cut function. The second option is to stay in Edge Sub-Object mode, Ctrl/click on the edges you want to have split, then choose Connect in the Modify Panel. The edges are split with new geometry. See the following image for this example.

16. Note that in the last image, I avoided target welding the free vertex to the nearest available point toward the back of the jaw. This would be too much of a stretch, and would break the smooth edge flow. Continue filling in the gaps, adding geometry where necessary, and reducing it when it is possible by Target Welding. Take a look at the progression of mine below.

Progression of Edge Extrusion, welding, cutting, and creating polygons



Defining the ear

17. Now, continue doing clean up work. Make sure that your edge lines flow together. Put in the finishing touches, such as making sure the back of the jaw juts out a little. Add your own personal style, and feel free to deviate from the reference image a little.

18. Zoom in on the side of the head where the ear will need to go. Think about exactly where the ear would attach to the head, as it may be a little hard to see in the reference image. Switch to Polygon Sub-Object mode, and use the cut tool to define this ear/head connection area.

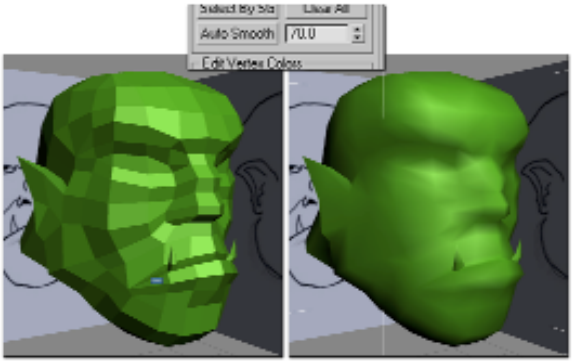
19. Next, select the polygons in the area that you just cut. Hit Alt/X on the keyboard to make the model see-through if desired. In the Modify Panel, click on Hinge from Edge. Move your mouse over the edge that is toward the models face, and click and drag downward. The extrusion swivels out, based on the edge that you hinge from.

20. Finish off the ear by moving vertices and/or using extrusion in conjunction with the Cut tool to get things defined correctly.

21. Exit Sub-Object mode, and create a small cylinder that will be a tooth. Ensure that you are not using too much segmentation. Move it into position, so that it is roughly aligned with the reference image. Select the Head, go to the Editable Poly level click on the Attach button, and click on the cone. Move any vertices around that you need to, and make sure that the vertices at the tip of the cone are welded together.

22. Now the model is basically done, but you may want to add some asymmetrical elements to one side of the model. Exit Sub-Object mode, and go back up to the Symmetry in the Modifier Stack. Right-click on the top of the stack, and choose Collapse All. (This is the same as Convert To from the Quad Menu.) Go ahead and make changes to the model as desired, taking into account that Symmetry will not be mirroring the alterations.

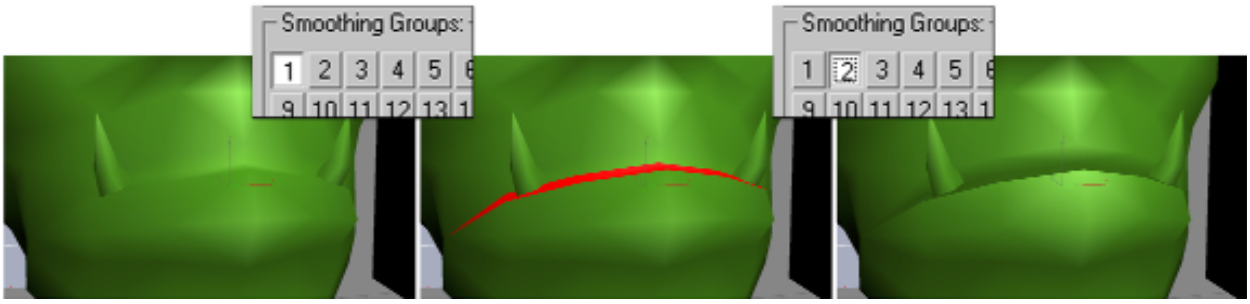
23. Check your work. Now, ideally you've got a clean mesh now, consisting of 3 elements: The main head, and two teeth which are non-contiguous elements. You should have a hole at the bottom of the head, which will be sealed up if we were to connect the neck to the body of the character. There should be no other holes in the geometry. To check that this is the case, activate Border Sub-Object mode, and click and drag around your entire model. Switch to wireframe view by hitting F3 on the keyboard. The edges around any holes should be highlighted in red. If you see any holes other than that at the neck area, zoom in and see what the problem is, and weld any vertices that you need to, so that the area is sealed up. If an area looks very messy, don't be afraid to delete it, and create new polygons.



Before and after using AutoSmooth

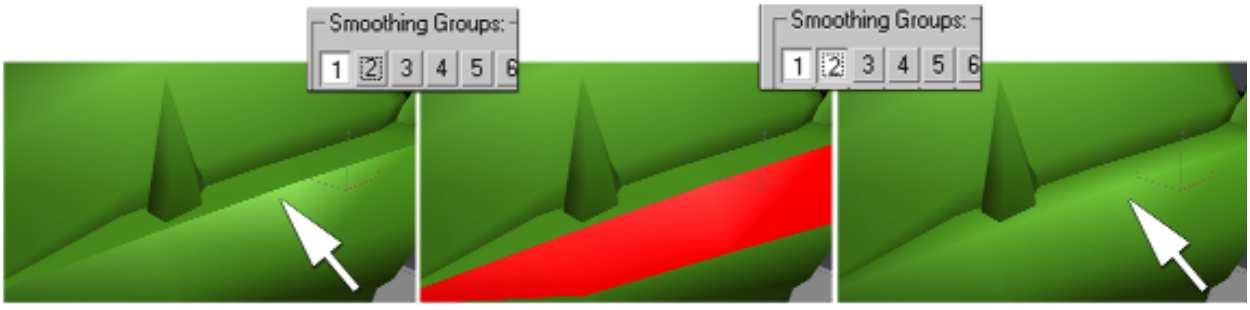
24. The model is done, but the light hitting it makes it look very faceted. Activate Polygon Sub-Object mode, and click and drag around the entire model so that all the polys are selected. Scroll down in the Modify Panel to the Polygon Properties rollout. In the AutoSmooth text field, type in a value of about 70, then click the AutoSmooth button just to the left. Smoothing groups are assigned, based off of the value that you entered. This value is actually an angle in degrees that you entered. Any polygons that are rotated less than 70 degrees to their neighbor will share the same Smoothing Group. Smoothing Groups are sets of polygons which light hits smoothly. They appear to blend together, even though the geometry itself is not altered.

25. Examining Smoothing Groups. AutoSmooth got you in the right direction, but some definition was lost where you wanted it. So now you've got to manually re-assign some Smoothing Groups. Lets work on the crevice where the lips meet first. Zoom in on the lips, and click on one of the polygons on the lips. Under Polygon Properties, see which Smoothing Group button is depressed. Notice that the polygons for the upper lip and the lower lip are all assigned to the same Smoothing Group, causing them to blend together.

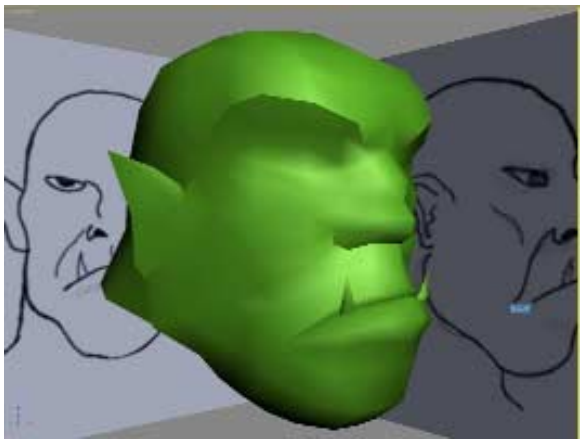


Re-defining the hard edge between the lips.

26. Select all of the polygons at the top of the lower lip. Under Polygon Properties, click on whichever group they are currently assigned to, so that they are un-assigned from that group, and the button is no longer depressed. Now, click on a new group Number, so that these polygons blend together. The number you pick is somewhat arbitrary, but I'm going to choose 2. Now these polygons blend with themselves, but not with the upper lip. See the following image.



Blending the bottom lip using multiple smoothing groups.



All done!

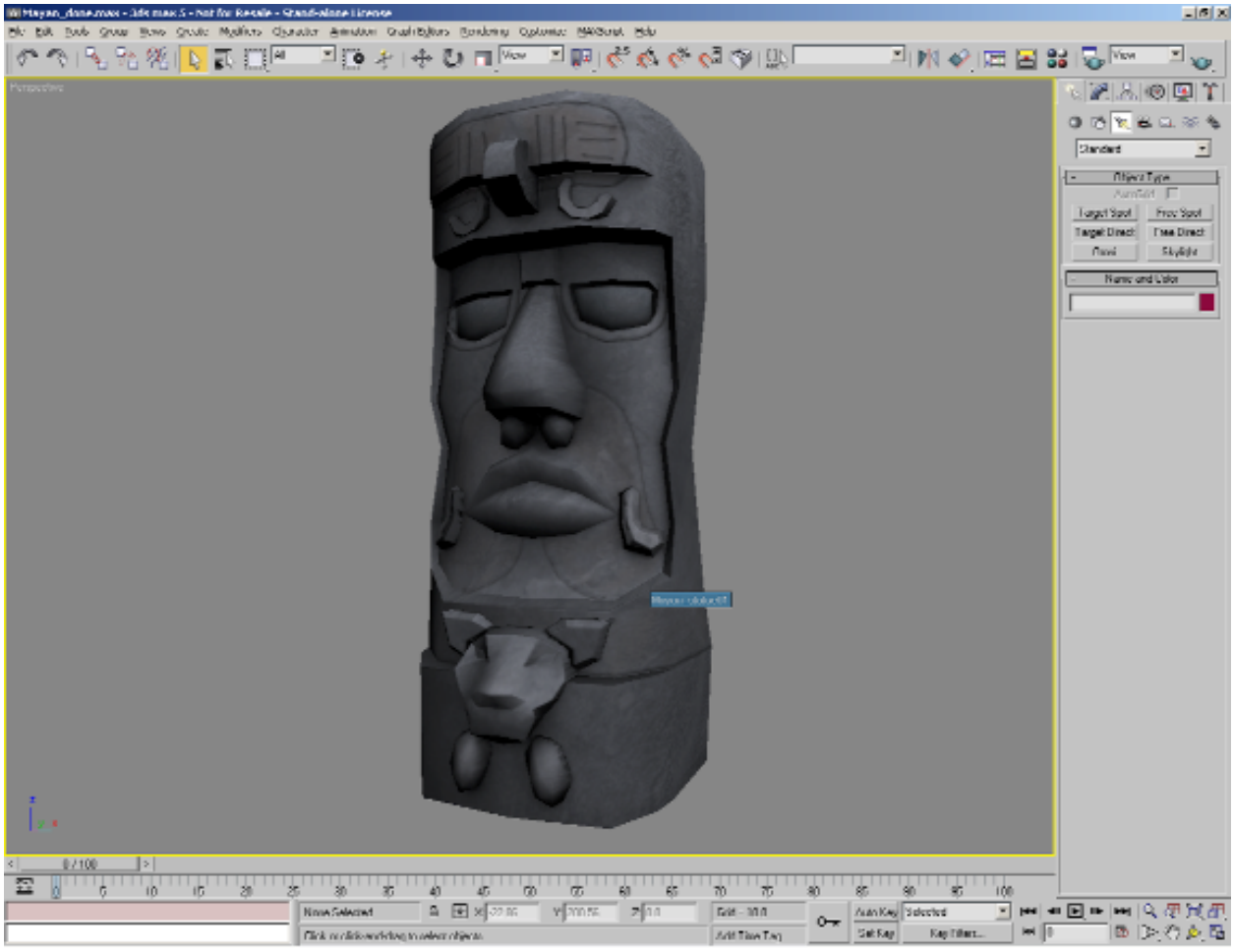
27. Now the seam between the lips is good, but the polys that we just re-assigned are not blending with the rest of the bottom lip. To fix this, select the polygons that are just below the ones we re-assigned. Check their smoothing group, and notice that they are all assigned to Smoothing Group 1, meaning that they don't blend with 2. To fix this, click on the Smoothing Group 2 button, so that they blend with both Groups 1, and Group 2.

28. Any time there is a drastic change in angle, you can get some bad shading if the polygons are assigned to the same smoothing group. Keep this in mind as you re-assign any other Smoothing Groups that you want to. I would recommend adjusting the bottom and sides of the nose, possibly the ears, brow, etc.

29. If you come to any areas where the interior of a polygon is not behaving properly, you may need to turn the interior edge manually. EPoly tries to be smart and manage interior edges automatically for you, but some times you just have to tweak things. To turn an interior edge, go to Polygon Sub-Object mode, and click on the Edit Triangulation button. Just click on the vertex where you want the interior edge to start, and the vertex where it should end.

30. Feel free to add more detail, accessories, etc. Polish things up a little, and you're done! I hope you had fun with this tutorial.

PROJECT 4: UNWRAP UVW AND TEXTURE BAKING TUTORIAL, - MAYAN STATUE



Using Unwrap UVW and Render to Texture

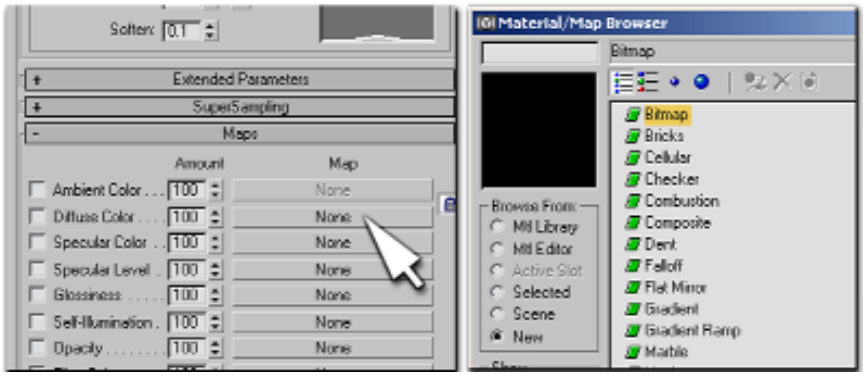
Files Required: file Mayan_start.max, and images Stone.bmp, and Stone_bump.bmp

In this tutorial, you will assign mapping coordinates to an existing model, using Unwrap UVW. You will create and apply a Material to the statue using a Bitmap for the main color. Then you will enable custom scene lighting, and bake this lighting into a new Bitmap using Render to Texture.

1. Open the file Mayan_start.max

2. Bring up the Material Editor from the Main Toolbar, or by hitting M on the keyboard. Click and drag one of the sample spheres onto the model in the viewport. Name the Material in this slow Mayan_head.

3. Click on the Plus button next to the Maps rollout to expand it. Click on the None button to add a texture map. From the Material/Map Browser make sure Browse from: is set to New, then choose Bitmap from the list. Select the Stone.bmp bitmap image supplied.



Add a Bitmap to the Diffuse Attribute

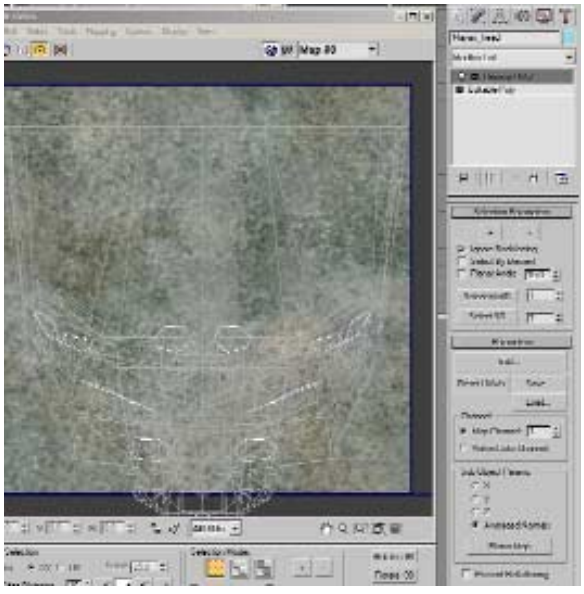
4. Now click on the Show Map in Viewport button in the Material Editor. The texture map is not displayed on the model, because it doesn't have mapping coordinates. Bring up the Material/Map Navigator, and use it to Navigate to the root of the Material. Note that the Stone Map is now noted next to the Diffuse Color.

5. To get the textures to display properly on the model, select the model, go to the Modify Panel, and from the drop-down list, apply an Unwrap UVW Modifier. The textures are now displayed, but are not aligned properly. Click the plus button next to Unwrap in the Modifier Stack, and activate Select Face Sub-Object Mode. Then click on the Edit button to bring up the Edit UVW window.

Show Map in Viewport, and the Material/Map Navigator.

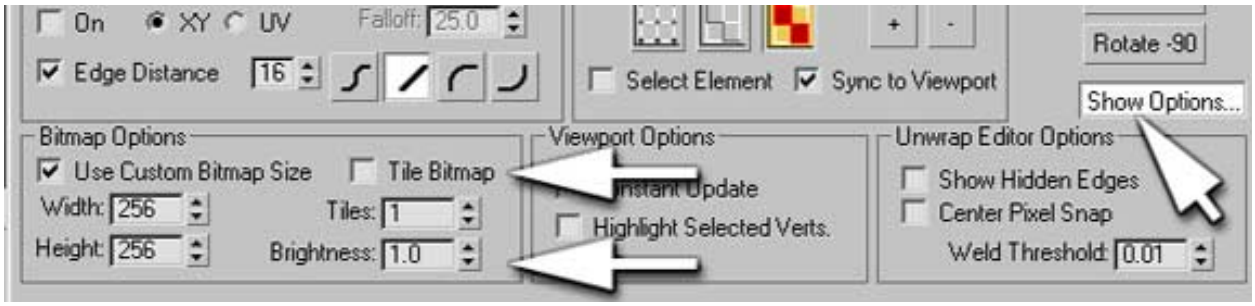
6. At the bottom right corner of the dialog, click on Show Option. Uncheck Tile Bitmap, and Turn the Brightness up to 1. This just makes it easier to see what we are doing.

7. Now, in the Edit Window itself, click and drag around all of the UV points. Notice that the selection syncs to the viewport when you have Select Face Sub-Object mode on. Go to the Mapping within the dialog, and choose Flatten Mapping. Enter the Face Angle Threshold value of 85, and hit enter. This automatically splits up the mesh into various UV elements, breaking them apart based on the angle that you set.



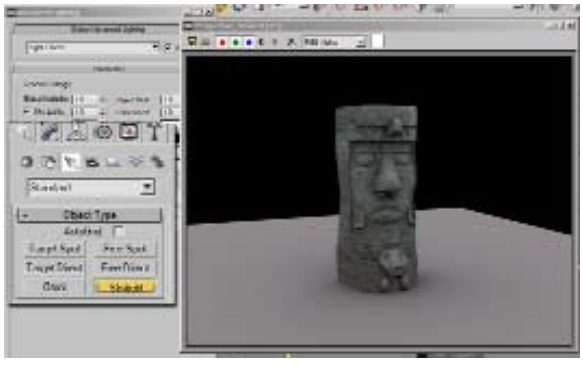
Using Unwrap

8. You can now use the Freeform tool (active on the Edit UVW toolgar by default) to Move, rotate, and scale the chunks if you wanted to. Click and drag to select UV points, and be sure to use Edge and Polygon selection modes within the Edit UVW window if need be. You can move the UVs by moving your mouse within the yellow bounding area and clicking and dragging. Just be sure not to move the pivot point of the UVs, located at the center of the bounding area. If you move your mouse over one of the corner marks, scale is activated, and over the side marks, rotate is enabled. Shift and Control can constrain the transformation of the UVs. The Flatten Mapping did a pretty good job however, so you may not need to do too much work.



Setting the Options

Using Flatten Mapping



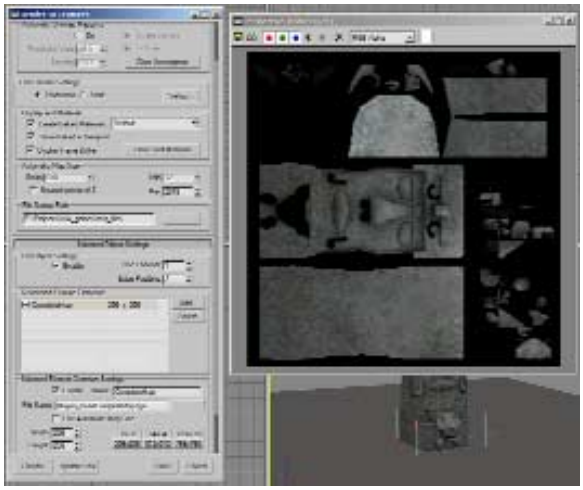
Setting up the Lighting

9. Select a single UV point at the edge of a UV chunk in the Edit window. Notice that other points are highlighted in purple. These are the points that the UV mesh were split apart from. If you wanted to hook up a few of these sections, you could move them into position, then from the Tools Menu, use the welding options to weld points back together. On the flipside, you could also select points, and choose Detach Edge Vertices, which would break them free of the UV mesh. Be sure not to overlap any UV coordinates for this exercise, even though that can be acceptable workflow in many cases. Make any adjustments you desire, and then close the Edit window and deactivate Select Face Sub-Object mode. This UV mesh could be used as a template for painting over. PrintScreen or the Textporter will allow you to take snapshots of this UV image.

10. Activate the Quick Render button from the Main Toolbar to see how it looks so far. Now, create a large box that will be the ground just below the model. Go to the Create Panel, choose lights, and create a Skylight somewhere in your scene. Now, from the Rendering Menu choose Advanced Lighting→Light Tracer. Do another Quick Render of your scene. The Global Illumination makes the model seem much more real, but the rendering takes longer.

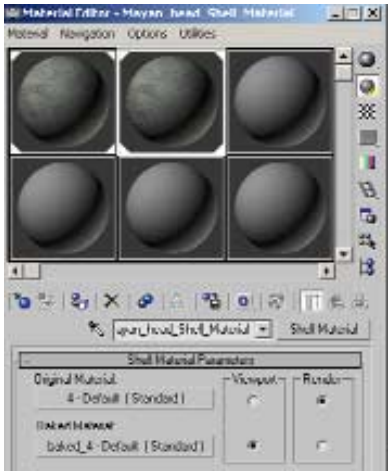
11. We will now bake this scene lighting into a new texture map. Make sure that the Mayan head model is selected, go to the Rendering Menu→Render to texture. Under the General Settings Rollout, make sure that Automatic Unwrapping is NOT on. Click the Settings button for File Output Path, and direct it to the folder that you want it saved to. Under the Selected Object Settings rollout, click the Add button, and choose Complete Map. Set the Size to 256x256. Next to Use Channel, set this to 1. Hit the Render button at the bottom of the dialog.

12. The lighting is now baked into a new texture. Dark areas that you see on the texture are areas that are not used by the UV coordinates, or are where one portion of the mesh obscures another portion, preventing it from receiving light.



Render to Texture

The results of the Tutorial



A Shell Material created by Render to Texture

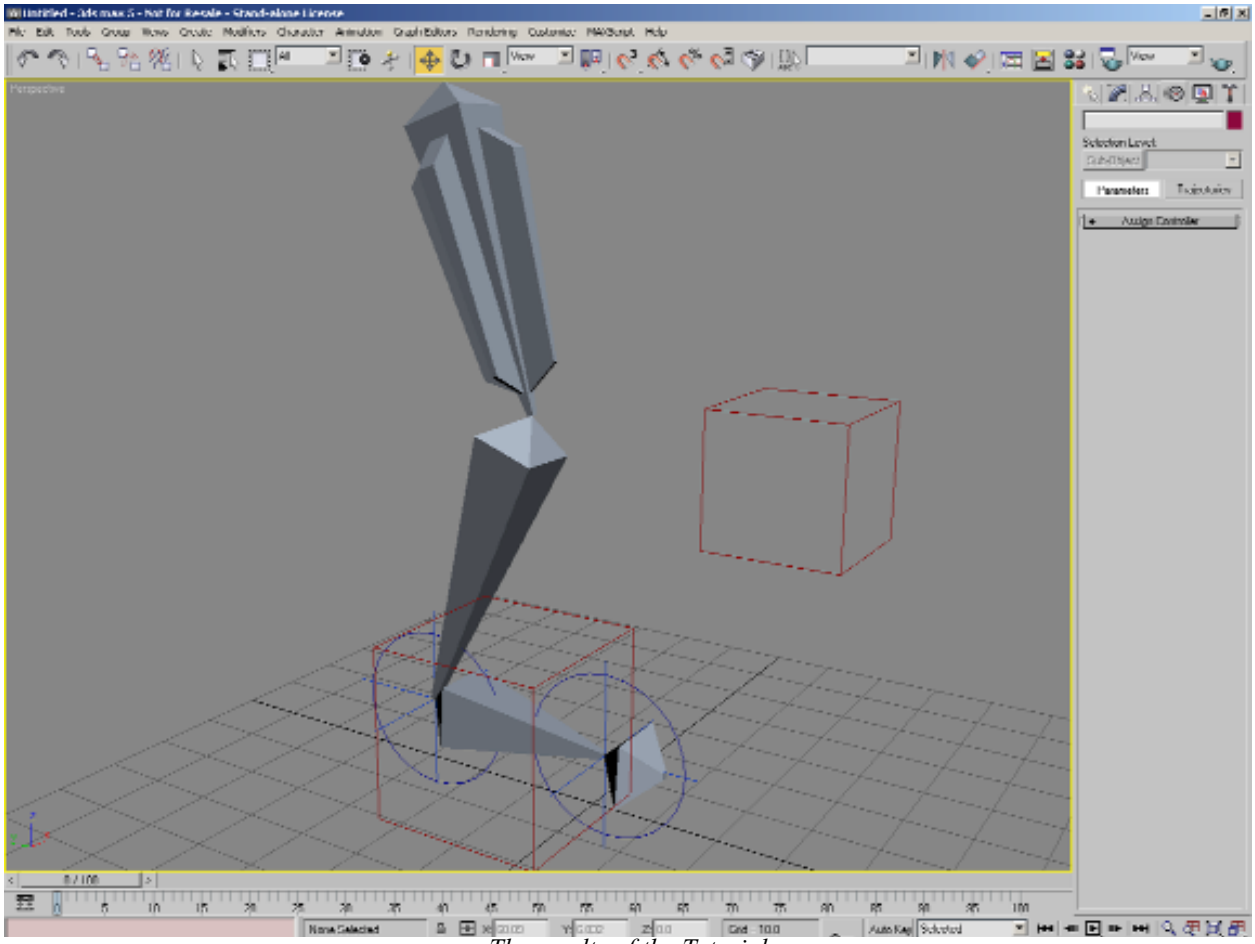
13. Open the Material Editor again, select a new slot, and activate the eyedropper tool just to the side of the Material's default name. Click on the Mayan head object to pick it's Material, which will appear in the active slot. Notice that the baking caused a special type of Material to be applied. Create a new Material that uses the baked texture map that you just made in the Diffuse attribute, and re-assign this to the Mayan head object.

14. Feel free to delete the floor box, the skylight, and to go to the Rendering Menu→Advanced Lighting and choose no plug-in to turn off the global illumination. After all, this lighting is now baked into the texture, so it is not needed.

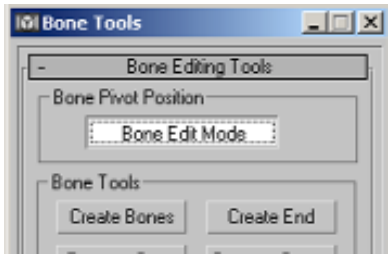
This tutorial taught you how to create and apply materials in Max. In order for the textures to be displayed properly, you utilized Unwrap UVW to assign and adjust mapping coordinates. You then added scene lighting, and baked this lighting information into a new texture map.

Project 5: Animation tutorial — 3ds max Illuminated: Foundation, p. 144

PROJECT 6: HIERARCHIES, BONES AND IK TUTORIAL - LEG RIG



The results of the Tutorial

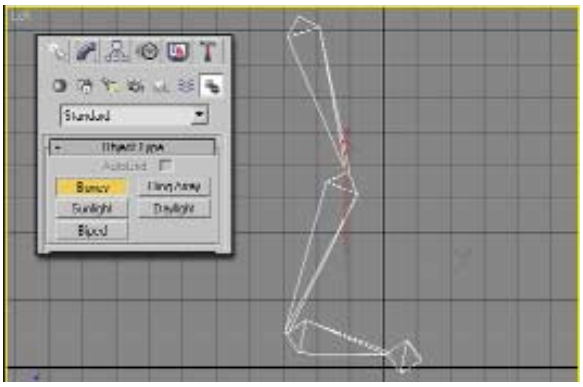


Using Bone Edit Mode. Turn it off when you are done.

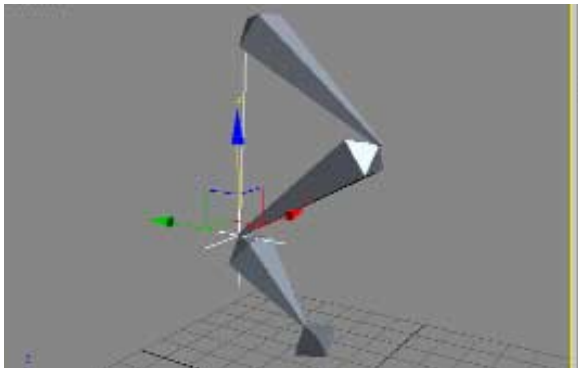
In this tutorial, you will be creating a Bones hierarchy for a leg and foot. You will then use IK solvers to create IK chains. After that, you will create several helper/control objects, which will help you manage the rig.

1. Open Max, and activate the Left viewport. Go to the Create Panel→Systems→Bones. In the left view, click and releare toward the top of the viewport to start creating the thigh bone. Move your mouse down and to the right and click again to finish the thigh. Create a shin bone and a foot bone as well. When you want to finish the bones hierarchy, right-click, and a little bones nub will be left at the end of the foot. Leave this nub.

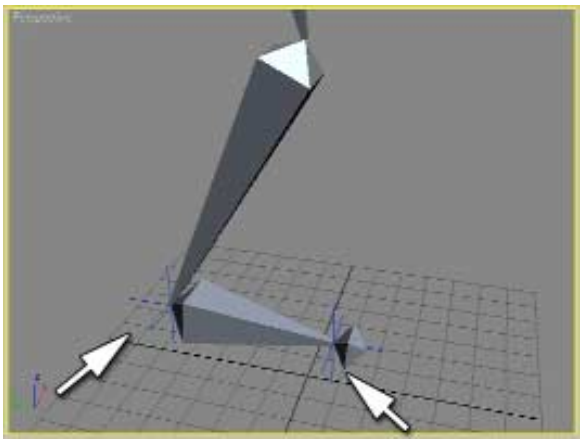
2. Name your bones something that makes sense. Preferably something like Bone_Thigh_Right. You can use Page Up and Page Down on the keyboard to cycle through the hierarchy, but be sure to name them all.



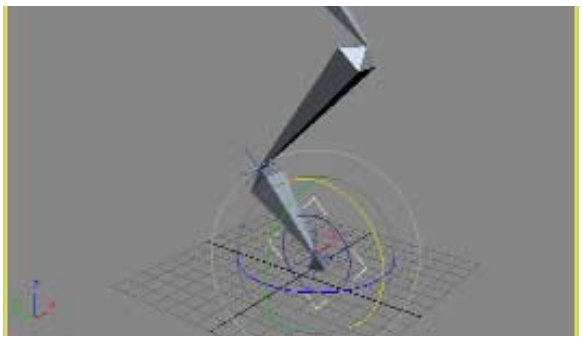
Your Bones should look something like this.



Solving from thigh to foot, creating an IK goal.



Solving from foot to toe nub.



Create a Circle Shape, position it at the Toe, and link the heel goal to it.

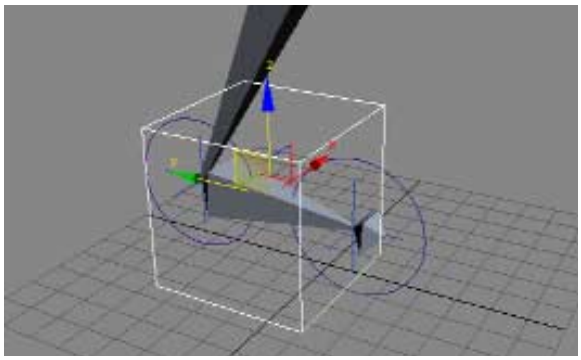
3. Go to the Character Menu and choose Bone Tools. Activate Bone Edit Mode, and Move the shin bone. The bone is stretched and resized. Use Bone Edit Mode to make any adjustments that you feel are necessary to your bones, then deactivate Bone Edit Mode.

4. Next, select the thigh bone, and expand the Fin Adjustment rollout. Turn on fins for the thigh bone. These are a nice visual cue to use, and can help the bones skeleton to fit more snugly inside a character mesh. Close the Bone Tools dialog, making sure that you are not in Bone Edit Mode before doing so.

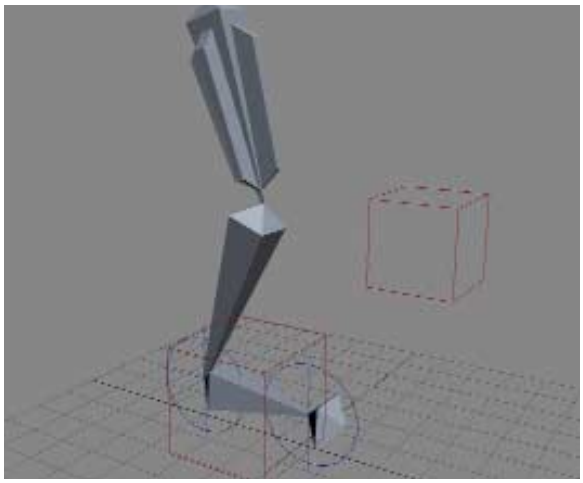
5. Select the thigh bone. Go to the Animation Menu, choose IK Solvers→HI Solver, then move your mouse down and click on the foot bone. If you are doing this in a wireframe view, be sure to move your mouse precisely over the wires. An IK chain is established, with a Goal at the foot. Move this goal up to test the IK solution. Undo this when you are done. Move the thigh bone up and down. The heel tries to stay planted, but the toe does not. Undo this movement.

6. Select the foot bone, Animation Menu→IK Solvers→HI Solver, then click on the toe nub bone at the end of the foot. Move the thigh bone up and down. The toe stays planted. Move the heel goal up and forward. The toe attempts to stay pinned, but is forced to break free at a certain point.

7. Now to aid us in getting the movement of the IK goals correct, we will add some helper objects. Go to the Create Panel, and click on the Shapes button. Create a Circle Spline shape in the left view that is about half as large as the foot. Move this into position around the toe, checking all viewports. Select the IK goal at the heel of the foot, and activate Select and Link on the Main Toolbar. Click on the IK heel goal, and drag to the new circle that you just created, and the circle should flash white to indicate a successful link. Rotate the circle, and see that this makes the heel IK goal lift up off the ground. Name the circle Control_HeelUp_Right.



The foot with Control objects

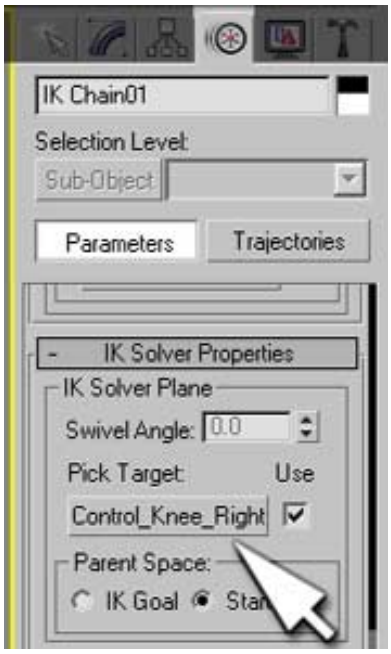


The results of the Tutorial

8. Create another circle around the heel, and link the toe nub IK goal to this circle. This will allow for toe tap. Name this circle Control_ToeUp_Right. Now to easily move everything, go to the Create Panel, choose Helpers→Dummy. With the Dummy button active, click and drag around the foot bone, creating a helper dummy that is slightly larger than the foot. Link both of the Control circles to this dummy, and name it Control_FootMove_Right. This will move the entire foot.

9. Now, the only thing lacking is control over where the knee is pointing. Create another dummy object, and position it out in front of the knee of the leg hierarchy. Name it Contol_Knee_Right. Select the IK heel goal, and go to the Motion Panel. Under the IK Solver Properties rollout, look at the Swivel Angle group of options. This controls the rotation of the leg between the IK goal at the heel and the thigh. Click on the None button under Pick Target, then click on the new knee dummy. Move the dummy from side to side, and note that the knee points at it.

10. The only problem that we have is that if we use the foot moving dummy, and push it up to far, the knee will flip around to aim at the knee dummy. So, to avoid this, select the knee dummy, and link it to the one controlling the entire foot. Now, you will only use the Control objects to animate the foot and knee. You could hide the IK goals themselves if you wanted to.



Controlling the Swivel Angle

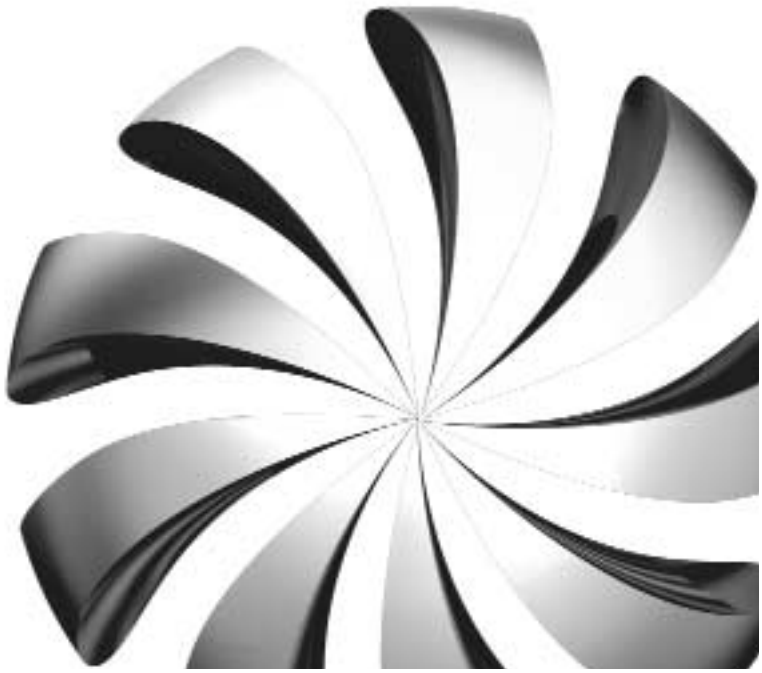
11. To check your hierarchy, choose the Select Tool, click on Select by Name, then check Show Subtree. Objects will be indented below their parents.

12. You could use this type of method for finishing the rest of the skeleton. You could create other bones chains for the left leg, the spine/neck/head, and the shoulders/arms/hands. To connect these branches, you could create a bone or a dummy in the pelvis area, which each branch could be linked to.

In this tutorial, you learned how to create bones hierarchies and edit them. You learned how to apply an IK solution to the bones, creating IK goals. Finally, you learned how to use helpers and control objects to better manage the rig.

Maya for Games

Rick McCann



MESMER

4.1 Course Overview

Create great games with Maya

Course Synopsis:

Maya for Games will present a wide array of techniques and insight for efficiently creating in-game content from previously existing reference material. This course will introduce Alias|Wavefront Maya as a general purpose 3D modeling, animation, and texturing tool. Students will learn complex organic modeling, texturing, and animating techniques for creating game characters.

Starting with the basics of referencing artwork and using organizational techniques such as layers to improve workflow, students will learn how to set up and maintain a clean directory of scene files from which to work. Maya's polygonal modeling tools, texture application, and character rigging features will then be introduced in a progressive series of lectures and tutorials. Modeling will begin with the basics of the polygonal tool set and progress to incorporate advanced techniques involving NURBS as well as deformers to create and modify geometry. Maya's 3d texturing, UV Texturing, material creation, and lighting procedures will be examined following the modeling portion of the class. Elements of the software such as physics and particle systems responsible for driving the cutting edge of gaming technology will be explored. Finally, longstanding practices of animation will be illuminated in stride with the challenging process of rigging a character. Along the way students will learn how to troubleshoot difficulties, yet emphasis will be given on how to work cleanly to minimize or avoid common problems. Students will export their work to a game engine to validate the results.

COURSE PREREQUISITES

Basic interface familiarity with A|W Maya and Adobe Photoshop.

COURSE OUTCOMES

After completion of course content, students will be able to….

- Comfortably build architectural and level elements within Maya
- Assign materials and textures
- Use the UV Editor to assign texture maps to polygon meshes
- Use advanced poly and subdivision tools for organic modeling
- Build IK skeletons and control rigs
- Bind the characters to the skeletons
- Animate game loop animation
- Export the results to the Unreal Engine

COURSE STRUCTURE

This is an Instructor-led 5-day 35-hour course

Each section begins with a lecture to introduce concepts and techniques needed to accomplish the exercises

Exercises follow to provide hands-on experience to engrain the concepts

COURSE RESULTS

After completing the Maya for Games course students will:

- Have a crisp knowledge of Maya's UI, polygon, and NURBS tools
- Model environments and characters with polygon and NURBS tools
- Be able to texture polygonal models and manipulate UVs
- Create models and textures for different Levels of Details
- Be able to create skeletons for game characters
- Animate characters with forward and inverse kinematics using keyframes
- Blend animation elements together
- Import motion capture data and assign it to a skeleton
- Understand data import and export into real-time systems

COURSE CONTACT HOURS

Instructor Led: 40 hours

Lab and Project Time: 20 hours

SOFTWARE REQUIRED FOR COURSE

Alias|Wavefront Maya
Unreal Editor and Unreal Engine (Unreal Tournament 2003)

MEDIA ACCOMPANYING CLASS

Maya for Games Scene Files in Maya Format

REQUIRED BOOKS FOR CLASS

Maya Illuminated: Games, Lane Daughtry, Mesmer Press 2003, ISBN: 0970753012

Patience is a virtue

4.2 Course Outline

Breakdown of tasks (Overview)

(Modeling) Understanding and Developing Characters

- Main characters
- Secondary characters
- Assets
- Complexity
- Level of detail
- Collision geometry
- Reference material

(Modeling) Modeling Concerns

- Geometry budget
- Budgeting Time
- Reusing material
- (Modeling) Modeling Techniques
- Efficiency

(Texturing) Introduction

- Palette
- Color palette
- Texturing
- Texture creation program
- Gamma
- Mood
- Lighting

Texturing Concerns

- vertex normal
- Mapping Polygons

Advanced texturing

- Alpha channels
- Luminance Maps
- Hardware shaders
- Toon shaders
- Volume shaders
- Particle shaders
- anistrophic material
- Raytracing
- Prelighting
- Particle lighting
- specularity
- environmental mapping
- Reflections
- Gamma
- Mood

(Rigging) Rigging and Weighting

- Animation concerns
- Forward kinematics vs Inverse kinematics
- Creating a rig

Idiot-proofing the rig
Trigger
Animation
Weighting

CHARACTER ANIMATION

Animation paths
Animation cycles
Animation
Emission rate
Frame range
Dynamics
Animation Principles
Functions of action
Animation
Animation Considerations
Incorporated actions
Particle designs

(LEVEL) LEVEL CREATION

Creating a pipeline
Level designs
Interface designs
Art team
Animation team
Engineering team
Production speed

(LEVEL) LEVEL DESIGN

Game progression
Goal
Assets
Traps
Spawn points
Save points
Animation

4.3 Course Content

(Lecture) Breakdown of tasks (Overview)

Lecture on this topic should coincide with a demonstration of a currently existing game. As the demonstration progresses, a description of the elements needed to accomplish the tasks at hand. Just outlining the basic progression of a game is the primary topic of importance in this lecture.

DIVISION OF TASKS

DIFFICULTY OF TASKS

NUMBER OF WORKERS PER TASK

AVAILABILITY OF SUPPORT RESOURCES

CONCEPT DEADLINE

FIRST OUTLINE DEADLINE

FINAL OUTLINE DEADLINE

GAME DESIGN WRITE-UP DEADLINE

CONCEPT ART: CHARACTERS DEADLINE

CONCEPT ART: LEVEL DEADLINE

CONCEPT ART: ASSETS DEADLINE

STORYBOARD DEADLINE

INTERFACE DESIGN DEADLINE

FINAL INTERFACE DEADLINE

STAND IN CHARACTER GEOMETRY DEADLINE

STAND IN ASSET GEOMETRY DEADLINE

STAND IN LEVEL GEOMETRY DEADLINE

ROUGH TEST DEADLINE

RIG DEADLINE

CYCLED KEYFRAME MOTION FOR CHARACTERS OR MOTION CAPTURED

CYCLED KEYFRAMED MECHANICAL MOTION OR MOTION CAPTURED

PLACEHOLDER GEOMETRY

COLLISION GEOMETRY

PRIMARY GAME OBJECTS

PRIMARY GAME CHARACTERS

LEVEL(S)

ASSETS

VERTEX LIGHTING

TEXTURE SHEETS FOR PRIMARY GAME OBJECTS

TEXTURE SHEETS FOR PRIMARY CHARACTERS

TEXTURE SHEETS FOR LEVEL ENVIRONMENTS

TEXTURE SHEETS FOR SECONDARY OBJECTS

LIGHTMAPS

REFLECTION MAPS

damage maps

LODs of primary game objects

LODs of primary game characters

damaged/destroyed models

Adjust and edit skin weights

lighting animation

texture animation

sprite animation

particle animation

alpha date

beta date

publishing date

ship date

Characters should be unique and easily identifable.

(MODELING) UNDERSTANDING AND DEVELOPING CHARACTERS

People are generally looking for themselves. Most art is a reflection of who the artist is. In game development, the goal is to have a character that a broad group of people will identify with. This is usually the job of the Lead Artist, however, the character to be created will have certain specifications that need to be evaluated before hand.

(LECTURE) MAIN CHARACTERS

Lecture on this topic should cover how to set up your assets so that they are of the best use when modeling and texturing.

GEOMETRY

How complex the geometry is can help to indicate how it should be approached and which tools to use when modeling.

PALETTE

The direction of creases and hardness of normals can effect lighting on a character, thusly making areas of the character darker. In those areas you can often reduce the geometry of the model

ANIMATIONS

How the character will be moving has a severe impact on how the model should be built. Additional features required for secondary animation may also need to be evaluated.

PERSONALITY

Generally there is something about a character that is unique, and requires special attention or emphasis in character creation.

WARDROBE

Changes in wardrobe could require additional models or specially constructed geometry that can morph into different shapes.

LIGHTING

Texture maps can simulate a lot of geometry quite and should be considered beforehand.

RANGE OF MOTION

The more a character needs to bend, the more a necessary it is to have geometry that can.

EXPRESSIONS

THE DETAIL AND MOVEMENT OF FACIAL FEATURES.

DESTROYED/DAMAGED/DEAD STATES

DESTROYED STATES FOR MODELS CAN BE EASILY CREATED FROM EXISTING

GEOMETRY.

(LECTURE) SECONDARY CHARACTERS

Focus on the differences and similarities of the geometry and texture information

GEOMETRY

PALETTE

ANIMATIONS

PERSONALITY

WARDROBE

LIGHTING

RANGE OF MOTION

EXPRESSIONS

DESTROYED/DAMAGED/DEAD STATES

(LECTURE) ASSETS

GEOMETRY

PALETTE

ANIMATIONS

FUNCTIONALITY

PARTICLE ANIMATION

FIGHTING

RANGE OF MOTION

DESTROYED/DAMAGED/DEAD STATES

(LECTURE) COMPLEXITY

FAR OFF

SMALL

FAST MOVING

FACIAL ANIMATION

HAIR

FUR

REFLECTIVE

TRANSFORMING

(LECTURE) LEVEL OF DETAIL

CLOSE UP

FAR OFF

NORMAL

(LECTURE) COLLISION GEOMETRY

GROUND

ASSETS

NORMAL DIRECTION

SIMPLICITY

FACING ANGLE

(Lecture) Reference material

PHOTOS

DRAWINGS

SCULPTURES

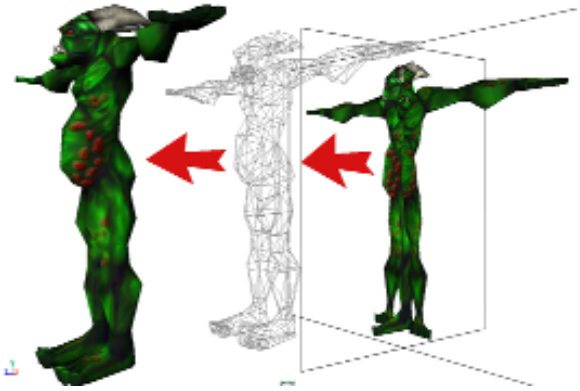
PAINTINGS

PRE-VIS MOVIES

MOVIES

COMICS

BOOKS



An efficient way to use reference is to place it on a semi-transparent plane and assign it to a referenced layer

Goals: Lecture

Getting the students up to speed with what to expect and some tings to consider is the only real goal of this lecture. If students are coming into the class from another in a structured course, they should already be familiar with the majority of the information presented. To liven things up, try demoing some games or showing outlines of production schedules. Alternately, the students may be presented with an opportunity to show their personal work, and explain where they want to go.

(Modeling) Modeling Concerns

This lecture will stress the importance of adhering to modeling restrictions. The goal of the lecture should be to implant the idea that better isn't always exact. To allow the game to function characters and assets have to be low-poly and still look good.

(Lecture) Geometry budget

HOW IMPORTANT IS THE CHARACTER

HOW CLOSE TO THE CAMERA DOES IT GET

HOW EASILY CAN THE TEXTURE BE DIVIDED

HOW WILL THE GEOMETRY RES UP

HOW MANY CHARACTERS IN THE SCENE

HOW MANY POLYS THE ENGINE/PLATFORM CAN USE

(Lecture) Budgeting Time

WHEN IS THE MODEL DUE

WHAT TAKES THE LONGEST TO ACCOMPLISH

RECOGNIZING EFFICIENCY

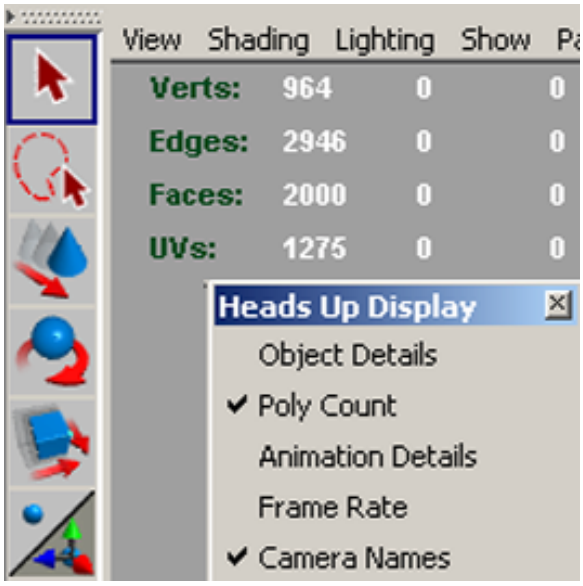
KNOWING YOUR STRENGTHS

GETTING HELP

(Lecture) Reusing material

CANNIBALIZING OLD GEOMETRY

FABRICATION TEMPLATES



An excessive amount of polygons will slow your game play

(MODELING) MODELING TECHNIQUES

CREATING PRIMITIVES

USING THE CREATE POLYGON TOOL

USING THE APPEND TO POLYGON TOOL

SNAPS (GRID, POINT AND CURVE)

(DEMO) REFINING SHAPE

EXTRUDING

Exercise - Create a tail for a monster using extrusions

SPLIT POLYGON TOOL

Exercise - Carve facial details for a character.

cut faces tool

Exercise - Add a division to a characters arm, both in object mode and in face component mode.

DEFORMING WITH LATTICES

Exercise - transform a sphere created along the Z axis into the rough shape of a face using an proportionally divided lattice.

DEFORMING WITH SIMULATIONS

Exercise - Turn a plane into a soft body and add a turbulence field to it. Hit play.

DEFORMING WITH JOINTS

Exercise - Place some joints in an elongated cone with at least 6 divisions in height and bind it. Rotate the joints to make a crooked tree trunk. Duplicate the skin to finalize the transformation.

MAKING SELECTIONS

EXERCISE - SELECT A FACES USING THE PAINT SELECTION TOOL, CONVERTING UV SHELLS TO FACES IN THE UV TEXTURE EDITOR, AND USE SELECT CONTIGUOUS EDGES.

laying out the UVs

Exercise - Copy a character, flatten a portion of the model such as the arm by scaling the vertices flat, do a planar projection on those faces, and use Transfer to copy over the new UV set to the original

COMBINING GEOMETRY

Exercise - combine a character and its accessories into one object

(DEMO) EDIT POLYGONS MENU

This demo should be performed on simple objects, such as a polygonal cube or a sphere, and reinforces with a demonstration of the technique on a character or asset.

FLIP TRIANGLE EDGE

Exercise - Flip an edge in the leg that won't deform properly

SPLIT POLYGON TOOL

Exercise - Take the provided model and modify the shoulder for better deformation

MERGE VERTICES

Exercise - Remove an unneeded edge in the foot

MERGE MULTIPLE EDGES

Exercise - Merge the leg with the body

MERGE EDGE TOOL

Exercise - Merge the other leg with the body

COLLAPSE

Exercise - Collapse the central back face

DELETE VERTEX

USING BACKSPACE, DELETE AN EDGE IN THE RIGHT FOOT, THEN USE DELETE VERTEX TO REMOVE THE UNWANTED VERTEX

DELETE EDGE

Exercise - Delete the unwanted edge in the left foot

TRIANGULATE

Exercise - Triangulate the model. An explanation will be given why it is necessary to have it triangulated before import into a game.

EXTRACT

Exercise - Delete the faces for the left arm, then select and extract the faces in the right arm for separation and editing. An explanation will be given of why this process is easier when the rest of the model is satisfactory but one of the arms or legs needs editing

SEPARATE

Exercise - Separate the newly extracted arm from the body

THE NORMAL EDITING TOOLS

Exercise - Using normal tools on the model. An explanation will be given of why normals sometimes become mixed up, and the importance of normal continuity in a surface, in particular as regards texture mapping.

SMOOTH PROXY TOOL

Exercise - Demonstrate the difference between the linear and exponential subdivision.

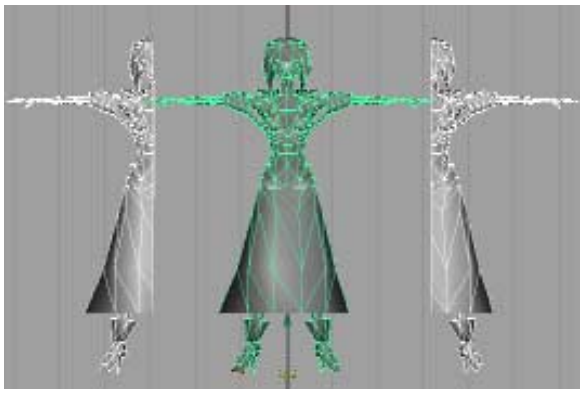
CUT FACES TOOL

Exercise - Create additional divisions in the leg geometry using both the planar cut tool as well as the interactive cut.

WEDGE FACES TOOL

Exercise - Create shoulder pads or armor on the joints of a model using the wedge faces operation.

CLEANUP



Working in halves saves a lot of time and effort

Exercise - There is some junk geometry, i.e. zero area faces, two point faces, in the model that needs to be cleaned up. Explain the effects of bad geometry to a game engine

(DEMO) POLYGONS MENU

CREATE POLYGON

Exercise - Draw a bat wing using the create polygon tool.

APPEND TO POLYGON TOOL

Exercise - In the front view create an initial face for a torso with Create Polygon, then use Append to Polygon Tool to add additional detail, finally fleshing out the side profile

EXTRUDE

Exercise - Create a six sided cylinder then use the Extrude tool, to model a torso and arms from the stomach up.

DUPLICATE FACE

Exercise - Model a quick gun then use Duplicate face to take one of the faces and create a magazine for the gun

MIRROR GEOMETRY

Exercise - Take the torso created to demonstrate Extrude, cut in half, then use Mirror geometry to rebuild it, be sure to demonstrate Mirror Geometry with stitch both on and off

TRANSFER

Exercise - duplicate a model move the vertices of a model in different fashions using a variety of techniques and then transfer the vertex information using the command. Repeat for UVs and Vertex lighting.

TOOL OPTIONS

Exercise - Show the effects of the keep faces together settings using the Extrude tool on a sphere. Demonstrate the results of an extraction using the same settings. Show the error message that comes on when the keep faces planar tool is active

DISPLAY POLY COUNT

Exercise - Show the usage of the tool by creating several polygonal primitives and watching the display feedback's

(DEMO) SMOOTHING SHAPE

This demo will focus on refining the rough shape into a final version for exportation. Discuss different techniques for dividing the geometry and refining the shape.

COLLAPSING EXCESS GEOMETRY

Exercise - Use the collapse function on faces and edges on a model with excessive geometry. Additionally, demonstrate the Merge Vertex Tool in the same areas.

AVOIDING SEAMS

Exercise - Use the Flip Triangle edges, the delete edges and the split polygon tool to reorient the edges of a character that has had its geometry mirrored and combined. Show the effects of offsetting the centerlines in specific locations before mirroring the geometry

SUBDIVIDING FACES

Exercise - Use the Smooth command on one side of a model and contrast it on another with the subdivide faces command.. Demonstrate the results of using Smooth Proxy and explain the functionality.

LINEAR VS EXPONENTIAL

Exercise - Alter the teselation method of a Smooth Proxy Object from exponential to linear for contrast. Repeat for the smooth command

REDUCE COMMAND

Exercise - reduce the number of polygonal faces of a dense model with this command. Highlight the problem areas and explain level of detail LOD.

SCULPT POLYGONS TOOL

Exercise - Smooth out angular components with the smooth function. Define the shape better with the push and pull command.

AVERAGE VERTICES

Exercise - Relax crisp edges and clustered geometry using this command on one vertex instead of many.

(DEMO) CLEANING UP GEOMETRY

To demonstrate warnings and errors, use an exporter and point out the detrimental geometry or warning and error flags that arise.

MERGING VERTICES

Exercise - Duplicate one half of a character or object and scale that duplicate across the appropriate axis. Combine both objects, then select the vertices down the centerline and set the merge vertex options to 0.01 and apply. Click on the CV in the channel box to show the number of vertices before and after the operation.

COLLAPSING EDGES

Exercise - select and collapse an edge on a model in an area to be refined

COLLAPSING FACES

Exercise - select and collapse a face on a model in on a that is extraneous

LAMIA FACES

Exercise - duplicate a face with the separate option checked off, and then select the vertices and merge them. Run the clean up operation to reselect the face.

NON-PLANAR FACES

Exercise - turn on the non-planar face display using the custom polygon display.

EDGES WITH ZERO EDGE LENGTH

Exercise - scale two vertices together in all axis and perform operation

FACES WITH ZERO MAP AREA

Exercise - select an edge and perform a bevel operation on a border edge.

CLEANUP COMMAND

Exercise - this should be demonstrated using the above procedures

FACE NORMALS

Exercise - Turn on the custom polygon display normals and use Normals→Reverse Normals to demonstrate bad normal direction.

VERTEX NORMAL

Exercise - Turn on the Custom Polygon Display Nornals and use Normals→Soften Harden to demonstrate vertex normal direction.

(LECTURE) EFFICIENCY

Efficiency separates the productive from the slackers. It also reduces headaches and problems and most of all, flailing around.

SAVING SCRIPTS

CUSTOMIZING SETTINGS

HOTKEYS

SAVING ACTIONS

PLANNING AHEAD

WORKING IN HALVES

USING INSTANCES

CANNIBALIZING GEOMETRY

GOALS: LECTURE

The goals of this lecture are to enforce the necessity for replicating the art in a stylistically true manner with as little complexity as possible.

GOALS: DEMO

Knowing the location of the tool and what it is used for takes some time. Demonstrating effective workflow using the modeling tools is as important as knowing how to approach a modeling task. When demonstrating modeling techniques, point out common mistakes and solutions to them as well as things to consider later.

(TEXTURING) INTRODUCTION

Textures define the mood and setting for the game. They are also responsible for generating interest in the environment the player becomes immersed in. Texturing geometry can be a complex process. In this portion of the class we will explore the roles of textures in games and examine how they are created, modified and applied.

(LECTURE) PALETTE

MOOD TO BE SET

THE VENUE ON WHICH THE ART WILL BE VIEWED

THE STYLE IN WHICH THE ART IS PRODUCED

COMPLIMENTARY COLORS/CONTRASTING COLORS

Geometry can be reduced in a variety of ways

effect of lighting on colors

effect of rendering on gamma

(Lecture) Color palette

warm colors

cool colors

complimentary colors

the color wheel

rgb vs hsv

cmyk

black

(Lecture) Texturing

Tileable textures

reusing textures

acquiring textures

blending textures

UV mapping

Antialiasing

mip mapping

materials

transparency

(Lecture) Texture creation program

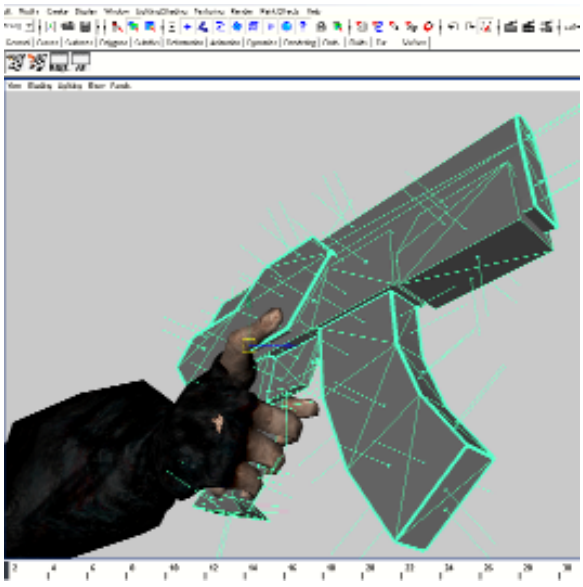
PhotoShop

Gimp

Maya paint

Deeppaint

(Lecture) Gamma



Textures make things look much more realistic

additive value

creating contrast

replacing black

(Lecture) Mood

use of lights

frequency of lights

palette

imagery

objects in the scene

griminess

avoiding confusion

(Lecture) Lighting

vertex lighting

shadow maps

baking lighting

RAYTRACING

PRELIGHTING

EFFECTS OF LIGHT ON TEXTURES

(LECTURE) LIGHTING

ENGINE STYLE

PRE LIGHTING

HARDWARE SHADING

VALUE IN MATERIALS

VERTEX LIGHTING

RAYTRACING

MENTAL RAY

TEXTURING CONCERNS

(LECTURE) MAXIMUM FILE SIZE

(LECTURE) FILE SIZE

A 3 X 8

B 4 X 8

C LAW OF 2

D 512

E 1024

F NONSQUARE PIXELS

(LECTURE) ALPHA

(LECTURE) MAXIMUM SATURATION

(LECTURE) MAXIMUM GAMMA VALUE

(LECTURE) TILEABLE TEXTURES

(LECTURE) TEXTURE PAGE

(LECTURE) TEXTURE SIZE

(LECTURE) VERTEX NORMAL

(DEMO) ASPECT RATIO

Exercise - Assign a non square texture file a material and in the UV texture editor turn on Use Aspect Ratio.

(DEMO) ALPHA CHANNEL

Exercise - Apply a black and white material to the transparency channel of a material. Associate the alpha channel of another image to the transparency channel of another node for contrast. Turn on hardware texturing for transparency and use the full resolution for the textures

(DEMO) CONNECTION TO MATERIAL

Exercise - Examine the shading group node and elaborate on the connection of the material to the UV texture editor.

The Maya for Games book has some excellent texturing tutorials

(Demo) gamma

Exercise - Increase numerical value for the value channel of a material and contrast it against a black background.

(Demo) multiple material assignment

Exercise - Place a material on each face of a cube.

(Demo) normal direction

Exercise - select an edge and perform a bevel operation on a border edge

Mapping Polygons

(Lecture) UV layout

(Lecture) UV sets

(Lecture) texture borders

(Demo) UV Projection Tools

automatic Mapping

Exercise - select an object, open the UV editor, and compare and contrast the results of various projections options in the option box.

planar mapping

Exercise - squash a character or object flat to show the results. Map a plane with a globe texture and use the bend and squash deformers to turn it into a globe.

cylindrical mapping

Exercise - map a plane and use the bend deformers to make the plane a cylinder

spherical mapping

Exercise - map sphere with a linear stripe pattern and contrast the difference to a spherical map.

(Demo) UV mapping

what a UV set is

Exercise - delete a UV set off a model and show the green texture results.

how the texture information gets on it

Exercise - display the results of moving a UV set in the UV texture editor.

selecting Uvs

Exercise - select UVs in the perspective view using the UV mask. Convert other selections of components to UVs using Selection→Convert selection.

moving Uvs

Exercise - select a UV shell and move it using the Move tool.

making UV shells

Exercise - select an edge and use Cut UVs to slice out a shell. Use multiple maps on faces to achieve the same results

SEWING UV SHELLS

Exercise - select a texture border edge and use Sew UVs to put it back together.

ARRANGING UV SHELLS

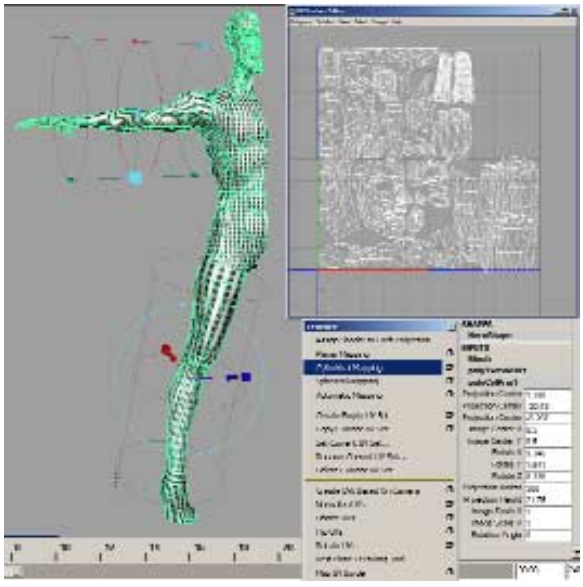
Exercise - use layout UVs to arrange shells in the 1 X 1 texture space.

EXPORTING UV SHELLS

Exercise - use UV snapshot on a selected piece of geometry to save a copy of the UV information out to a file.

OTHER USES FOR UV SHELLS

Exercise - select a shell and convert the selection to faces for painting on with the 3d texturing tool.



There are many ways to layout a UV set onto a flat plane

(DEMO) TEXTURE ASSIGNMENT

HOW TO CREATE A SHADER

Exercise - assign a material to an object using the marking menu and the hypershade.

HOW TO ADD INPUT CONNECTIONS TO THE SHADER

Exercise - map the color channel of an object with a file using the map button. Create the same connections to a material using the hypershade work area.

WHAT THE TYPES OF SHADERS ARE

Exercise - place the gamut of materials on separate objects and demonstrate the differences in the specularity and color value.

HOW TO REUSE PORTIONS OF THE TEXTURE MAP FOR OTHER OBJECTS

Exercise - make and map two materials onto two objects and use different portions of each material on a selection of faces.

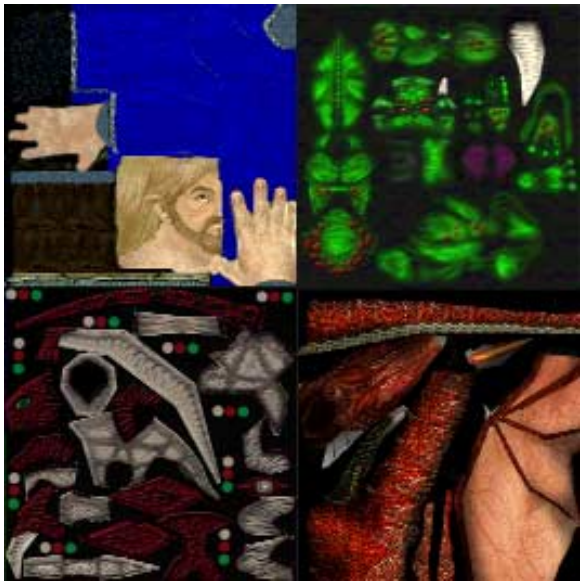
(DEMO) TEXTURE EXPORTATION

HOW TO SAVE TEXTURES PROPERLY

Exercise - show the process of saving a texture file from the 3d paint tool. Demonstrate the use of the render view window for opening and saving images.

COMMON ERRORS

Exercise - show compression that does not load into Maya, reversed alpha channels and so forth.



A good UV layout results in the best use of the available texture space.

Advanced texturing

(Lecture) Alpha channels

what alpha channels are

what type of antialiasing to use

how to connect them to the material

hardware texturing

(Lecture) Luminance Maps

what luminance maps are

what type of antialiasing to use

how to connect them to the material

hardware texturing

(Lecture) Hardware shaders

a what hardware shaders are

b how to create complex hardware shaders

c camera versus facing vs light angle

d hardware texturing

(Lecture) Toon shaders

what toon shaders are

how to make them

where to get them

hardware texturing

(Lecture) Volume shaders

what volume shaders are

how to project textures on them

software rendering for sprite files

(Lecture) Particle shaders

types of particles

assigning materials to the shaders

displaying the effects in real time

sprites and image sequences

Lighting

(Lecture) anistrophic material

what anistrophic shaders are

ow to control their hi-light

how specularity works.

(Demo) Vertex light

defining vertex light

Exercise - Showcase the Edit Polygons→Colors menu

painting vertex light

Exercise - Use the Paint vertex Color tool to paint the vertices of an object with color.

PAINTING VERTEX TRANSPARENCY

Exercise - Increase transparency for the vertex color and paint a selection of vertices.

VERTEX LIGHT VALUES

Exercise - use the component editor to alter the RGB and alpha of the object display

(LECTURE) RAYTRACING

DEFINING RAYTRACING

TURNING ON RAYTRACING IN THE RENDER GLOBALS

TURNING ON RAYTRACING ON THE LIGHT

LIMITING THE REFLECTIONS AND REFRACTION S

ENVIRONMENT MAPPING

(LECTURE) PRELIGHTING

3 POINT LIGHTING

COLOR CORRECTION

OVERLIGHTING

UNDERLIGHTING

TECHNIQUES

IPR RENDERER

(DEMO) BAKING LIGHTS

USING THE HYPERSHADE

Example - Select the material on an object and the object and use convert to file texture to bake the shading group.

TRANSLATING THE FILE

Example - Use the render view to save the texture as something else, and paint to work with the texture file in Maya

CREATING TEXTURE FILES WITH MAYAS RENDERER

Example - Flatten a character for effect, return its scale and then planar map it. Render a frontal view with full lighting and replace the texture file on the character with the newly rendered image.

CREATING TEXTURE FILES WITH MENTAL RAY

Example - Switch the renderer to mental-ray and repeat the above process. Alternately select the material and the object it is on and use Edit→Convert To File Texture.

(DEMO) ANIMATING LIGHTS

KEYFRAMING INTENSITY

Example - Use the RMB and key selected on the intensity channel of a light.

KEYFRAMING COLOR

Example - Change the color value of a material to another color and set the intensity.

BREAKING UP THE LIGHTING

Vertex lighting and pre-lighting can be combined to lower the workload of the game engine.

Example - use the render regions on a light to create a strobe effect. Couple this with barn doors for an additionally cool effect.

(LECTURE) PARTICLE LIGHTING

ADJUSTING THE VALUE

(LECTURE) SPECULARITY

(LECTURE) ENVIRONMENTAL MAPPING

MAPPING THE SPECULAR HILIGHT.

(LECTURE) REFLECTIONS

REFLECTIVE MATERIALS

(LECTURE) GAMMA

ADDITIVE VALUE

CREATING CONTRAST

REPLACING BLACK

(LECTURE) MOOD

USE OF LIGHTS

FREQUENCY OF LIGHTS

PALETTE

IMAGERY

OBJECTS IN THE SCENE

GRIMINESS

AVOIDING CONFUSION

GOALS: LECTURE

Focus the students on minimizing the work they have to do to successfully texture an object, and how to make the best use of their texture space.

GOALS: DEMO

Show effective techniques for increasing efficiency and problem solving. Help the students conceptualize the goal and minimize their frustration at having to do the work more than once. Make sure the students are exposed to common difficulties, and know where to go in order to solve them.

(RIGGING) RIGGING AND WEIGHTING

Game animation must be created in a very short amount of time. This places a lot of importance on being able to effectively transition from pose to pose. To facilitate this action, the joints of a character will be created and then a complex structure will be generated to move them around. Attaching the geometry presents problems if the character deforms improperly. A variety of attachment methods and weighting tools will be used to insure the characters deform properly when weighted.

(RIGGING) RIGGING AND WEIGHTING

Skeletons will be built for the characters created in the previous lessons. The polygonal geometry will be attached to the skeleton using the smooth skin technique and IK will be added to the skeleton to control the character's actions.

ANIMATION CONCERNS

VERTEX ANIMATION OR JOINT

TYPE OF BINDING USED

MAXIMUM NUMBER OF JOINTS THAT INFLUENCE A VERTEX

HOW EDGES WILL COLLAPSE

MOTION CAPTURE OF DISPROPORTIONATE SCALE

(LECTURE) FORWARD KINEMATICS VS INVERSE KINEMATICS

WHAT FORWARD KINEMATICS ARE

WHAT INVERSE KINEMATICS ARE

HOW TO SET KEYS ON FORWARD KINEMATICS RIGS

HOW TO SET KEYS ON INVERSE KINEMATICS RIGS

(LECTURE) CREATING A RIG

PLACING THE JOINTS

CREATING THE CONSTRAINTS

CREATING THE CONTROL STRUCTURE

PARENTING THE CONTROL STRUCTURE

CREATING EXPRESSIONS

IMPLEMENTING SET DRIVEN KEYS

REMOVING EXCESS CHANNELS

SETTING JOINT ROTATION LIMITS

ASSIGNING CHANNELS TO A CHARACTER SET

(DEMO) SKELETON MENU

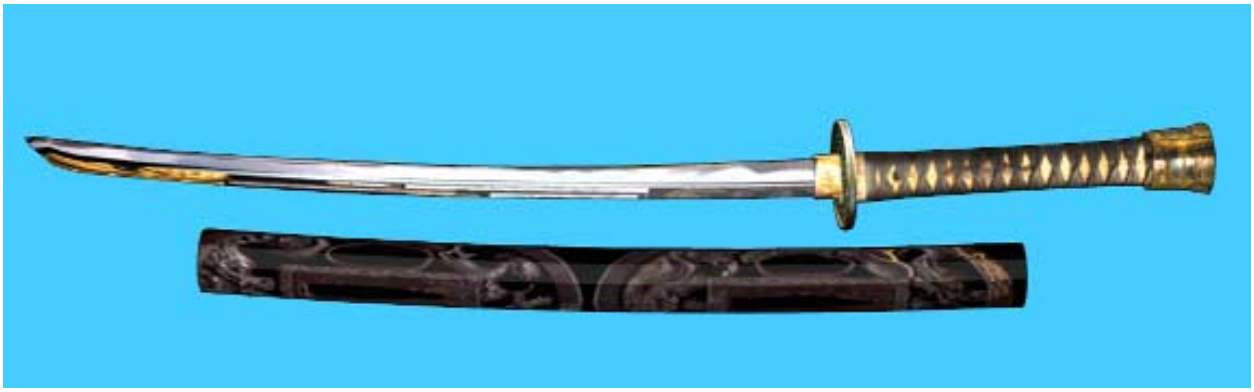
CREATE JOINT TOOL

Example - create a joint structure on an existing character.

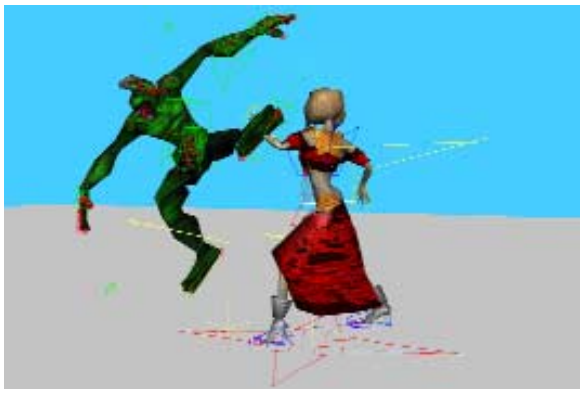
IK HANDLE TOOL

Example - place an IK handle in a straight joint structure and translate it through space. Repeat for a bent leg.

IK SPLINE HANDLE TOOL



Exercises should be fun and enjoyable, and reinforce the lecture amterial



Good rigging and weighting are critical to clean animation

Example - Create a joint chain and place a In handle in it. Use Show→Joints to turn off the joints visibility and move pints of the underlying CV curve. Redisplay the joints and continue to demonstrate the results of moving the CV's around.

SET/ASSUME PREFERRED ANGLE

Example - Create a bent joint structure with at least 4 joints and rotate the last three. Return them to their original position.

INSERT JOINT TOOL

Example - Hi light a joint and use the LMB to place a new joint in the structure.

ADD JOINT TOOL

Example - Display the Hypergraph and add a new joint to the chain.

CONNECT JOINT

Example - Create two separate joint structures. Select the root joint of the second structure and the lowest child of the first structure and watch the second chain translate over to the first.

DISCONNECT JOINT

Example - Select a joint, apply the command and move it off the hierarchy with the move tool.

MIRROR JOINT

Example - create a joint structure on one side of the Axis and replicate it across the to the other side. Use both Mirror Function and mirror behavior from the option box. Demonstrate the difference using the rotate tool along the local rotational axis.

ORIENT JOINT

Example - Use the miscellaneous mask from the pick masks of the component mode in conjunction with a selection of the joints and point out the change in the local rotational axis.

REMOVE JOINT

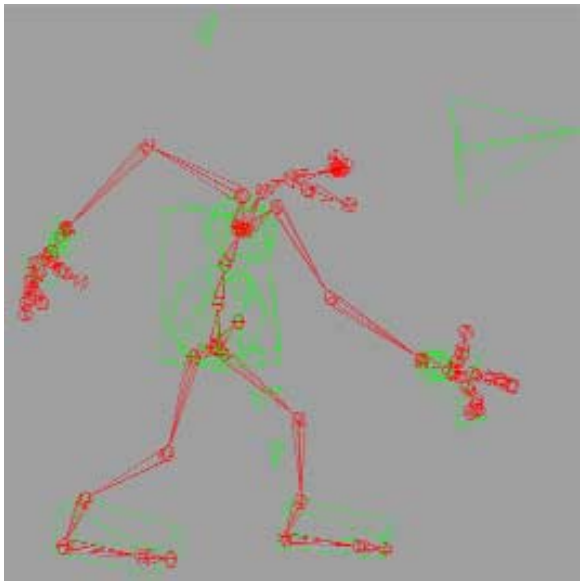
Example - Select and remove a joint form a skeletal hierarchy.

DELETE JOINT

Example - Delete a joint from a skeletal hierarchy. Contrast this with using backspace or delete with a joint selected.

(DEMO) CONSTRAINT MENU

Explain the differences between constraints and parenting. I often liken constraints to magnets. Be sure to stress the importance of selecting the target first and the component second. Also demonstrate how the weight of the constraint can be altered in the channel box or the component editor.



A rig can be used to control bones through IK, constraints, and parenting.

CONSTRAIN TO POINT

Example - Create a locator and a sphere. Constrain the sphere to the locator, and decrease the weight of the constraint. Create another locator and apply another point constraint. Move the weighting between the constraints by lowering the value of one weight or another.

CONSTRAIN TO GEOMETRY

Example - Assign a geometry constraint to a series of planes and pull the first one along. Point snap the pivot point to a corner of the plane and demonstrate the results.

CONSTRAIN ORIENTATION

Example - create a cube and a sphere. Make the sphere the target for the orient constraint and rotate it to demonstrate the results.

CONSTRAIN AIM

Example - make two arrows using the create polygon tool, and then create a target locator. Rotate one arrow 180 degrees and apply an aim constraint to both arrows to demonstrate the results. Be sure to touch on the importance of making sure the vector for the constraint is correctly aligned.

CONSTRAIN NORMAL

Example - create a low resolution sphere and an arrow. Constrain the sphere to the arrow and then rotate the faces of the sphere to demonstrate the constraint.

CONSTRAIN PARENT

Example - create individual joints using the joint tool. Use the constraint to contrast the difference to normal parenting.

CONSTRAIN POLE VECTOR

Example - create an bent joint structure comprised of three joints laid out on a 45 degree angle.

(DEMO) GROUPS AND PARENTING

WHAT A GROUP IS

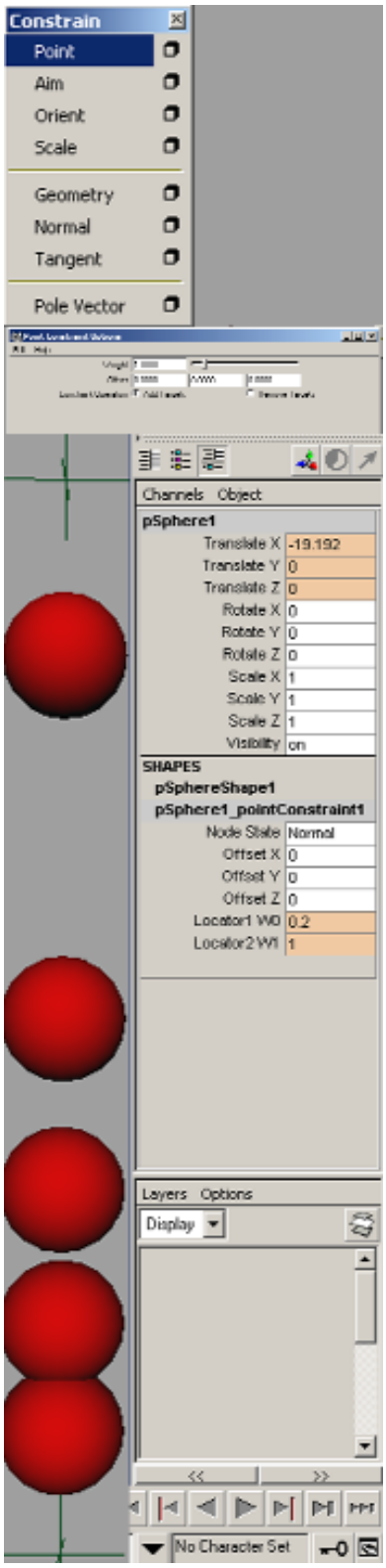
Example - Create a NURBS cube. Select the individual faces and slide them apart.

WHAT A PARENT IS

Example - Create a joint structure with many branches. Move and rotate the children independently of the parent. Make another structure using groups. Show the effect of transforming across a pivot point.

WORLD VS LOCAL

Example - Rotate a ball with world space selected in the tool box, and then switch it to local space.



Constraints control the transformation channels of an object with an value that can be keyframed.

CENTER VS ORGIN

Example - make a group containing two objects with both the center and world options from the Edit→Group Option Box.

ADJUSTING THE PIVOT POINT

Example - Press insert, move the pivot point of an object with the move tool, press inset and rotate the object.

PARENTING IN THE HYPERGRAPH

Example - Select a node in the Hypergraph and hold the LMB, point out the mouse icon change before dragging.

PARENTING WITH THE KEYBOARD

Example - Select two nodes and press p to make the second node a child of the first.

PARENTING WITH THE OUTLINER

EXAMPLE — LMB-DRAG ONE NODE UNDER ANOTHER.

(DEMO) TESTING THE RIG

USING THE RIG TO TRANSLATE

Example - Attach the rig to a motion path and move it through space.

USING THE RIG TO ROTATE

Example - Use the rig to make the character do a flip.

USING THE RIG TO SCALE

Example - Scale the legs of the character so that they are longer, or make the fingers shorter or the character fatter or thinner. Make sure the skin geometry isn't in the same group as the node used to scale.

USING THE SET DRIVEN KEY SLIDERS

Example - Test finger expressions and facial controls, or the foot roll.

TESTING THE EXPRESSIONS

Example - Rotate the back or translate the feet apart to use the expression to center the pelvis.

(LECTURE) IDIOT-PROOFING THE RIG

HOW THE RIG CAN BE BROKEN

INTUITIVE LEAPS THAT NEED TO BE MADE

REMOVING UNUSED CHANNELS

LOCKING DOWN UNUSED CHANNELS

MAKING THE CONTROL OBJETS USER FRIENDLY

(LECTURE) TRIGGER

DISTANCE TO TARGET

SWITCH

DYING

TIME

(LECTURE) ANIMATION

BUTTON ASSIGNMENT

LINKING MOVES

AVAILABLE SEQUENCES

RESPONSE TO ENVIRONMENT

STYLE

(LECTURE) WEIGHTING

BINDING THE SKIN

RIGID OR SMOOTH BINDING

ADJUSTING THE WEIGHTING

(DEMO) WEIGHTING TOOLS

SKIN MENU

Example - Bind a mesh to a skeleton using smooth bind and compare and contrast the differences.

COMPONENT EDITOR

Example - Select a group of vertices and show the Smooth skin tabs and the Rigid skin tabs in the component editor.

PAINT SKIN WEIGHTS TOOL

Example - Adjust the weight on a character using this tool. Be sure to demonstrate all of the tool settings and give examples of workflow.

SET MEMBERSHIP TOOL

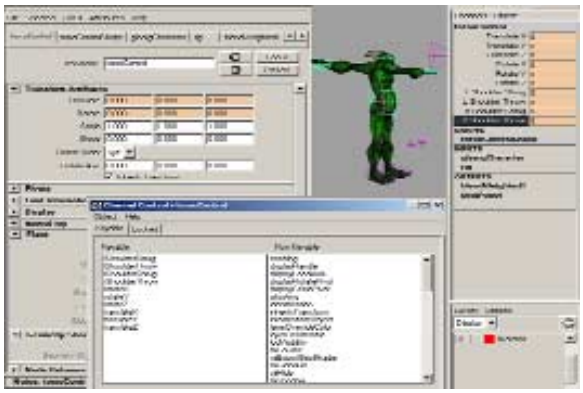
Example - Using a rigidly bound skin, swap the weighting from joint A to point B

GOALS: LECTURE

Some complex relationships work better if they are created in a specific order. It can also ensure that there are no sudden surprises down the line. It is also important to know where the pivot point of any object is, as the action that it drives will take place from that location. Once the control structure is active assigning weight becomes easier. The character can be quickly posed and animated to test the weighting relationship.

GOALS: DEMO

Once a valid control structure is set up, it is easy to animate. It is important to demonstrate how certain objects and workflow are more intuitive than others, and where to go fix problems. The importance of hierarchies relationships should be stressed, as should be the order of operations in regards to applying constraints and parenting.



Locking and removing the extraneous channels on a rig simplifies the workflow.

Character Animation

Animation is the art of communicating ideas and emotions through the motion of a series of pictures. The main goal of an animator is to evoke an emotional reaction from the audience. Regardless of the response if the person can identify with the character, they can envision themselves in its place. This will help make the animation will be successful. Any pure reaction is a valid one, and could be the best gauge for improving your skills. Criticism should be a welcomed aspect to any work and always listened to. They say that the honesty of a child can open your eyes to things you would never see.

Introduction

Animation in Maya can be performed in a nonlinear manner. Poses and different animation sequences may be put together to create sequenced variations. The pieces may be rearranged, split up, scaled, or cycled. In Maya, this work is done in the Trax Editor.

(Lecture) Animation paths

what a motion path is

how to apply an object to a motion path

how to make effective use of motion paths in games

how to offset the animation from the path

the use of nulls

motion path for cameras and fly-throughs.

(Lecture) Animation cycles

defining an animation cycle

Animation clips

Motion capture vs traditional

(Lecture) Animation

Maximum degree of rotation

maximum number of joints in skeleton

maximum length of clips

maximum number of morph targets

maximum number of animation curves on a joint

(Lecture) Emission rate

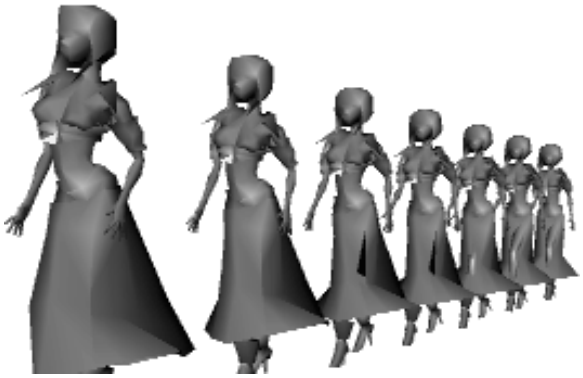
continuous

specified duration

omnidirectional

directional

volume shape



Ghosting and motion trails can help refine your animation.

(LECTURE) FRAME RANGE

SPECIFIED

RANDOM

CYCLING

OCCILATING

(LECTURE) DYNAMICS

RIGID SIMULATIONS

RAG DOLL EFFECTS

COLLISION GEOMETRY

PARTICLE CLOUDS

(LECTURE) ANIMATION PRINCIPLES

TIMING

WEIGHT

SQUASH AND STRETCH

ANTICIPATION

ANTICS

ARCS.

FOLLOW THROUGH

SETTLE

FRAME NUMBERS

TRICKS

OVERACTING

(LECTURE) FUNCTIONS OF ACTION

SHOOTING

STABBING

SLICING

CHOPPING

DRILLING

BASHING

(LECTURE) ANIMATION

TEXTURE ANIMATION

PARTICLE SEQUENCES

MOTION PATHS

ENEMY AI

DOORS AND HATCHES

FOLIAGE

SKY

SUN

MOON

(LECTURE) ANIMATION CONSIDERATIONS

TYPE OF CYCLE

TRIGGER OF CYCLE

ANIMATION TO BLEND TO

COLLISION WITH INTERPOSING OBJECTS

MOTION CAPTURE VS SETTING KEYFRAMES

MAXIMUM DEGREE OF ROTATION

MAXIMUM NUMBER OF JOINTS IN SKELETON

MAXIMUM LENGTH OF CLIPS

MAXIMUM NUMBER OF MORPH TARGETS

MAXIMUM NUMBER OF ANIMATION CURVES ON A JOINT

FUNCTIONALITY OF RIG

SIMPLICITY OF RIG

SHARING ANIMATION

SET DRIVEN KEYS

CHARACTER SETS

TRAX CLIPS

(Lecture) Incorporated actions

Demonstarte some of the following actions

- WALKING
- RUNNING
- SKIPPING
- JUMPING
- FLIPPING
- SHOOTING
- SWINGING AN OBJECT
- PICKING SOMETHING UP
- FALLING DOWN
- SWIMMING
- SWINGING
- THROWING
- TALKING
- BLINKING
- LOOKING AROUND
- BREATHING
- KNEELING
- SITTING
- KICKING
- MAKING A FIST
- CURLING FINGERS
- CRAWLING
- TWISTING
- ROLLING
- FLUTTERING

(LECTURE) PARTICLE DESIGNS

NUMBER OF PARTICLES IN A SCENE

SHAPE OF PARTICLE EMISSION

TEXTURE SEQUENCE ASSOCIATED WITH EMISSION

FRAME RATE

LIFESPAN

INTER-PENETRATION OF GEOMETRY

LIGHTING

ANIMATION SEQUENCE

PARENTING

SPAWN POINTS

TRIGGERS

MAXIMUM POLY COUNT

MAXIMUM HEIGHT AND WIDTH

MAXIMUM DIVISIONS IN UV INFORMATION

MAXIMUM UV SETS

MAXIMUM MORPH TARGETS FOR AN ANIMATION SEQUENCE

MAXIMUM ANGLE BETWEEN POLY FACES

MAXIMUM DIVISIONS OF THE GEOMETRY

(DEMO) TEXTURE ANIMATION

ANIMATING THE UV SHELL

Exercise - animate the position and scale of image ratio on a texture projector to change the position of the image on an object. The

ANIMATING THE TEXTURE PLACEMENT NODE

Exercise - animate the place2d or 3D texture node2 on a material.

(DEMO) SETTING KEYS

AUTOKEYFRAME

Exercise - Set a key using S, move forward in time, and then move the object through space. Show the graph editor and the point out the keyframes are created.

HITTING S

Exercise - Press S to set a keyframe, move forward in time and them move the object in somewhere else in space.

KEY SELECTED

Exercise - select a channel and hold down the RMB. From the drop menu select the key selected. Demonstrate the functionality of this on a color channel, and the visibility channel.

ADD KEYS IN THE GRAPH EDITOR

Exercise - Use the add keys function to add keys to the end of the time line.

(DEMO) CREATING TRAX

CYCLING THE ANIMATION

Sprites are nothing more than a series of images that are looped on a plane that faces the camera.

Exercise - Select an object with animation, and in the graph editor go to Keys→Pre-Infinity/PostInfinity and select cycle.

EXPORTING THE CLIPS

Exercise - Select an animation sequence in the timeline by holding Shift + LMB and dragging or select an object.

CREATING POSES

Exercise - With selection of keys in the timeline use Edit→Create Pose.

CREATING A POSE LIBRARY

Exercise - Copy the poses to the visor or export them to a folder.

EDITING TRAX

Exercise - Use the Split Clip on a continuous track.

IMPORTING CLIPS

Exercise - Import the clips from a file resource to the visor.

GOALS: LECTURE

Animation takes a good deal of practice. The principles of animation are easy to explain, but harder to implement. The use of clips is an excellent way to accelerate and store the animations on a rig. These clips can be blended together to form new animations as well

GOALS: DEMO

Demonstrating good animation practices and shortcuts should come directly before how to adjust the animation that is created in the scene. It is important to demonstrate the results of failing to follow the animation principles, as well as how to catch mistakes as they are happening.

(LEVEL) LEVEL CREATION

INTRODUCTION

Over the period of the day the class will design and create a game level using all available resources and techniques learned during class. The students will conceptualize, storyboard and implement the manufacture of a basic game.

(LECTURE) CREATING A PIPELINE

PRE-VISUALIZING THE NECESSARY COMPONENTS

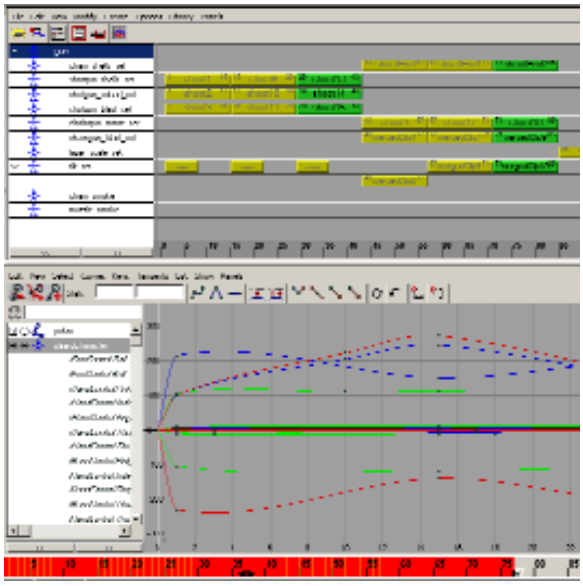
IMPLEMENTING THE AGREED UPON STRUCTURE

FOLLOWING NAMING CONVENTIONS

ESTABLISHING A CENTRALIZED LOCALE

DRY RUN

FEEDBACK AND IMPROVEMENT



The Graph Editor, the Trax Editor, and the Dopesheet are all good for refining animation.

MANAGEMENT

(LECTURE) LEVEL DESIGNS

ENVIRONMENT

PATH OF ACTION

GOAL OF LEVEL

ASSETS IN LEVEL

CREATURES IN LEVEL

PARTICLES IN LEVEL

(LECTURE) INTERFACE DESIGNS

MOVEMENT IN THE ENVIRONMENT

ITEM SCREEN

HEALTH BAR

TIME BAR

COUNTERS

CHAT

CURRENT WEAPON

CURRENT ITEM

POWER BAR

(LECTURE) ART TEAM

MODELING

UV MAPPING

TEXTURING

LIGHTING

(LECTURE) ANIMATION TEAM

RIGGING

CREATING TRACKS AND CLIPS

ROTOSCOPING

(LECTURE) ENGINEERING TEAM

EXPORT ASSETS

INTEGRATE ART

CODING

DEBUGGING

(LECTURE) PRODUCTION SPEED

HOW CAN THE PROCESS BE ANIMATED

WHAT SHORTCUTS CAN BE TAKEN

WHAT S THE BIGGEST WASTE OF TIME

STAYING FRESH

MAINTAINING FOCUS

AVOIDING STRESS

Cooperation is the key to production

(Level) Level Design

(Lecture) Game progression

establishing the characters

setting the period

setting the tone

teaching the player the buttons

teaching the player the moves

rewarding the effort of the player

(Lecture) Goal

be a hero

destroy the enemy

get the money

solve a puzzle

tell a story

(Lecture) Assets

things found in the level

items used to interact with the surroundings

weapons/tools

rewards

damaged opponents or environments

(Lecture) Traps

pit

projectile

acid

lava

lightning

fire

crushing

cleaving

disinformation (maze)

(Lecture) Spawn points

monsters

lighting effects

particles

characters

(Lecture) Save points

start

checkpoints

end of level

end of session

END OF GAME

(DEMO) CAMERAS

FIRST PERSON

Example - show a first person perspective and list games that use this

THIRD PERSON

Example - show the perspective and list games that use it.

ORBITING CAMERA

Example - explain the other directions that a person might be able to move in and the games that use this

FIXED CAMERA

Example - demo some cut scenes that use this

(LECTURE) ANIMATION

TEXTURE ANIMATION

PARTICLE SEQUENCES

MOTION PATHS

ENEMY AI

DOORS AND HATCHES

FOLIAGE

SKY

SUN

MOON

SKYBOX

GOALS: LECTURE

Knowing what is going to happen is paramount to making it happen. Recognizing that an overly complex process will lead to disaster is also good. Communication is essential to incorporating the needs of the institution and team into a deliverable product.

GOALS: DEMO

Showing the standard, and how some of the goals in it were achieved can help to establish realistic goals.

4.4 Quizzes and Projects

Answers

Quiz 1

1. A) True
2. B) False. Vertices define the end of an edge.
3. B) It is found under Polygons→Tool options
4. B) False. Pressing insert allows you to move the pivot point of any selection.
5. B) False.
6. B) False. There is always another way.
7. B) False. The orthographic views can be an invaluable asset.
8. A) True. Polygons can have any number of sides.
9. A) True. The fill hole operation works on a border edge.
10. A) True. The Combine command merges the selected objects into one.

Quiz 2

1. B) False. UV shells can be entire objects.
2. A) True. The UVs can only be adjusted in the UV texture editor.
3. A) True. Using a new texture projector on a selection of faces makes a new shell.
4. B) False. Select a face and use the Rotate UVs command.
5. B) False. Selecting the operation in the history list of the channel box allows the projector to become active again.
6. B) False. Automatic mapping splits up the shell to evenly distribute texture information.
7. A) True you can make a UV set based on camera angle.
8. B) False. You can make an unlimited number of UV sets.
9. A) True. The interior of the shell becomes visible.
10. B) False Using the option box from the Polygons→Transfer menu allows you to copy UV sets from two objects with similar components.

Quiz 3

1. B) False. A local rotational axis can point in any direction.
2. A) True. The target should be placed directly in front of the joint that the IK handle is created from.
3. B) False a node may have numerous input connections.
4. A) True. Joints can mirror behavior or orientation.
5. B) False. Destroying the bind pose allows you to attach new skins to the skeleton in a different pose.
6. B) False. A rig makes animating faster, but an experienced animator can have an FK animation done before you are finished rigging
7. A) True. All elements of a character set get keyed at once.
8. A) True. Expressions can automate tasks.
9. B) False. Set driven keys can have their relationship graphs altered in the graph editor, for one.
10. A) True. Simplicity is always best.

QUIZ 1: MODELING

1) NORMALS EXIST ON BOTH THE FACES AND VERTICES OF POLYGONAL OBJECTS.
   A) True
   B) False

2) A VERTEX CAN EXIST WITHOUT AN EDGE
   A) True
   B) False

3) THE KEEP FACES TOGETHER TOOL IS FOUND IN THE EXTRUDE OPTIONS
   A) True
   B) False

4) YOU CAN'T MOVE THE PIVOT POINT ON A SELECTION.
   A) True
   B) False

5) USING EDIT→DELETE BY TYPE→HISTORY IS USEFUL FOR AVOIDING PROBLEMS.
   A) True
   B) False

6) THERE IS ONLY ONE SOLUTION TO ANY GIVEN PROBLEM
   A) True
   B) False

7) USING THE ORTHOGRAPHIC VIEWS IS A BAD IDEA.
   A) True
   B) False

8) A POLYGON CAN HAVE AN INFINITE NUMBER OF TRIANGLES MAKING IT UP.
   A) True
   B) False

9) THE FILL HOLE TOOL IS USED IN CONJUNCTION WITH EDGES
   A) True
   B) False

10) THE COMBINE COMMAND CREATES A NEW OBJECT.
   A) True
   B) False

QUIZ 2: TEXTURING

1) A UV shell only exists after an edge is cut.
   A) True
   B) False

2) UVs share the same position in 3D space as vertices.
   A) True
   B) False

3) You can create a new shell by using a texture projector.
   A) True
   B) False

4) You cannot rotate UVs incrementally.
   A) True
   B) False

5) Once a projector is deselected, you can never make it active again.
   A) True
   B) False

6) Automatic mapping makes one continuous UV shell
   A) True
   B) False

7) Creating a new projection using the cameras' facing angel is possible.
   A) True
   B) False

8) An object may only have 1 UV set.
   A) True
   B) False

9) Displaying the UV topology allows you to select shells easier.
   A) True
   B) False

10) You can't copy UV information from one object to another.
   A) True
   B) False

QUIZ 3: RIGGING

1) A LOCAL ROTATIONAL AXIS ALWAYS RUNS PERPENDICULAR TO THE Y AXIS.
   A) True
   B) False

2) IK RP SOLVERS REQUIRE A TARGET FOR A P0LE VECTOR CONSTRAINT.
   A) True
   B) False

3) A NODE MAY ONLY HAVE 1 CONSTRAINT ON IT..
   A) True
   B) False

4) JOINTS CAN BE MIRRORED AND WORK THE SAME , BUT HAVE OPPOSITE ROTATION.
   A) True
   B) False

5) DELETING THE BIND POSE ON A SKELETON REMOVES THE WEIGHTING INFORMATION.
   A) True
   B) False

6) BUILDING A RIG IS FASTER THAN ANIMATING A CHARACTER WITH FK
   A) True
   B) False

7) CHARACTER SETS ALLOW YOU TO SET KEYS ON A BROAD GROUP OF CHANNELS.
   A) True
   B) False

8) EXPRESSIONS ARE GREAT TIME SAVERS.
   A) True
   B) False

9) SET DRIVEN KEYS CAN NEVER BE ADJUSTED ONCE SET.
   A) True
   B) False

10) EXTRANEOUS CHANNELS FOR ANIMATING SHOULD BE LOCKED AND REMOVED
   A) True
   B) False

PROJECT 1: MODELING

(Modeling) Suggested Practice Exercises

RECOMMENDED LESSON SOURCE:

Maya Illuminated: Games, by Lane Daughtry, Published by Mesmer Press

These exercises provide students with instructions and build the students competence with the modeling tools, and it is recommended that the students work through the tutorials in the presence of the instructor so that they can ask questions and have them answered.

Model a Gun. Using the Maya for games book as a tutorial reference, model a low polygon gun and save it for use in a texturing exercise later. This exercise should acquaint the students with the main modeling tools.

Model a Sword. This exercise is useful in explaining the concepts of combining and extruding geometry. The students should be placed under a time constraint to accomplish the task of modeling this simple object.

Model a Brood Warrior. This exercise will reinforce all the aspects of creating a clean and polished low-poly model.

GOALS: SUGGESTED PRACTICE EXERCISES

Students should be able to replicate the assigned and produce models similar to the ones presented in the book.

(MODELING) ADDITIONAL EXERCISE SUGGESTIONS

If the students complete the tutorials from the book and wish to progress on their own, assign them modeling tasks that can be incorporated into the final level creation project. It is best to have an idea of what is going to be created, so take a few minutes to get a clear idea of what the students want to make, or establish the project the students will be working on. Each character or asset should be modeled from approved reference material.

Many of the students will be particularly interested in doing their own work or character at this time. This should be discouraged, or supplanted with an achievable facsimile. While the students express the greatest desire to do their own work, they will also become personally attached to the model and may resent constructive criticism. Giving them another project that is similar allows them to channel their constructive energy, but diminishes the risk of allowing the student to invest too much time and energy into achieving something that is beyond their current skill set.

Enforce modeling a character using the DaVinci pose with the elbows and knees slightly bent in the direction they will move in the majority of the time. Make sure that the Display→Heads Up Display→Poly Count is turned on. Make sure that the reference material is on a referenced layer.

Keep in mind the game project at the end. Below is an example of all of the elements that would need to be created to model a classroom.

Desk, computer, monitor, keyboard, plants, chairs

Be sure to point out things that would work better as texture maps, such as electrical sockets, baseboards, window sills, etc.

EXAMPLE MODELING ASSIGNMENTS:

Handgun, rifle, chain-gun
Car, truck, tank
Chair, desk, table
Trashcan, mailbox, newspaper dispenser
Seahorse, Hammerhead Shark
Bull, Elephant
Scorpion, Beetle, Spider
Mecha Bipeds
Superhero/heroine

(MODELING) EVALUATION

The initial project is designed to enhance the students modeling ability. The students should acquire reference material of their choosing either from a previous class, the Internet, or by drawing and scanning it in. A default character Male/Female is available in the scene file, however the appropriate material node will have to be connected to the material. Try creating the model with different techniques and workflows. Some ideas might be to use cylinders to construct all of your geometry, and then append them or merge them together into one object. An alternate wok flow might be to extrude the character out of a basic polygonal shape. You might have the students trace the front outline of the character, and split out all of the geometry using the Split Polygon Tool.

To successfully complete the exercise, students should be able to quickly create basic objects and simple characters. These should be passed off to another student for further revision and critique. A student's model will generally demonstrate their proficiency and speed, and you should always encourage the students to evaluate each others' work.

The following is criteria for evaluating the modeling project. Does the student:

1) Use the orthographic views effectively?
2) Navigate the menus and sub-menus?
3) Understand the difference between component mode and object mode?
4) Know how create and assign objects to layers?
5) Have a grasp on the hot keys and Hotbox?
6) Utilize the Snap-To menu?
7) Correctly create an instanced object?
8) Delete history often?
9) Follow the reference art?
10) Use different components to model with?

RESOURCE FILES:

modelingSetup.Ma

Project 2: Unwrapping UVs

Recommended Lesson source:

Maya Illuminated: Games, by Lane Daughtry, Published by Mesmer Press

UV Map the Gun. This project will demonstrate basic UV texture layout on a preexisting model, with a preexisting texture map, while allowing students to work with their models.

UV Map the Sword. This project will reinforce basic UV texture layout, demonstrate pixel aspect ratio and planar mapping.

Examine the Brood Warrior using the Striped Texture File. This exercise is excellent for pointing out the problem areas of a model and how mutated the original texture projection is.

UV Map the Brood Warrior. This allows the student to approach unwrapping at texture map in a variety of ways, and provides good experience laying out a complex shell into the 1 X 1 texture space.

Goals: Suggested Practice Exercises

The UV shells of the practice objects should be correctly divided and placed over the texture sheet in the UV texture editor. The students should be able to take a UV snapshot and create their own texture if desired.

Additional Exercises Suggestions

Layout and texture previously created characters/assets using tricks and techniques taught in class. This should begin with the more basic assets and progress to allow the students an opportunity to examine and create their own maps for objects. If a resource library of texture files can be made available, it helps. The characters should be mapped before their geometry is mirrored and rigged.

Paint textures onto the geometry using the 3D-texturing tool. Have the students use their laid out character or asset and the 3D-paint tool to make a map for the character.

(Texturing) Evaluation

Texturing 3d objects can be a difficult concept to convey. Some times it take a while before the students understand which maps to use and when. When I'm teaching I always try to explain things in the simplest manner I can, and use a visual aid to reinforce this idea. Explaining that textures are unwrapped like a map from a globe is has helped me convey the idea fairly easily. Students seem to have a tough time accustoming themselves to working in the 2D UV texture editor after learning how to navigate in 3D. Cutting and sewing UV shells together is the most important topic in this portion of the class. I like to make them practice selecting shells a lot.

Proficiency will be demonstrated if the student can split up and arrange the UV information of a polygonal object into the 1 X 1 texture space. They should also know how to display the Texture borders and UV topology. Of critical importance is the ability to select move and scale shells in the texture editor. Be sure they can do this if nothing else.

The following is criteria for evaluating the texturing project. Does the student:

1) Know what color UVs are displayed as, opposed to Vertices?
2) Know how to apply a projection to an object?
3) Know how to apply a projection to a face?
4) Know how do display texture specific information?
5) Remember to activate the move or scale tool in the UV texture editor?
6) Demonstrate skill in cutting and sewing UVs?
7) Understand that a new projection will overwrite the last?
8) Know how to reactivate a projector with the channel box?
9) Export the UV map in the correct aspect and format. ?
10) Line up the directional map on the mesh?

RESOURCE FILES:

stripeTEX.jpg
globe.ma
poolCue.ma

Project 3: Rigging a character

Recommended Lesson source:

Maya Illuminated: Games, by Lane Daughtry, Published by Mesmer Press

Basic two bone IK. This exercise will demonstrate basic IK and its use in driving the rotation of joint.

Rigging an arm. This exercise will result in a fully functioning simple arm and introduce the pole vector constraint to the students.

Rigging a leg. The complex nature of control objects will be explored in this tutorial as hierarchy and constraints are introduced.

Scaling a Rig. This tutorial is really about how to transfer your rig from one character to another.

Rigid bind a character. This exercise should demonstrate how to rigid binding effects a character. It should form the basis for a students concept of weighting.

Smooth bind a character. This exercise should explain the differences between rigid binding and smooth binding.

Transfer the weighting across the characters. This exercise is crucial to the students knowledge as the process can be a real timesaver.

Goals: Suggested Practice Exercises

The student should have a simple, yet fully functioning character that is progressively built and bound to a skeleton.

Additional Exercise Suggestions

Create a skeleton and rig the models created previously in the course.

Weight the new characters using the paint skin weights and the Component editor.

(Rigging) Evaluation

Rigging is a critical element in animation, and the students sense it. This can lead to confusion. Confusion usually leads to frustration. Rigging starts to incorporate more complex relationships and hierarchy connections. The differentiation between parenting, grouping and constraining must be explained to in multiple way, and practiced in exercises at least three times before it sets in. I place a lot of emphasis on knowing which direction on object is facing and the order of operation. Make sure to stress the importance of freezing transformations and deleting history on objects. I would rate that as the greatest cause of difficulty.

On the weighting issue, always add weight. Period.

In the exercises, students should pay close attention to the location of the pivot point of the object. Local and Global orientation are going to come into play a lot, and the students should demonstrate a higher level of proficiency with the channel box.

The following is criteria for evaluating the rigging project. Does the student:

1) Make sure to delete the history of control objects
2) Perform an Freeze transformations on the control objects
3) Name things appropriately
4) Create the constraints before applying parenting
5) Use the Hypergraph to create hierarchies
6) Use spaces and correct node names in the expression editor
7) Remember to set the preferred angle on joints
8) Know what a local rotational Axis is
9) Know how to use the manipulator display to classify which direction an object is facing
10) Understand what each part of the rig is designed to control

Resource Files:

legRig_start.ma
legRig_done.ma
bodyControls_start.ma
bodyControls_done.ma

Project 4: Animating a character

Recommended Lesson source:

Maya Illuminated: Games, by Lane Daughtry, Published by Mesmer Press

Animate a character performing a walk cycle, starting and ending in a passing pose.

Goals: Suggested Practice Exercises

The student should have a decent walk cycle on a character rig that can be looped indefinitely.

Additional Suggested Exercises

Create character sets and poses for previously created models.

Create additional on a sub-character set, create a Trax clip and then blend Trax clips to produce a new animation.

Examples:

Hide
Walk
Run
Arm reach
Jump
Jump up and Reach
Jump and Flips
Head Hits
Shoulder Hits
Stomach Hit
Falls
Flying Deaths
Import a motion capture skeleton
Examine and use Mocap

(Animation) Evaluation

Evaluating animation goes a lot better if you can explain how to make the changes to the problems you see, and why. Outside resources become critical for this portion, and the students should be encouraged to get up and act out any animation they do. Also suggest/impose the use of a stopwatch. Be sure to stress that the start of the animation cycle must be the same as the end of the animation cycle, with the appropriate tangent handle facing in the graph editor. Be sure to stress the usefulness of character sets and demonstrate how to restore keys from the Trax Editor to the timeline. As a tip on animation, I find it's easier to use discrete rotation with whole numbers.

Lethargic animation is a common error, When animating, all of the mistakes in setup become obvious. The students should try to create smooth action if at all possible,

The following is criteria for evaluating the modeling project. Does the student:

1) Activate and deactivate the character sets?
2) Remember to set a key before activating the Auto-keyframe button?
3) Work from the center out to the appendages when creating poses?
4) Know how to create a track and work with it in the Trax editor?
5) Use the Graph Editor and Dope Sheet to refine their animation
6) Hide everything that they don't need to see, or template the layer the objects are on?
7) Act out their animation beforehand?
8) Use playblasts to check their work?
9) Demonstrate the principles of animation in their motion?
10) Make the animation Pop?

PROJECT 5: CREATING A GAME LEVEL

(LEVEL) SUGGESTED EXERCISES

Create a game level with under 15000 polygons. This project should be based on a previously devised set of parameters, agreed to by the students and within the parameters specified by the engine. For example the Unreal engine has a scale of 1 foot = 16UU. UU is unreal units. In metric the conversion is 1 meter = 52.5 UU. A bot is 78 units tall, 34 units wide and so forth.

Planning for this project should take place through out the class and should incorporate elements the students have created over that time. This project will also be an opportunity for them to go back and fine tune some of their previous work if necessary.

Remember, as the static mesh will be converted into a brush, be sure to check it for errors. The remainder of the elements must be exported to the correct format. They must then be imported into the new level and saved as a new package. An example is MyLevel→Package name of the new class→Class name of the texture or resource. Resources must be used or referenced by the actor. Note that an actor can be any piece of geometry to Unreal editor.

EXAMPLE LEVEL SUGGESTIONS

Mountain terrain
Create a glade and stream in a valley
InnerCity
Create a plaza and surrounding buildings
Galactic Models
Planets, stars and a spaceship
Inside of a House
Make a bar, bathroom or an office

OTHER ELEMENTS FOR CONSIDERATION

Create a master file and reference in all other elements and scene files. Distribute them in the scene. As elements are finished, bring them into the scene, Create a camera for each person, and put it on a motion path according to the type of camera desired. (3/4 camera, 1st person, Cut scene camera) Name these cameras and place them on motion paths at the correct height above the characters or scene they are displaying. Associate the students assets to the level using references. Create a Trax clip library, particle effects and triggers using SDK.

GOALS: SUGGESTED PRACTICE EXERCISES

There should be a building or enclosure decorated with objects. If nothing else, a motion path should be created and the students' character and its walk cycle should be imported and applied. (Be sure to set the post-infinity curves to cycle for the keys of the character set or track.) A camera should be created to follow the character as it progresses along its motion path.

(LEVEL) EVALUATION

This project is really all about teamwork and responsibility. The number one problem most artists have is that their ego gets in the way of them being productive. Emphasize teamwork and productivity, set deadlines and have a lot of meetings. This will help keep everyone on track. Use the Reference Editor to create and maintain a scene file to update the tools. Pass one students work off to another student for evaluation and proofing. In actuality I incorporate the work from the whole week into this exercise. At the time of this writing though, the only way to get anything into the Unreal Engine is to use the Personal Learning Edition of Maya that comes with the software.

The following is criteria for evaluating the modeling project. Does the student:

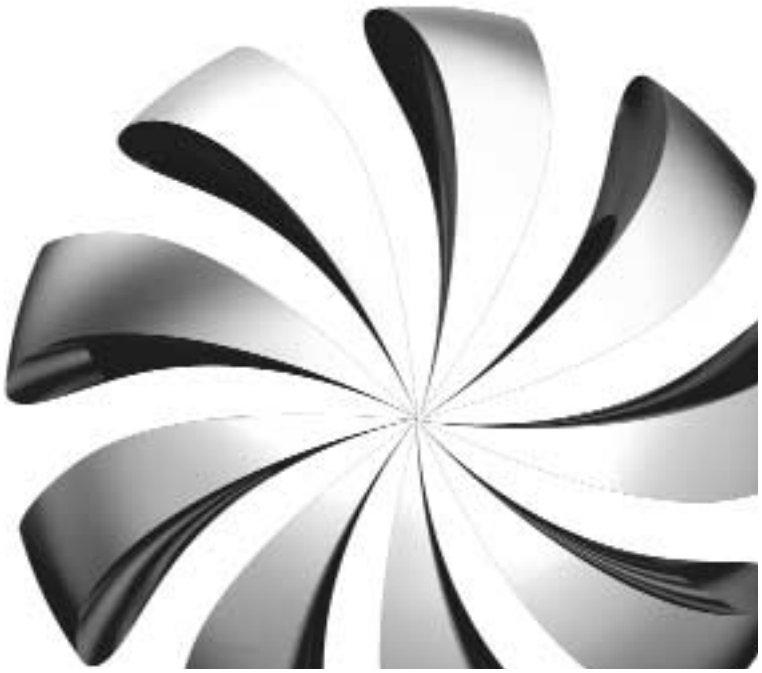
1) Work well with others?
2) Meet their project deadlines?
3) Create and maintain an effective file directory?
4) Bring all the files together?
5) Vertex light the scene?
6) Incorporate animation?
7) Finish the project?
8) Hook up the texture files?
9) Get anything into the Engine?
10) Follow instructions?

RESOURCE FILES:

levelSize.ma

Level Design with Unreal

Lane Daughtry



MESMER

5.1 Course Overview

Course Synopsis

This course leads individuals who are already familiar with stand alone 3D applications through the process of moving 3D content into a Game Engine. Students will learn how to design and build levels, test those levels on a PC, and build in interactivity and network functionality. Students will learn the basic functionality of the Unreal Editor, including modeling, placing imported 3D art into the level, adding texture maps, building visual shader effects, adding interactivity triggers and more. The resulting levels will be tested and played in the Unreal Engine.

A major component of the class will be a group project, where all students will collaborate to create a simple but playable game.

Course Prerequisites

Experience in 3DS Max or Maya is required.

Course Outcomes

After completion of course content, students will be able to:

- Import models and textures into the Unreal Editor
- Use the Unreal Editor to place and modify game content
- Create additional art and level material with the Unreal Editor
- Export levels from the Unreal Editor to the Unreal Engine
- Test play the levels, and examine them for defects

Course Contact Hours

Instructor Led: 40 hours
Lab and Project Time: 20 hours

Software Required for Course

Alias|Wavefront Maya, either Academic or PLE version, The Unreal Editor and Unreal Tournament 2003 game engine must be installed and licensed on each student machine.

Students should also purchase a personal version of Unreal Tournament 2003 and Editor for home use.

Media Accompanying Class

Sample Scenes in Maya format
Sample Unreal Editor Scenes
Game Shell for student projects
The course content
Chameleon Classes – the \Chameleon\ folder goes in the base \UT2K3\ folder.
Chameleon.u – This goes in the \UT2K3\System\ folder.
Chameleon.int – This goes in the \UT2K3\System\ folder.
ChameleonK.ukx – This goes in the \UT2K3\Animations\ folder.
ChameleonS.usx – This goes in the \UT2K3\StaticMeshes\ folder.
ChameleonT.utx – This goes in the \UT2K3\Textures\ folder.
Chameleon Content
Exercise Maps – these go in the maps folder.

REQUIRED BOOKS FOR CLASS

Game Design: The Art and Business of Creating Games, by Bob Bates, Premier Press 2002, ISBN: 0761531653

If the classroom main 3D tool is Maya:
Maya Illuminated: Games, Lane Daughtry, Mesmer Press 2003, ISBN: 0970753012

If the classroom main 3D tool is Max:
3ds max Illuminated: Foundation, Ryan Greene, Mesmer Press, 2002, ISBN: 0970753020

BIBLIOGRAPHY

Level Creation for PC Games, by Andy Clayton, published by Charles River Media, ISBN: 1584502053

TEACHER NOTES

KNOW THE TOOL

A lot of this work is written assuming you as the teacher know the tool. It has been written on top of many texts that are referenced within it that are the actual tutorials. If you are unfamiliar with this tool it will be essential for you to spend a goodly amount of time with it to acclimate yourself and be proficient with it.

KNOW THE UNREAL CONTENT

Another thing that is going to be very handy for you as a teacher is just simply knowing what the UT2K3 content is. There is an absolute wealth of static meshes and textures to use. Enough so that you should never have to bring anything else in. But key to using it is having a general idea of what is present. Spend some time going through it and perhaps make yourself a list of packages you find interesting.

UNDERSTAND THE FOCUS OF YOUR COURSE

Keeping a consistent focus to what you're teaching is one of the most important things you can do with this type of complex material. Decide what the focus of your course is and make sure that how you present this material reinforces that.

It's my suggestion that you have the students spend as much time building content with the editor as possible. Get them through the relevant courseware for the section they are in as quickly as possible and have them start working on content that reinforces what they have learned up until that point.

Along those lines it is a good idea to have seeds of ideas for them to develop into maps if they don't have ideas to work with of their own accord. The only way the complexities of this editor are going to sink in is if they are continually building content in it. It also gives you a chance to take a breather and spend time helping students when necessary.

Also, there is a lot of content to cover here even for a basic introduction to the editor and engine. For example the lighting and emitter tutorials themselves could constitute an entire class. You should cull down what you want to teach to match the length of your class. Trying to cover everything is a time consuming endeavor so make sure you have enough to cover the material properly if you are going to.

5.2 Course Outline

Level Design

- The importance of understanding roles and responsibilities on a large production team
  - Teams need to work together
  - Work done twice is bad
  - Decisions made by wrong person is bad
  - Important to know who is responsible for what
  - Lines get blurry at crunch time
- The role of the level designer
  - Level designer still an ambiguous term
  - Different companies have different definitions
- The importance of authoring with budgets in mind
  - Performance budgets are the end all be all
  - Fudging them is acceptable when done right
- Thinking in milliseconds
  - The importance of ms as a metric
- Monitoring performance in Unreal
  - Unreal stats
- Level Building in UnrealEd
- Creating functionality
  - High level explanation of how game logic works
  - How this corresponds to Unreal
  - Understanding event and time driven systems
- Level design in Unreal
  - BSP construction
  - Zones
  - Static Meshes
  - Actor placement and editing
  - Antiportals
- Dressing an Unreal level
  - Populating with static meshes
  - Static Mesh Browser
  - Static Mesh Editing
- Lighting an Unreal level
  - How lights work in Unreal
  - Ambient lighting
  - Shadows on BSP, how lightmaps work
  - Shadows on static meshes
  - Lighting Only Mode
- Triggers and Events
  - How triggers and events work
  - Other ways objects are triggered
  - Importance of conventions
  - Triggering different types of objects
- Movers
- Authoring for performance
  - Authoring zones and zone portals
  - Visualizing zones
  - Exercise 12 – Zoning a map

- Understanding antiportals
- Authoring antiportals

Creating complex systems
- Every game is going to have its own unique complexities
- Describe ones in Unreal
- Understanding how to author these is important and often a role in and of itself
- Explanation of Emitters
- Scripted Sequences
- Explanation of scripted sequences
- Example of how they provide complex interaction in game

Content Production - Asset creation and integration
- Definition of asset
- Definition of content
- Different ways assets are produced
- Importance of smooth pipeline
- How a smooth pipeline helps content producers
- Example of time wasted by bad pipeline and tools

Examining a sample art production pipeline
- Example pipeline structure
- Initial Concept
- Design
- Functional Prototype
- Final Concept
- Production Level 1
- Production Level 2

Following an asset through the previous art pipeline
- Initial Concept
- Design
- Functional Prototype
- Final Concept
- Production Level 1
- Production Level 2

The role of a production artist
- Design
- Concept
- Production Level 1
- Production Level 2

The content tree
- Explanation of what a content tree is

The Importance of Convention
- Naming Conventions
- Storage Conventions
- Publish conventions to team
- File revisions
- Further reinforcement of importance

Importing textures

UnrealEd materials
- Material Demo Map

FINAL PROJECT

5.3 Course Content

Abstracts of Lectures

This course will abstractly focus on game production issues and then use Unreal for tangible examples. The class will first present performance issues and then discuss how they are addressed in production, then talk about how this is applied within the UT game engine.

The final portion of the class is a lab project in which the class designs a weapon with both a primary and alternate fire. This will entail defining the look of both attacks, the rate of fire, maximum ammo for both, firing effects, in-air effects, and impact effects. These will all be within certain design restrictions and will be accomplished by creating assets that are named as per a predefined naming convention and then imported on top of existing assets. Once this is done, simple editing of the base classes using the Unreal object properties editor will enable the students to define things such as rate of fire, max ammo, sound effects (using built in Unreal sounds) and so on.

Content Management Abstract

Discussion about the importance of naming conventions and storage conventions in a production setting with content files and UnrealEd created files.

Organization means finding what you want in a sea of hundreds of assets and their revisions

Working with many people

Keeping a revision history

Helps with project management and task assignment

UT Specific

Description of conventions that will be used on this project.

Storage – Below are the storage conventions for this project. These folders will be copied and populated on a per-class basis.

Assets
- Package name
  - Asset name
    - Animations
    - Meshes
    - Setups
    - Textures

Naming – Below are the naming conventions for assets in this project

package.group.name

Package – This is the package the asset goes into. The name for these will be determined by the content I create for the class. IE, the students won't be creating any new packages of any importance. This corresponds directly to the Package Name folder that the asset comes from in the asset storage tree.

Group – This is the functional group of the asset. For example in a texture package would have a Wall, Floor, and Ceiling group. A static mesh package might be broken up into Columns, Doors, Struts, etc…

Name – This is the name of the asset itself. It will be verbose and in a Unix case, ie. rocketGunSkin, rocketGunMesh, rocketGunScopeMesh, etc.

INTRODUCTION TO A LEVEL EDITOR

ABSTRACT

Discussion of the usage of a level editor and its definition.

Each editor students will encounter will be different to service the needs of the engine and game it was made for.

At its core a level editor is simply a tool for arranging data and linking functional entities.

Some editors include import functionality, others require external tools to create the assets that they work with. The more wholly integrated the editor is the easier iteration becomes.

UT SPECIFIC

Introduction to core elements of the UnrealEd environment.

INTERFACE ELEMENTS

Navigating the viewports

Toolbars
Browsers
Utility Bar

Lab – Familiarizing yourself with UnrealEd

A simple exercise that has students use elements of the interface to make a simple map and place some premade objects in it, then move them around

EXTERNAL CONTENT INTEGRATION

ABSTRACT

Discussion of what constitutes an asset and asset creation prior to game engine integration.

What is an asset – important to understand this terminology

Why its important to think of assets as small granular pieces

How are assets created

Discussion of the importance of tools that enable quick iteration.

Give examples of both poor production pipelines and good production pipelines and describe how the sole deciding factor is the tools

Discussion about content paths, flat file systems, secondary file systems etc., aimed at acquainting the student briefly with some of the ways that they may have to work in a production pipeline to get content into a game. Talk about the life of an asset and some of the ways that different production pipelines take an asset from inception to final integration. This is simply to demonstrate to the student the many, and often convoluted, ways that people make content.

UT SPECIFIC

Introduction to the notion of packages and how students will work with them

Description of what a package is

How to think about packages

Caveats to working with packages

Different packages for different asset types

Loading package woes

Saving package woes

Discussion and Lab describing the asset types students will be working with in UnrealEd and how to import them.

TEXTURES

Reiteration of how to load, browse, and save textures.

SKELETAL MESHES

Reiteration of how to load, browse, and save skeletal meshes.

STATIC MESHES

Reiteration of how to load, browse, and save static meshes.

LEVEL BUILDING

ABSTRACT

Discussion talking about the notion of a Level Designer and its ambiguous definition given the ambiguous nature of their place in the industry

Focuses on stressing the importance of the students understanding exactly what the role of a level designer is in a given project and how it's so important to know that definition. This is so that they can understand the boundaries of their roles.

This is also to drive home the point that understanding explicitly what the roles and responsibilities of the entire production staff on a project is essential to the project running smoothly.

Discussion talking about the different ways that a level designer might have to think about authoring a level or chunk of gameplay.

Meeting design specs
Authoring with performance in mind
Deciding how to break up a level into manageable chunks

The importance of using established building standards

UT SPECIFIC

Lab focusing on efficient BSP Authoring and editing.

WHAT IS BSP?

How BSP authoring impacts performance and look.

Rebuilding BSP and rebuild settings

TEXTURING BSP

Editing BSP Properties and special BSP face flags

BSP visualization and data reporting aidsLab focusing on Performance Authoring.

Covers tools used in performance authoring

Portals
Antiportals
Zones

Performance visualization aids

Lab focusing on object placement and editing.

Static mesh browser
Static mesh placement

Static mesh property editing on a per instance basis in the levelLab focusing on light placement and editing.

Light placement
Light editing

BUILDING LIGHTINGPRODUCTION CONCERNS

ABSTRACT

Discussion focusing on performance concerns. This will address both performance and resource budgets and the importance of defining and sticking to them. This discussion will also introduce the metric by which to gauge performance.

Focuses on the responsibility of a level designer to author with performance in mind and all of the areas that encompasses.

Straight performance
Resource budgets

The topic of thinking about performance in terms of milliseconds will be addressed thoroughly.

Secondary discussion focusing on the importance of attempting to address performance and resource concerns at the design phase.

A short portion addressing the importance of having someone, preferably the designer who will be responsible for authoring a given level, be conscious of performance issues at the conceptual design phase. The reason being that many times the role of level designer really means level implementer in that they simply build what others have designed.

UT SPECIFIC

Lab explaining the usage of the UT utilities for monitoring both resources and performance.

How to invoke, read, and use the in-game performance monitoring stats and visualizations.

Lab examining and addressing a resource problem.

How to invoke, read, and use the in game resource monitoring stats and visualizations.

Lab examining and addressing a performance problem.

A rather abstract look at a few performance problems and how they might be addressed.

COMPLEX SYSTEM AUTHORING

ABSTRACT

Discussion focusing on the manner in which arbitrary logical entities are wired together within a game environment to create a functional game experience.

Very high level, perhaps with a brief look at some wiring in UT. The bulk of this explanation will be in the actual UT labs. Discussion focusing on complex systems authored within the game editor such as particle systems or physics properties.

Talks about the usage of external tools and special in editor tools to create complex systems and assets. Again very high level with the bulk of this being explained in the actual labs.UT Specific

Discussion about the differences between the notion of an asset and an actor in UT.

> Asset is simply the aesthetic element of an actor.
> Actor is the functional entity in the level that has code and behaviors attached to it. Assets simply define how actors look.

Lab that looks at an existing UT2K3 level to see some examples of how they are wired together.

Discussion and Lab focused on triggers and events.

> Describe the way the Event and Tag system works.
> Creating Triggers.
> Objects that can be triggered

Editing the created trigger actors to trigger events. Discussion and Lab focused on movers using both doors and lifts as examples.

An introduction to Movers
How create Movers
How to edit Movers

Really just an extension of the previous lab focusing on how to use keyed movers with triggers. Discussion and Lab focused on examining UT2K3's particle system and how an emitter is authored.

A very simple look at emitters
How to create one
How to make it create a simple sprite emitter

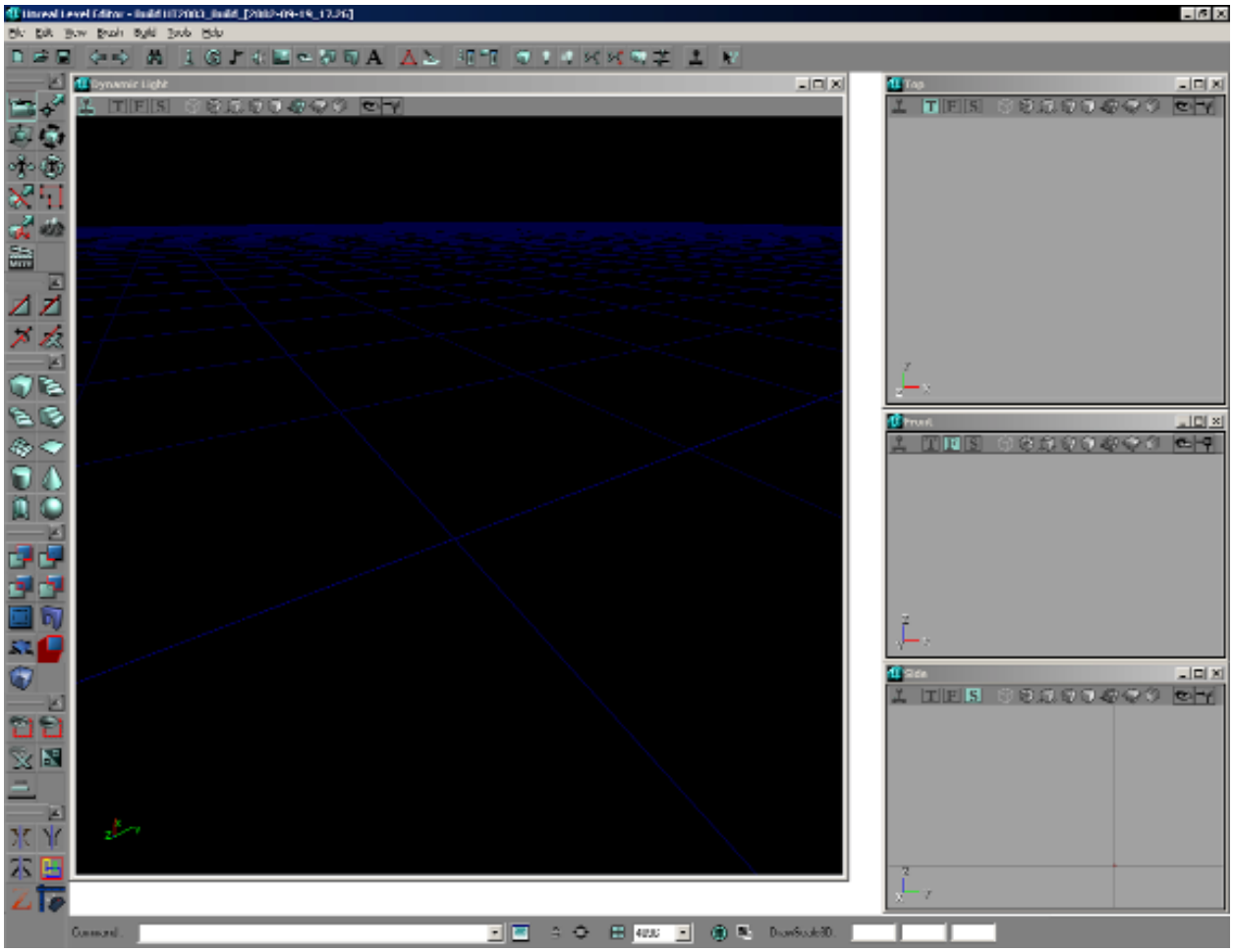
This is one of the most complex systems in UT2K3. An entire class could be devoted to this. As such we're only going to scratch the surface. Discussion and Lab focused on examining UT2K3's material system and how a material is authored.

The difference between a texture and a material.
Examination of the different type of materials that can be created.
Creation of a Shader Material to show how multi-pass shaders are made in UT2K3.

FINAL PROJECT

The final project will be designing and creating a weapon and it effects, building and importing the assets into UnrealEd, defining their functionality using predefined classes, and also designing a simple multiplayer map.

INTRODUCTION TO A GAME EDITOR



The UnrealEd level editing environment

The goal of this topic is to give the students an introduction to the notion of a game editor and talk about how they will use it in production. This section is an abstract discussion. If using UnrealEd services the above goal then do so but try and keep topics abstract still.

DEFINITIONS OF A LEVEL OR GAME EDITOR

The definition of a game or level editor is somewhat ambiguous. That is to say it means something a little different with every separate game engine that you encounter. For that matter many game engines don't have only one editor that is used to assemble all the elements of the game into just that. Many of them have several separate smaller editing packages that are used to make the data that is used by the engine at runtime.

SOME EXAMPLES

For example, for an RTS game there might be one editing environment that allows you to define the elements of a unit. The models it uses, the texture it uses, its properties and behaviors. In yet another one you could assemble similar elements to define buildings. And in yet another you could assemble all of these together on top of terrain to define a final gameplay map.

EACH ONE WILL BE DIFFERENT

The important thing to understand is that the editing path and process for each game is going to be a bit different and that you as a person working on that game will have to acclimate yourself to that particular processes intricacies.

SIMPLY A TOOL

At its core a game editor is simply a tool that allows you to define data that means something functionally to a game engine. For graphical elements you will be defining which assets something uses. For functional elements it will mean defining what properties and behaviors something has. In many cases you will be defining both for an object. It could mean you are defining height fields for a terrain-based map. Or that you are defining primitive which combined define the space in which the player is meant to interact with the game environment. In short, the ways you could possibly use a game editor are as numerous as the amount of games that are the market and in development.

THE UNREALED EDITOR

The goal of this section is to familiarize the student with UnrealEd as a tool. Keep the topics to the tool and the operational paradigms of it. How to build a level will be discussed later.

READING – Unreal editor intro

`http://udn.epicgames.com/pub/Content/UnrealEdInterface/`

`http://udn.epicgames.com/pub/Content/UnrealEdKeys/`

`http://udn.epicgames.com/pub/Content/BrushClipping/`

`http://udn.epicgames.com/pub/Content/VertexEditing/`

`http://udn.epicgames.com/pub/Content/BoxSelection/`

`http://udn.epicgames.com/pub/Content/ShapeEditor/`

The goal of this topic is to familiarize students with the UnrealEd level editing environment. Don't get into specifics of how to make a level work but rather focus on the tool itself.

INTRO TO UNREALED

All of our game editing labs will be done in the UnrealEd proprietary level editor. Level editor is a bit of a misnomer really since UnrealEd is actually a fully integrated game editing solution for the Unreal engine. This means that instead of just making levels it is also used to import all content used by the engine, define the in-game properties of that content, define the functionality of objects, create the game levels and many other game creation based tasks.

EXERCISE 01 — INTRODUCTION TO UNREALED

Be sure to cover these areas as covered in the UDN intro. Following each item will be key elements you should focus on.

- Starting

  This should simply be how to start the editor.

- Maneuvering in editing view ports

  This should be how to pan, zoom, and adjust perspective in the viewports.

  - o Keyboard Modifiers

    Ctrl adds and subtracts from selections.

- View port settings

  Focus on how to modify the layout of the viewports.

  NOTE – Sometimes the viewports will go completely blank. To restore them perform a View→Viewports→Configure… and select a viewport lay out. This will reset them.

- Functionality of different view port display settings

  A guided tour of the various viewport types, ie textured lit, textured, wire frame etc.

- Browsers

  A quick overview of all of the browsers and how to access them. At this point simply give a blurb about what the purpose of each browser is.

  - o Textures

    Used for browsing, selecting, and editing material packages.

  - o Actor Classes

    Used for browsing, selecting, and editing functional actor classes.

  - o Meshes

    Used for browsing, selecting, and editing old style vertex animated meshes. This is legacy technology that is not really in use anymore.

- o Animations

  Used for browsing, selecting, and editing skeletally animated meshes.

- o Static Meshes

  Used for browsing, selecting, and editing static meshes.

- o Prefabs

  Used for browsing, selecting, and editing prefab packages.

- o Groups

  Used for creating groups, deleting groups, and controlling group visibility.

- o Sounds

  Used for browsing, selecting, and editing sounds.

- o Music

  Used for browsing, selecting, and editing music.

• Properties Editor

This should focus on how to bring up and navigate the properties editor. The specific of item properties will be discussed later.

• Toolbars

Focus on the basic usage of each tool. Similar to the browser sec tion above. More detailed usage instructions will come later.

- o Manipulation Tools

  Translation, rotation, and scale tools. Also vertex editing tools.

- o Clipping Tools

  Used for clipping off portions of BSP geometry or splitting them in half.

- o Creation Tools

  Used for creating various primitive shapes that constitute the builder brush.

- o CSG operation and Special Creation Tools

  Performs the CSG operations with the current builder brush.

o Visibility and Manipulation speed

Controls the rate at which the mouse moves the view and objects in the world.

• Bringing up the console in game

To bring up the console in game you will press the tilde (~) key. Or you can optionally press Tab to bring up a one line console that will go away after you enter a command.

Any in game commands that we discuss from now on will be executed from the console.

DEALING WITH TOOL INSTABILITY

The goal of this topic is to drive home the responsibility of the student to save often to preserve their work in an environment that is unstable.

TOOLS CAN BE UNSTABLE

All software tools are unstable. Even ones that have teams of hundreds of people doing QA and-bug checking them. When working with custom and proprietary editors it's usually only a few people if any doing any real testing on them. As such these tools can be very unstable at times. UnrealEd is no exception. The only way you have to combat this problem is to save often. It can't be stressed enough how important this is.

AN INTRODUCTION TO WORKING WITH UNREAL LEVELS

The goal of this section is to look at how the tool as whole is used to put together a level. Don't focus on the details of how the level is being put together, but rather on how the tool is used to facilitate all of the pieces being assembled together.

HOW TO THINK ABOUT WORLD SPACE IN UNREAL

It is important to understand how to think about creating space in Unreal. Many FPS (First Person Shooter) engines define areas and rooms by encapsulating them in a set of sealed walls. For example a simple box room would be created by taking six rectangular primitives and forming the walls of the room thereby sealing that room off from the space outside of it. Unreal is a bit different.

WORLD IS CLAY BLOCK

In Unreal, instead of the space being empty and you conversely having to create boundaries to define areas, you can think of the world initially being one large solid block of clay. To create areas you define primitives which you then use as the template to carve out of the "clay". Once you have carved out this area you can select the individual faces of it and assign textures and properties to them.

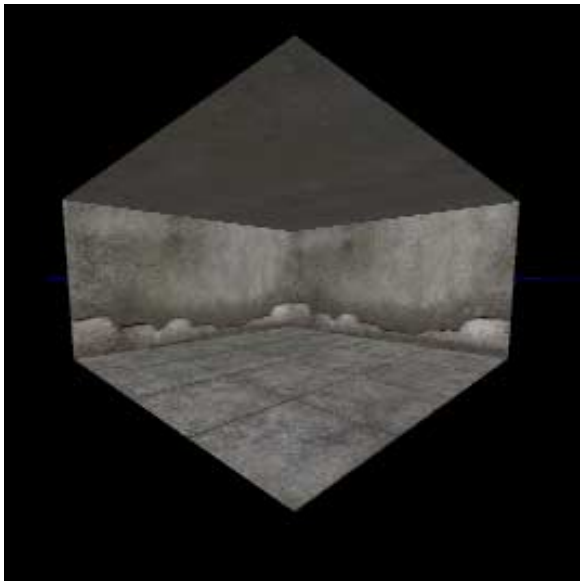
ADDING PRIMITIVE SHAPES

You can also add shapes back to the carved out areas using the same primitive shapes. For example you could carve out a larger rectangular area and then add back smaller shapes to define walls, columns, pedestals, etc.

NOTE — REBUILDING THE LEVEL

Sometimes when rebuilding the level you'll encounter a problem where UnrealEd hangs right after calculating the lighting. What's happening is that it's trying to build paths but finding no path nodes. Path nodes are beyond the scope of this class (a description of them and their usage is here http://udn.epicgames.com/pub/Content/NavigationAI/). To address this problem open the Build Options dialog and uncheck the Define Paths check box and then build.

A simple level



Texture the room as you see fit

EXERCISE 02 — BUILDING A SIMPLE LEVEL

The purpose of this lab is to have the students build a simple level from start to finish simply so they can get a taste of the steps in the whole process. Don't spend too much time dawdling on details here. This is a precursor to more detailed instruction later.

- Create a room
- Texture the room
- Place lights in the room
- Place a weapon spawner in the room

  Place a xPickUpBase→xWeaponBase in the center of the room.

  Open its properties and in the xWeaponBase roll out set the WeaponType variable to whatever type of weapon you want to create.

- Build the lighting in the room

  o Lighting settings

  Alter the light properties until you are happy with the illumination in the room.

- Placing a player start

  Right-click on the floor in the perspective view.

  Select Add Player Start Here.

- Running the map

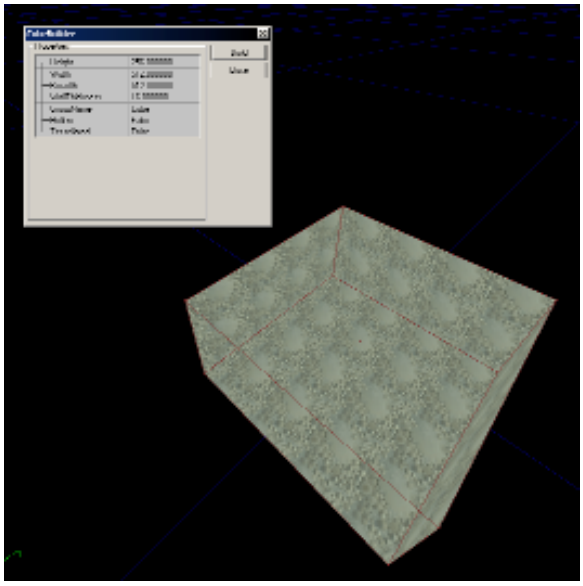
  Save the map as yourName_exercise02.ut2.

  Click the run map button.

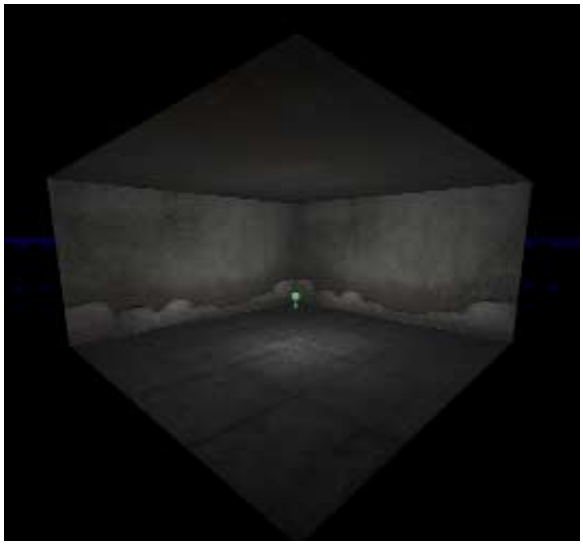
- Add some decoration to the room
- Rebuild it

  Click the rebuild all button.

- Save the map



Create a simple box room with the dimensions 512x256x512, and then subtract it from the world



Place a light in the center of the room

Save the map as yourName_exercise02.ut2

- Run the map

AN INTRODUCTION TO ACTORS

The goal of this topic is to give the students an understanding of what an actor is in the context of Unreal and how they should approach working with them. Also the notion of inheritance is discussed.

HOW TO THINK ABOUT ACTORS

Once you have created an area you can begin to populate it with objects that have functionality attached to them. This is a very important concept to understand with most game editors. When you place an enemy or a light or any other object you are placing an entity that has properties and behaviors attached to it that define how they work in the engine. In the Unreal engine these are referred to as actors.

HOW ACTORS FUNCTION

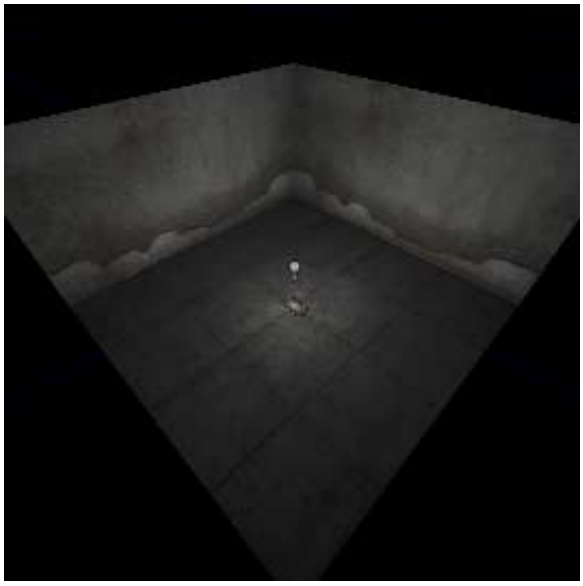
In the previous exercise you placed lights and an object that defined where you entered the world. How did the game know what to do with these? There is code attached to each of these objects that define how it works when the game is running.

MODIFYING ACTOR PROPERTIES

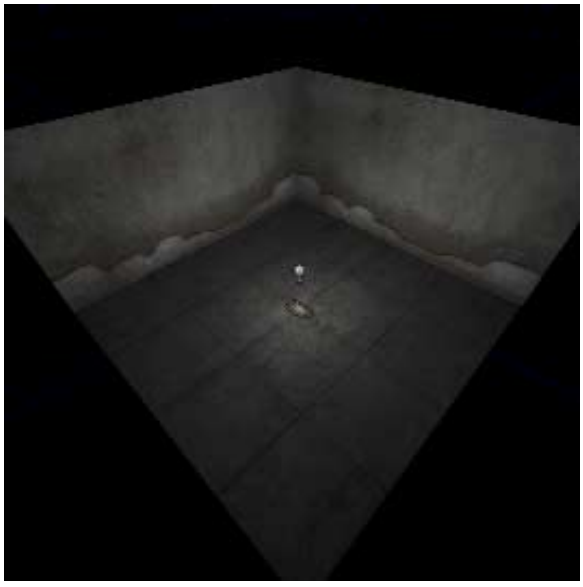
There are also properties that each of these has that modify certain aspects of how they act. For example, in the previous exercise we placed a light. That light has modifiable properties that define how bright it is, what color it is, what objects it can illuminate and many other things. The playerStart actor also has many modifiable properties. Understanding how to work with these properties is key to understanding how to piece together a full level.

EXPLANATION OF WHAT AN ACTOR IS

It is important to understand exactly what an actor is and how to work with it. There are two key things to grasp right now, the notion of base objects and instances. If you look in the Actor Browser you will see an exhaustive list of objects. Each of these are individual objects that have specific functionality coded into them. Most of them are placeable objects though some are not.



Place a Weapon Spawner in the center of the room



Click the Build All button to rebuild the BSP and lighting

EXPLANATION OF INSTANCES

All of these are base objects. They have defined properties and functionality. If you want to use one of them you can select it in the browser and place one in the world. What you are placing in the world is an instance of that actor that you had selected. This is very important to understand. The placed actor uses the actor selected in the Actor Browser as its template. The placed actor is its own separate entity though. This is important because once you have placed it in the world you can modify any of its existing attributes to customize that particular instance of it. When you place a light and modify its settings this is exactly what you are doing, using the base light actor as the template for the settings and functionality of the new light.

EXPLANATION OF INHERITANCE

You'll also notice that in the Actor Browser there are cascading hierarchies of actors. Similar to 3D hierarchies, the actors below another actor are children of the top one. The important thing to understand is that any object that is a child of another has all the functionality of its parent. There are two reasons that children exist in this type of a hierarchy. The first is that the child has added functionality. The second is that the child is a frequently used variant of the parent's properties.

EXERCISE 03 — EXAMINING A HIERARCHY WITH INHERITANCE

The goal of this exercise is to reinforce the notion of class hierarchies and how they work functionally.

- Create a simple box room and texture it

  Create a simple rectangular box area and texture it as you see fit.

- Place a base light actor

  In the actor browser find the light actor.

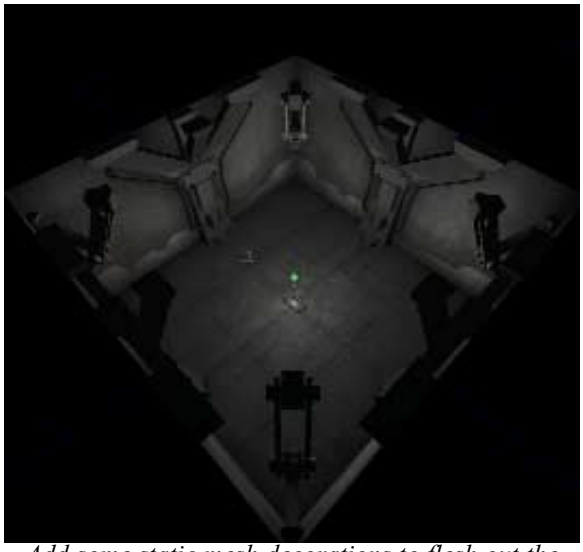
  Place one in the level.

- Place a spotlight actor

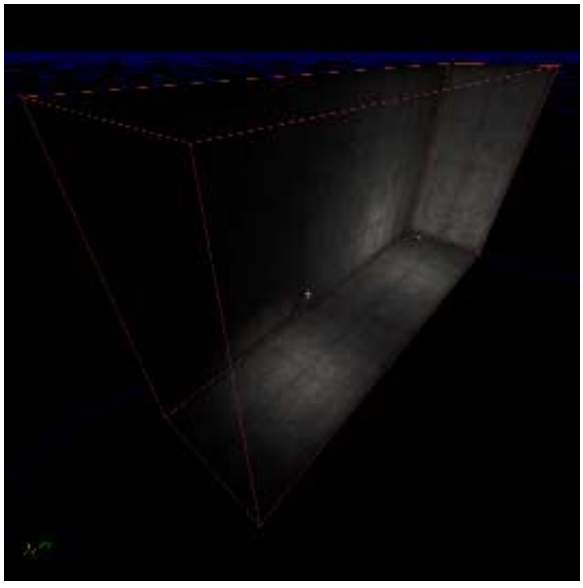
  Underneath the light actor find the spotlight actor.

  Place one in the level.

    - Spotlight is simply an often-used variant of the parent light actor



Add some static mesh decorations to flesh out the room



A spotlight actor on the left and its parent, the light actor on the right

Open the properties window and expand the Lighting rollout.

Select the light and look at its properties, then select the spotlight and look at its properties. The LightEffect variable is different in the child class.

AN INTRODUCTION TO PACKAGES

The goal of this topic is introduce students to the notion of packages and how they will work with them. Of special importance is the Caveats section as the information therein can save the students from the frustration of lost work.

EXPLANATION OF PACKAGES

The notion of packages is a very important one to understand in the Unreal engine. All assets and actors are stored in packages. A package consists of collections of meshes or textures or actor definitions or script or any other elements of the final in game actors. For convenience's sake Unreal separates out different types of packages. For example, there are texture and material packages, there are static mesh packages, and there are actor packages. All of these different package types have different file name extensions. They are as follows:

Actors - .u
Static Meshes - .usx
Skeletal Meshes and Animation - .ukx
Textures and Materials - .utx
Sounds - .uax
Music - .umx

CAVEATS TO WORKING WITH PACKAGES

One thing to note when working with packages. And it happens under very specific circumstances. If you open a level, and based on the contents of that level the editor loads a previously unloaded package, it will only load into memory the objects that it needs from those packages.

The problem you will encounter is that if you import any new content into that package and then try and save it you'll be saving it without all of the unloaded content. If so it will usually give you a message like this. "The file on disk(fileSize) is larger than the one in memory(memorySize). Are you sure you want to overwrite it?"

If you see this message don't overwrite the package unless you are positive you have loaded the entire package. The only way this message will appear if you have imported content correctly is if you overwrote content that exists in a package with content that's data is smaller. For example if you imported a static mesh that the new version of has less geometry, or perhaps a texture that was smaller. Again, the important thing here is that you understand explicitly why you have gotten the message before you accept overwriting the file when the data with is less than it was previously.

The surefire way around this is to stress the importance of executing an Edit→Load Entire Package within the browser before you start to import content into the editor.

EXERCISE 04 — EXAMINE SOME PACKAGES

The goal of this exercise is to show how packages are stored and retrieved and that different types of packages exist in different areas of the directory structure.

- Bring up the Actor Browser and open an actor package...

  Bring up the Actor Browser.

  Perform a File→Open Package...

  - Note that it allows you to open .u files stored in the \System\ folder

    You're automatically placed in the \System\ folder to open .u files

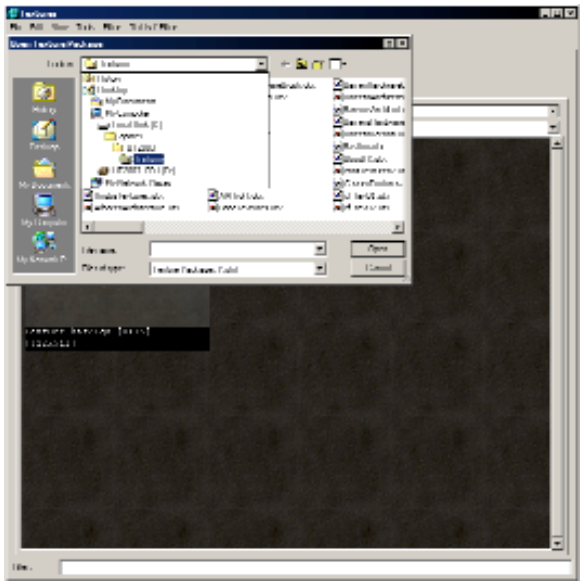
- Bring up the Texture Browser and open a texture package...

  Bring up the Texture Browser.

  Perform a File→Open...

  - Note that it allows you to open .utx files in the \Textures\ folder

    You're automatically placed in the \Textures\ folder.



Examining texture packages

CREATING YOUR OWN ACTORS AND SAVING .U PACKAGES

A spotlight actor on the left and its parent, the light actor on the right

The goal of this topic is to build on what the students know about actors and packages and show them how to create their own for functionality that they define.

WHY CREATE YOUR OWN ACTORS?

Creating your own actor subclasses is a very important feature of UnrealEd. Depending on the work you are doing it is very likely that you are going to working with variants of the base Unreal actor classes. For example, you could have a light that was a specific value that you wanted to be placing frequently. A good way to do that quickly and efficiently is to make one of them, and then save that as a child of the base class.

HOW IT HELPS EFFICIENCY AND CONSISTENCY

This is good for several reasons. In the first place you have saved and made readily available an actor that you ostensibly need to place in many places. Also, in a production where more than one person is working on many maps someone can specify that all light fixtures of a certain type use this particular light class. This ensures consistency.

SAVING YOUR CUSTOM ACTORS

When you create a child of an actor and then tell UnrealEd you want to save it, it prompts you to enter what package it is going to go in. This is a critical part of project organization. The packages you create should be a thought-out and logical grouping of actors. For example you could have a city lights package, a mechanical monster package, or whatever. Once again the important thing is that the packages you create are logical grouping of actors.

EXERCISE 05 — MAKING A CUSTOM SPOTLIGHT ACTOR AND SAVING IT IN A LIGHT PACKAGE

The goal of this exercise is to show the student how to make their own actors and save them to their own packages.

- Open the exercise02finished.ut2 map
- Place a spotlight in the level

  Open the actor browser and find the spotlight actor.

  Place the spotlight in the level and move it so it is approximately 384 units off the ground right above the weapon spawner.

- Modify it so that facing down towards the weapon pickup

  Select the spotlight and in the front view rotate it so that the arrow is point down towards the

- Place a fill light in the level

  Place a regular light in the level and modify it to have a brightness of 32

- Open the Actor Browser and perform a Class→New From Selection...

  Call the package myPackage, and name the actor mySpotlight. Click Ok to finalize.

- Perform a View→Show Packages

  Show the packages panel by performing a View→Show Packages.

- Find myPackage in the list of packages that is displayed on the bottom and check it

  In the list of packages check on myPackage.

- Perform a File→Save Selected Packages

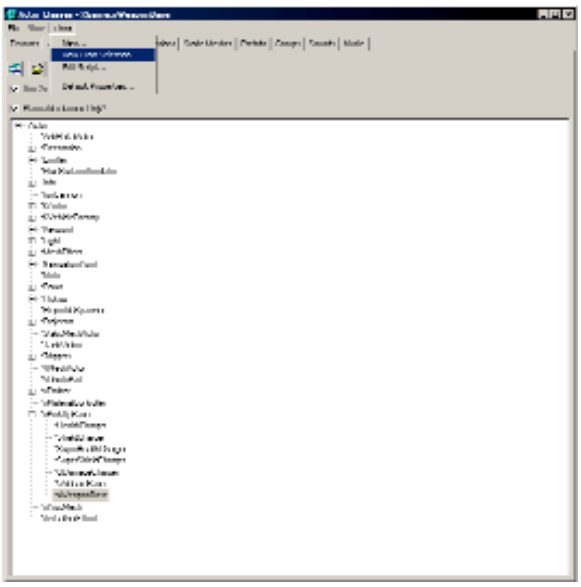
  Perform a File→Save Selected Packages to save that package and all actors in it.

LEVEL DESIGN

The goal of this section is to introduce the student to the role and responsibility of a level designer.

THE IMPORTANCE OF UNDERSTANDING ROLES AND RESPONSIBILITIES ON A LARGE PRODUCTION TEAM

The goal of this topic is to address the pitfalls associated with poor role definitions within a team and how to address them up front.



Creating a new class from the selected actor

TEAMS NEED TO WORK TOGETHER

Critical to the success of any game project is that the team work well together. For a team to work well together it needs to have clear definition and understanding of who does what within the team. Why? Because in a time-critical production environment there is no room for the entire production to slow down because there is ambiguity over who is supposed to be doing what.

WORK DONE TWICE IS BAD

If there is confusion over who does what in a role or which person on a team has responsibilities over a certain area of the production it is entirely possible and in fact likely that work is going to be duplicated. This is bad for several reasons. The first and most important being that it wastes time. The second being it introduces inconsistency into the project and potentially causes asset tracking problems when there are two objects that are made for the same asset.

DECISIONS MADE BY WRONG PERSON IS BAD

Understanding who makes decisions at a given level of the project is very important. There is a hierarchy of responsibility in each project. A producer makes high-level project-wide decisions, like "We're cutting out double jump". A lead makes specific team-wide decisions, like "Bob is going to work on Egyptian Mobile Units this week". A specific member of the team makes decisions about their individual tasks, like "I'm going to use lots of the blue crates in this level". The reason it is important to understand this hierarchy of decision-making is the team needs to know who will be responsible for a given decision so they can talk to that person if there are problems or concerns regarding it. It is also crucial to the success of the project that this decision-making hierarchy is adhered to. Otherwise two bad things happen. People step on other people's toes and tempers flare up, and decisions are overwritten or confused, which damages the smooth production process as a whole.

IMPORTANT TO KNOW WHO IS RESPONSIBLE FOR WHAT

Knowing who is responsible for accomplishing what work is important because it allows anyone on the team to understand who to talk to about problems with an asset or about concerns they may have about upcoming work.

LINES GET BLURRY AT CRUNCH TIME

All this being said, games production is a very hectic environment. When it comes down to it at crunch time the lines between roles get blurry as everyone is trying to get as much done as possible at once. The important thing to remember then is to stick as closely as possible to established process and don't get upset if someone happens to be overlapping into your responsibilities. In the end everyone should be working together to make the best game possible and that's what's important.

THE ROLE OF THE LEVEL DESIGNER

READING – Game Design: The Art and Business of Creating Games, Chapter 5 – Game Design Theory and Practice, Chapter 21

The goal of this topic is to address the role of the level designer from both an art centric and design centric view. The reading will get more into the elements of how to go about doing level design from a theoretical point of view.

LEVEL DESIGNER STILL AN AMBIGUOUS TERM

Level designer is probably the most widely used and loosely defined job in this industry. Everyone knows that a texture artist makes textures, or that a physics programmer writes code to handle physics. But a level designer could be one of several things and more likely is a combination of many. This is mostly due to the fact that for every different engine there is going to be a different way to create levels for them. The level design process is usually the final task in making a game come together and as such requires that all the various elements of the game are integrated together into it. Given that each game has different elements that combine into that whole, it's going to be different for each game.

DIFFERENT COMPANIES HAVE DIFFERENT DEFINITIONS

On top of the fact that different games demand different level design processes, different companies are also going to have their own established way of working. You can really break these down into two different categories though, art centric and design centric.

In the art centric level design the level designers have carte blanche over the whole level. They define how it works and define how it looks. They will usually work with a team of artists underneath them to provide them with what they need.

In the design centric level design the level designer is responsible almost entirely for functionality. They either work with premade art assets or they simply define level functionality and then hand it off to and art team to fill in the aesthetic touches.

THE IMPORTANCE OF AUTHORING WITH BUDGETS IN MIND

The goal of this topic is to drive home the importance of understanding and respecting performance budgets. Also discussed is when it acceptable to and how to fudge performance budgets.

PERFORMANCE BUDGETS ARE THE END ALL BE ALL

Meeting performance budgets is one of the most difficult things to do in the game industry. This is because it is always easy to keep adding to a level to make it more immersive and perhaps longer. However, you need to know the limits of your system and know what targets you are trying to hit performance wise.

FUDGING THEM IS ACCEPTABLE WHEN DONE RIGHT

No matter what hard fast budgets you have you always have room to fudge on them. This can take several forms.

The first is taking a hit in one area to add to another. For example you could take a hit in script execution to give more time to rendering in an area where you wanted it to be gorgeous but there wasn't a lot going on there.

The second is just letting your number drop below your average. This is less attractive because it potentially makes for a poorer game experience. Sometimes though it is a necessary evil to achieve what you want to in an area of the game.

THINKING IN MILLISECONDS

The goal of this topic is to give the students a solid metric to think about performance in, that being milliseconds.

THE IMPORTANCE OF MS AS A METRIC

The one hard fast metric you have as a game developer for the performance of your game is milliseconds. Everything that happens in a game can be expressed performance-wise in milliseconds.

If you want to hit 30 frames per second it's simply a matter of math to determine how many milliseconds you have to work with. 1 second divided by 30 frames comes out to 0.033 or 33 millisecond.

From there you break down the elements of your game into logical groupings from a processing time point of view and figure out what your budgets are.

For example, in the space of 33 milliseconds you may break it down like this:

Rendering: 16 ms
Occlusion: 4 ms
Physics Processing: 3 ms
AI Processing: 4 ms
Script Processing: 6 ms

Within each of those areas there are also further break downs of what is taking up time but you get the general idea. The important thing is knowing that those are your budgets and how exactly you go about monitoring them. Then if you need to, you can fudge them one way or the other in an educated fashion.

MONITORING PERFORMANCE IN UNREAL

The goal of this topic is to introduce the student to the tools used in Unreal to monitor performance.

UNREAL STATS

The way that you monitor performance in Unreal is by using stats. Executing a stat command brings up a display over the game screen that shows you relevant stats to what you are requesting info on.

The command "stat game" will bring up a display that shows you processing times of various gameplay-related systems.

The command "stat render" will bring up a display that shows you processing times of all sorts of rendering systems in the game.

The command "stat all" will display every stat in the game.

The command "stat none" will remove all stats from the screen.

To execute these commands in the editor you need to bring up the command prompt.

To execute these commands in-game you need to bring up the console.

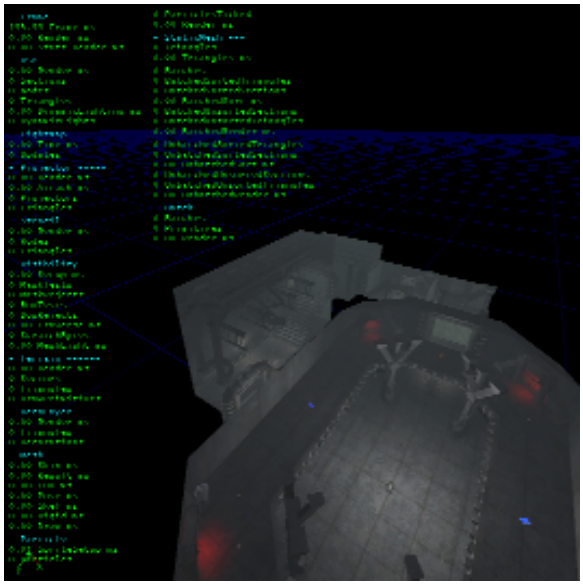
The students will inevitably wonder what all of the stats mean. Most of them aren't terribly important to them, especially the more arcane ones. It's the bigger, all-encompassing stats that they will be concerned with such as Script ms in the stat game readout, or Triangle ms in the StaticMesh section of the stat render readout.

In a production role it's not the designer or artist's job to be able to decipher these stats. Rather it is the job of the engineering staff to tell the content creators what performance monitoring utilities are in the engine and how to use them.

EXERCISE 6 — LOOKING AT STATS IN A MAP

The goal of this exercise is to introduce students to the performance monitoring utilities that Unreal has. Don't spend too much time exhaustively going over all of the stats, focus on the big main ones mentioned below.

Open the exercise06example.ut2 map



Examining performance stats in engine

Examining render stats in engine

Examining game stats in engine

Run the map

Examine render stats

Have the students execute "stat render" from the console.

Some stats of note:

Frame ms : Total time spent in rendering

Stats Render ms : How much time is spent of the frame time rendering stats

The rest of the stats are broken down by functional groups. BSP, projectors, etc. At the top of each of these groups will be the total render ms for them.

Have the students walk around within a map and watch the corresponding jumps in ms as they get into more populated and less populated areas.

Examine game stats

Have the students execute "stat game" from the console.

Collision: Collision is potentially another performance drain. This section allows you to see how much time is being spent on collision resolution.

Script ms: Total amount of time executing script. This is very helpful to ascertain whether or not you're populating your areas with too many functional objects or perhaps with functional objects that aren't properly optimized.

LEVEL BUILDING IN UNREALED

The goal of this section is to teach the student the basics of building a fully functional level in UnrealEd

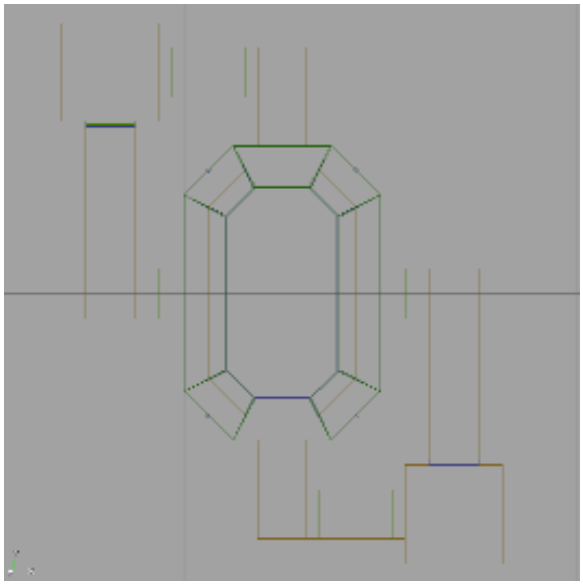
CREATING FUNCTIONALITY

The goal of this topic is to explain how functionality is created in a game engine and how it works together to form a cohesive whole.

HIGH LEVEL EXPLANATION OF HOW GAME LOGIC WORKS

Defining functionality in a game environment is a very time consuming thing. It requires telling objects how they interact with other objects and which objects that they interact with.

A finished functional Unreal level



From a top view we can see how BSP is used to define the basic shape of the game areas

This is either directly and explicitly defined for each object, or complex systems are designed that define how objects interact with one another at game time. Usually it is a combination of both of these things.

HOW THIS CORRESPONDS TO UNREAL

In Unreal it is a combination of complex systems and authored scripting and wiring that defines functionality.

Everything is a child of the actor class. That class has base functionality that allows them to interact with the engine itself. They also have functionality that enables them to interact with one another if they are authored correctly.

UNDERSTANDING EVENT AND TIME DRIVEN SYSTEMS

Interactions in Unreal are driven by two things: time and events. Time is fairly simple to explain. As time progresses you can cause functionality to be triggered on actors.

Events are a little bit different. And for that matter you could say that time is one of the things that triggers events. Events happen when one actor causes another actor to trigger a set of its functionality based on certain circumstances. For example, a trigger objects turns on a light when the player enters the trigger radius.

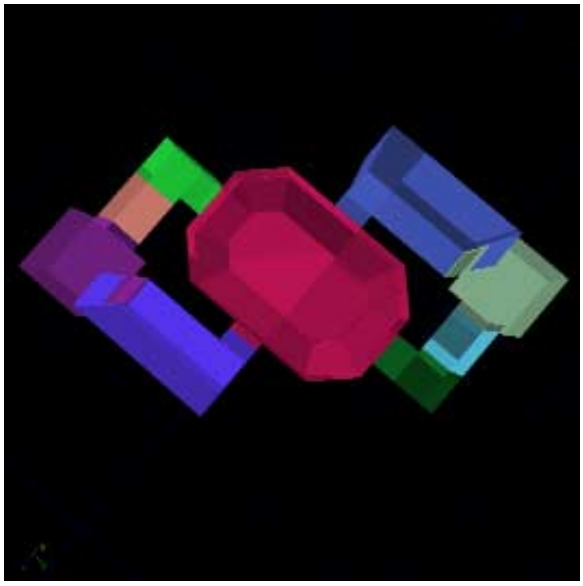
This paradigm expands out into almost all interactions in the game world. The player triggers functionality based on getting with their collision radius, or perhaps by shooting an object. Enemies cause events based on what they do in the world, or perhaps when they shoot the player it causes the player to lose health.

A final game is a combination of many, many scripted and event driven interactions. As the level designer its your responsibility to define which of these elements interact and how.

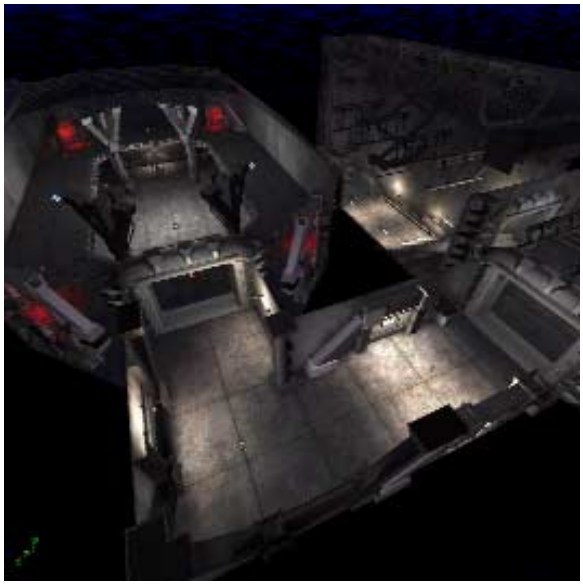
LEVEL DESIGN IN UNREAL

READING - http://udn.epicgames.com/pub/Content/IntroToUnrealEd/

The goal of this topic is to walk the student through the basic elements of level design and allow them to examine a functionally complete level to see those elements in effect.



Zones are authored to help performance



Efficient lighting is used to illuminate areas and highlight focus points

BSP CONSTRUCTION

BSP brushes are critical to working with Unreal. A BSP brush allows you to define an area to either add to a level or subtract from a level.

The world is initially a solid, meaning that in order to make an area you need to define brushes and then subtract the volume of the brush from the world.

You then can subtract other areas or add other smaller brushes back to the areas you have subtracted from.

ZONES

Zones are used in Unreal to define areas that the level is broken down into for performance reasons. It is important to understand them because without proper zoning large levels become performance nightmares.

Essentially creating a new zone allows it and all of its contents to not be rendered at all if the player can't currently see into it.

STATIC MESHES

Static meshes are the primary asset type used in decorating a level in Unreal. This is because they have very little functionality and therefore are easy on processor time and because they are handled by the graphics hardware differently than other types of geometry in Unreal.

ACTOR PLACEMENT AND EDITING

Actors are anything that is not BSP geometry in the world. All objects you place are actors. All triggers you place are actors. All static meshes you place are actors, all lights, etc. Whereas BSP defines how a level is shaped, actors define how a level works.

Actors have many editable properties. How you define their properties affects both their graphical representation in the world and their behavior in many cases.

ANTIPORTALS

Anti portals allow4 the level designer to create sheets and other shapes that clip out anything that is behind them in the client view. This is very helpful for tuning performance in large areas with large decorations or in outdoor areas.

EXERCISE 07 — EXAMINING A FINISHED LEVEL

The goal of this exercise is to allow the students to examine a more polished level and the how all the elements they are going to learn to use are combined into a complete level. Don't focus too much on the details of how individual things are built.

Open the exercise07example.ut2 map

EXAMINATION OF BSP NOTING HOW AREAS WERE BUILT

The BSP used to build the level is simple and efficient. Nothing fancy, all of the fancy details are added later with static meshes.

EXAMINATION OF ZONES NOTING HOW AREAS WERE SEPARATED

Zoning is used to create logical areas that can be completely culled from the client machine should they not be in view.

EXAMINATION OF LIGHTING NOTING HOW LIGHT WAS USED EFFICIENTLY

Lighting is done as efficiently as possible to minimize processor overhead for realtime lighting calculations.

Larger fill lights are used to achieve overall illumination.

Smaller point and spot lights are used to achieve dramatic lighting.

Static meshes are used to add detail to the level

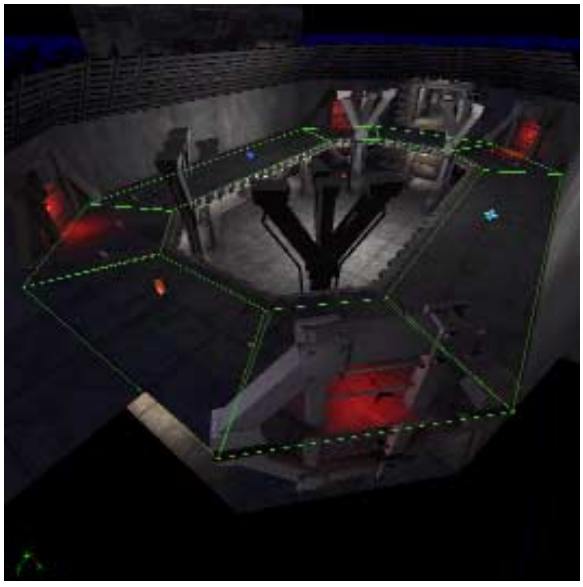
EXAMINATION OF STATIC MESHES NOTING HOW THEY ARE USED TO ADD DETAIL

Static meshes are used to add detail to the environment because they are handled faster than BSP is.

Pretty much all geometric detail is a static mesh element.

EXAMINATION OF ANTIPORTALS NOTING HOW THEY ARE USED TO OCCLUDE AREAS OF THE MAP

Antiportal are used to achieve even more performance gain by occluding objects that exist behind larger elements within a zoned area. The walkway is a good example.



Antiportals are placed in the elevated walkway to help performance

DRESSING AN UNREAL LEVEL

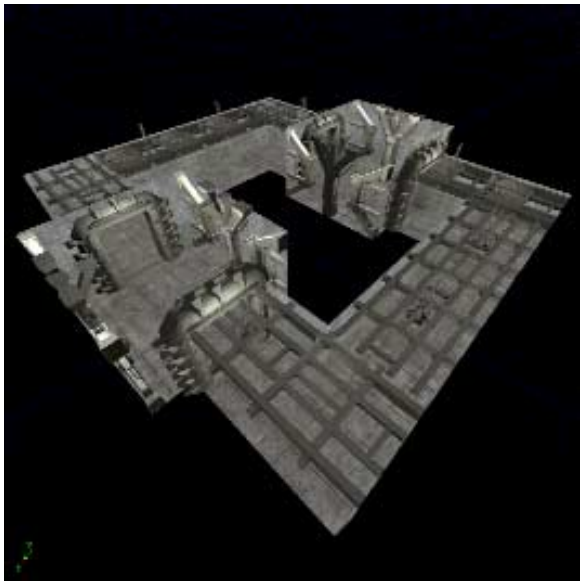
The goal of this topic is to have the student understand how to dress a level with static meshes.

POPULATING WITH STATIC MESHES

Static meshes is what is used to make the bulk of the detail in any given level in Unreal. This is because they are much, much quicker in terms of processor time and rendering time due to the fact that they don't affect zoning at all and are put into memory on a graphics card once and then referenced for every instance of them thereafter.

STATIC MESH BROWSER

The static mesh browser is used to select what mesh you can place in the level. Open it up and select a static mesh you want to place in the level. Then by right-clicking in the perspective panel you can place it at the point of clicking.



A level with BSP and static mesh placement completed

A doorway made from the staticmesh Pipe_Static.General.TubeDoorFrameb

Explanation of how sometimes it doesn't work because of grid snapping.

STATIC MESH EDITING

Static meshes have many variables that affect how the look and act within a level.

Explanation of functional elements of static mesh properties.

Explanation of the aesthetic elements of static mesh properties.

EXERCISE 8 — DRESSING A LEVEL

The goal of this exercise is to allow the students to dress the level both with textures on the BSP and with static mesh geometry. They should be concerned solely with the dressing of and definition of the space here.

CREATE SEVERAL ROOMS ADJOINED BY HALLWAYS

TEXTURE THE BSP SURFACES

Many of the texture packages are separated into groups that make selecting an appropriate texture for a type of surface easy. Suggest that they use the textures from the ceiling group for the ceiling, walls for walls, etc.

POPULATE THE WORLD WITH STATIC MESHES

Have the students populate the world with static meshes to dress the fairly simplistic BSP geometry.

SELECT AND PLACE A DOORWAY MESH

SELECT AND PLACE SOME ROOM TRIM

SELECT AND PLACE SOME LIGHT FIXTURES

ADD OTHER DETAILS AND BUILD THE MAP

Add other static meshes to your liking and then build the map.

SAVE THE MAP

Remember to add a playerStart and then save the map as yourName_exercise08.ut2.

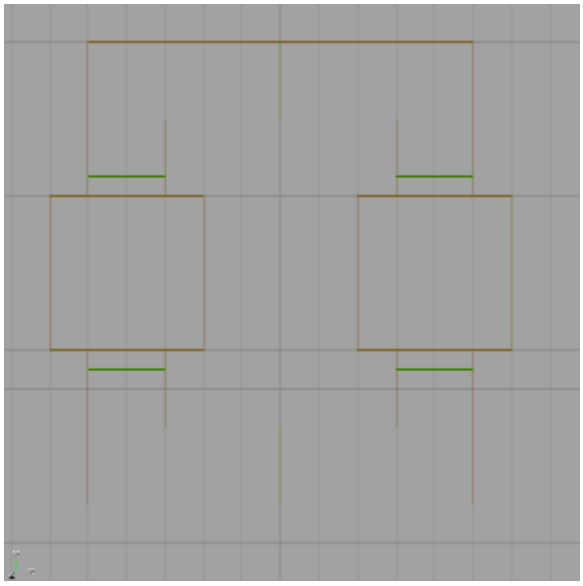
RUN THE MAP

LIGHTING AN UNREAL LEVEL

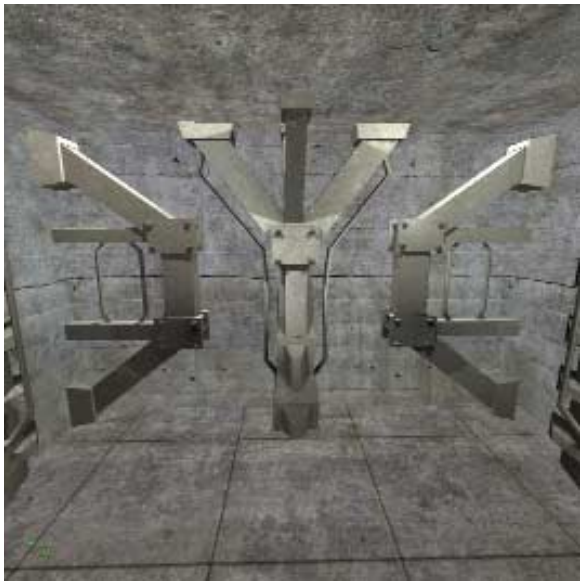
The goal of this topic is to have the student understand how to author lighting in UnrealEd

READING

http://udn.epicgames.com/pub/Content/LightingTutorial/



The room layout for the level



Room trim made from the staticmeshes Pipe_Static.General.TubeSupportE and Pipe_Static.General.TubeSupportc

How lights work in Unreal

A light in Unreal has several important key properties. Its brightness, hue, saturation, radius, and effect. Spot lights also have light cone.

Using these variable you can accomplish almost all of the lighting that you will need to do in a level.

Ambient lighting

On top of lights you can place you can specify ambient light values. This is a good way to help key off the general mood of your level.

Also, ambient values are settable on a per-zone basis. This is done in the zoneInfo node.

Shadows on BSP, how lightmaps work

BSP uses a technique called lightmapping to create their shadows. What happens is the engine determines how the shadows from the light in the area are falling across BSP surfaces and then creates a greyscale texture map to represent it. This allows very accurate, very crisp shadows to be drawn onto every surface.

Given that it is a texture that is being applied as the shadow map you have control over the resolution of the texture. This is so that on critical surfaces you can make it very high resolution resulting in larger shadow map sizes but crisper textures. And on others you can make it very low resolution to conserve memory.

Shadows on static meshes

Static mesh shadows are calculated using vertex lighting. Light is only sampled on a per-vertex basis. What this means is that in order to get smooth lighting across a surface you will need to have a more densely tessellated one so that there are more vertices to sample.

Lighting Only Mode

To see only the light values being applied to either a BSP or Static Mesh surface you can turn on the Lighting Only Mode. This will show you only the light values being applied to surfaces. It's a good way to visualize just how the lighting is working in your level.

Exercise 09 — Lighting a level

The goal of this exercise it have the students complete the lighting in this level to the point that it is considered finished, attractive lighting.

Open the exercise08finished.ut2 map

Light fixtures made from the staticmesh Pipe_Static.General.TubeLightB

PLACE SOME GENERIC FILL LIGHTS

Fill lights should be placed within the center of the room and are generally a little dimmer than your interest lights.

PLACE SOME POINT LIGHTS FOR FIXTURES

Point lights are to provide highlights in areas of interest or on obvious light sources. Point lights are simply regular light actors made a bit brighter with a lesser radius and localized on fixtures or areas of interest.

PLACE SOME SPOT LIGHTS FOR FIXTURES

Spotlights work well right off of ceiling fixtures and make a nice pool of light below them.

BUILD THE MAP

SAVE THE MAP

Save the map as yourName_exercise09.ut2.

RUN THE MAP

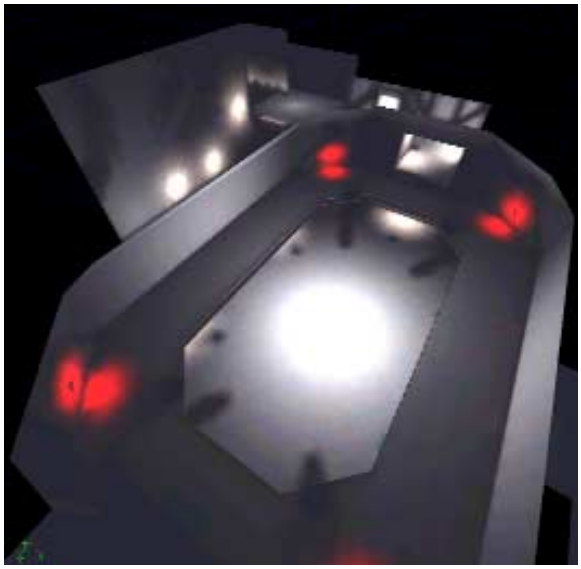
TRIGGERS AND EVENTS

The goal of this topic is to have the student understand explicitly how the trigger and event system works and how to author these interactions.

READING - http://udn.epicgames.com/pub/Content/TriggersTutorial/

HOW TRIGGERS AND EVENTS WORK

Understanding the trigger and event system in Unreal is critical to being able to make fun functional levels. Every actor that is placed in a level has what is called a tick. A tick is basically that actor checking a list of conditions to see what has happened to it or what it is queued up to do.



In lighting only mode you can see just the luminance values of the surfaces

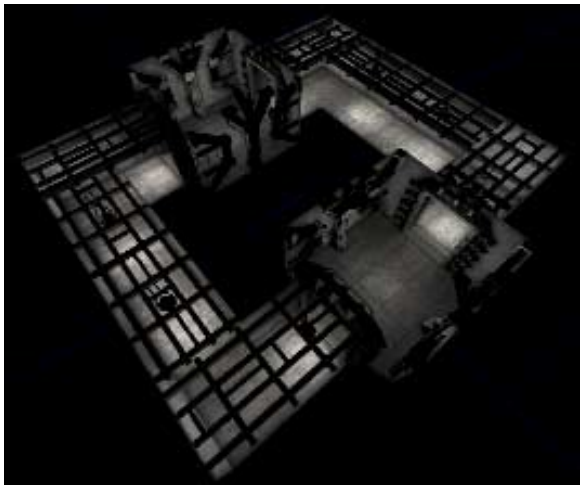
When a trigger is encroached upon it is flagged as being triggered by the engine. The next tick the trigger then knows to fire it event. The event is a variable in the Events rollout of the trigger's properties.

Every actor also has a tag variable, which is also in the Events rollout of that actor's properties. When a trigger is activated it triggers all the objects whose tag variable matches its event variable. Those triggered actors then perform whatever functionality they are programmed to do when triggered.

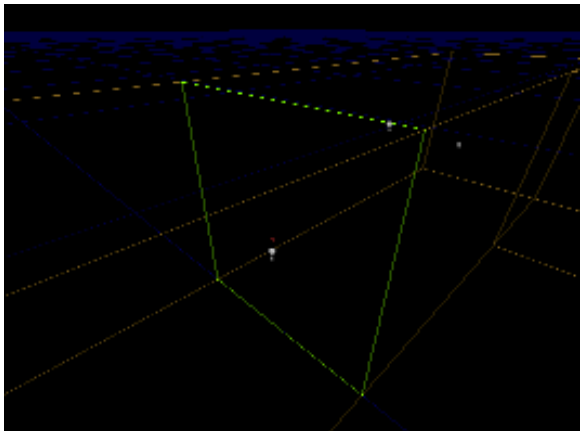
OTHER WAYS OBJECTS ARE TRIGGERED

Some objects also trigger in different ways than simply encroaching the bounds of a trigger object. For example movers, which are covered later, are capable of being triggered when they are touched by the player or other objects.

Fill lights illuminate large areas



The finished lighting for this level



Zone portals are planes of geometry that seal the borders of zones

IMPORTANCE OF CONVENTIONS

The trigger and event system is another one that requires naming conventions. This is because a given map will have dozens if not hundreds of triggers and objects being triggered by them. If someone has to come back to the map who didn't author it or hasn't worked with it in a while, deciphering the relationship between triggers and the actor they work with can be very troublesome.

TRIGGERING DIFFERENT TYPES OF OBJECTS

Every object is going to have its own unique functionality that handles a trigger event. For example TriggerLights have a list of different functionalities that can happen depending on their settings (go to http://udn.epicgames.com/pub/Content/LightingTutorial/#TriggerLights to examine their functionality). Point being that you're going to have to refer to the Unreal tutorials to get a description of what each item's triggered functionality is.

EXERCISE 10 — WORKING WITH TRIGGERS

The goal of this exercise is to allow the student to examine a pre-setup trigger and the actor it is triggering.

OPEN THE TRIGGER EXAMPLE MAP

Open the exercise10example.ut2 map

EXAMINE THE WAY IN WHICH THE TRIGGERS ARE WIRED TO THE LIGHTS

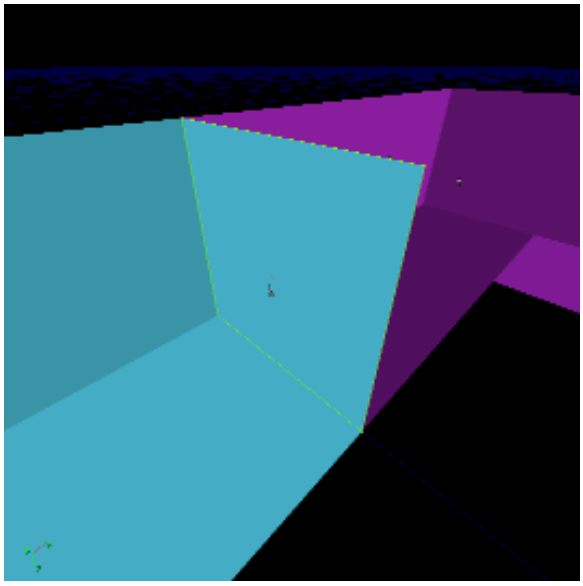
In the perspective window menu perform a View→Show Event Lines (shortcut key E).

SELECT THE TRIGGER ACTOR THAT IS PLACED IN THE MAP

Make note of the line that appears between it and the TriggerLight actor. This is the event connection line, a way of visualizing the triggers and events in your level. This line will only appear when you have a trigger selected, and then only for that trigger's connections.

Bring up the trigger's properties and expand the Events rollout. Note that its event is TriggerLightA.

Leave the properties window open and select the TriggerLight. Note its tag is TriggerLightA.



Zones are differently colored in zone visualization mode to enable you to easily see their separation

Point lights provide concentrated illumination

RUN THE MAP

The Trigger has been left visible (they usually aren't rendered) so that you can see its placement. Run across it and watch the light come on.

MOVERS

The goal of this topic is to have the student understand how movers work and how to author them.

READING - http://udn.epicgames.com/pub/Content/MoversTutorial/

EXERCISE 11 — CREATING LIFTS AND DOORS

The goal of this exercise is to allow the student to examine a pre-setup mover.

OPEN THE MOVER EXAMPLE MAP

Open the map titled exercise11example.ut2.

EXAMINE THE MOVER

Select the mover that is placed as the door of the room. Right-click on it. In the popup menu perform a Mover→Key 1. Note that the mover moves to its second and, for this one, final keyframe.

RUN THE MAP

Have the students run the map and activate the mover to see it in action.

MODIFY THE MOVER PROPERTIES

Have the student modify mover properties of their choosing. Perhaps the move time or adding sounds to the mover.

SAVE THEIR OWN VERSION OF THE MAP

Save the map as yourName_exercise11.ut2.

RUN THE MAP

Have them preview their work.

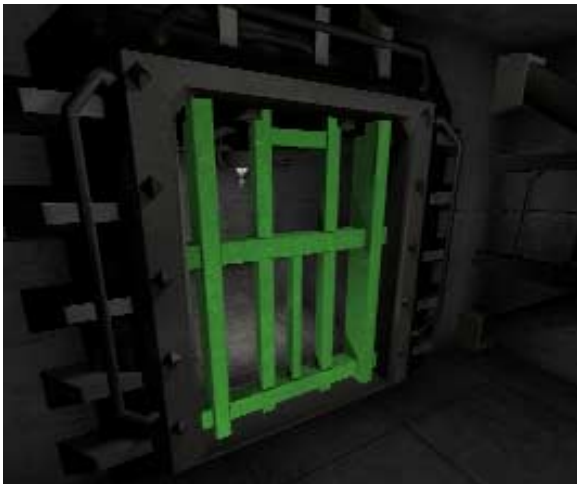
AUTHORING FOR PERFORMANCE

The goal of this topic is to have the student understand how to author for performance. Concentrate on zones and antiportals.

READING - http://udn.epicgames.com/pub/Content/LevelOptimization/

Spot lights provide the directional light that many light fixtures would create

Spot lights provide the directional light that many light fixtures would create



Spot lights provide the directional light that many light fixtures would create

AUTHORING ZONES AND ZONE PORTALS

Authoring zones is critical to making performance reasonable in almost any level you will ever create. They allow you to separate the level into logical chunks so that at runtime those chunks and everything in them doesn't have to be rendered if they are not in the viewable area of the client machine.

Zones are separated by what is called a zone portal. A zone portal is created by making a BSP sheet of the appropriate size using the Create Sheet button, moving the sheet into place so that it intersects all the walls of a BSP brush so as to form a seal, and clicking Create Special button.

VISUALIZING ZONES

UnrealEd has a special visualization mode that lets you see just how the level is being split into zones. Once enabled the viewport will show the level with all the zones being shaded with a different color. This is very handy for figuring out if the zones are being made the way you think they are.

NOTE — ZONE PORTAL VISIBILITY

To turn off the visibility of zone portals in the editor hit the P key with the focus in the perspective view. You can also do this by clicking on the little joystick icon in the perspective view toolbar.

EXERCISE 12 — ZONING A MAP

The map that is created here is going to be the basis for the rest of the exercises. Be sure and have the students understand that they need to save versions of this map as they go as indicated below.

OPEN THE MAP

Open the map exercise09finished.ut2

CREATE A SHEET BIG ENOUGH TO SPAN LENGTH AND WIDTH OF HALLWAY

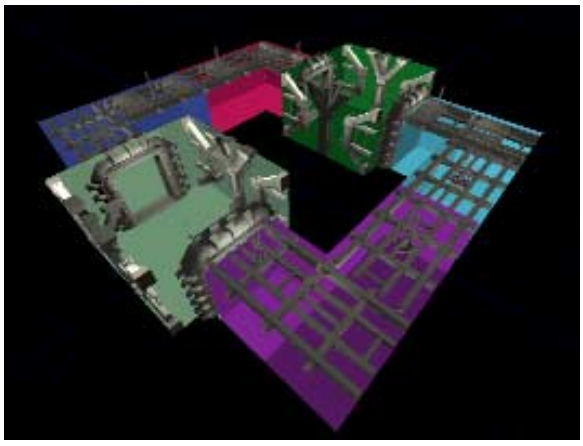
The hallway is 256x256. Make a sheet brush that large. Move and rotate sheet to completely cover hallway as shown on the next page.

CREATE A ZONE PORTAL WITH THE ADD SPECIAL TOOL

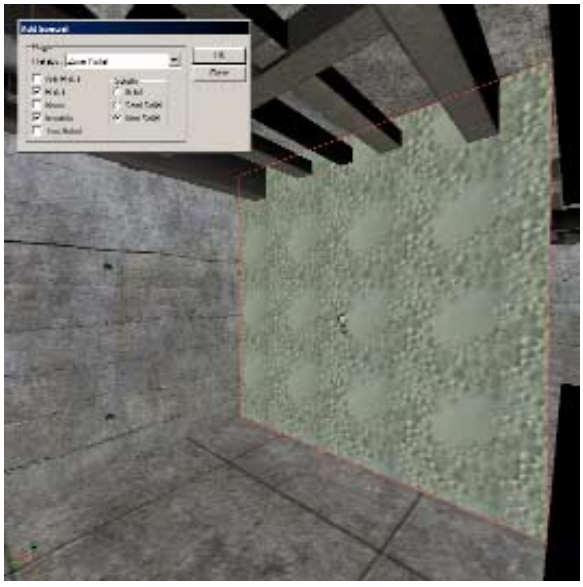
Bring up the Add Special dialog box by pressing the add special button on the left side.

In the Prefabs pull down select Zone Portal.

Hit the Apply button.



A fully zoned map



Create the Zone Portal with Add Special tool

Create a sheet and place it so as to seal the hallway

WASH RINSE REPEAT

Add the indicated zone portals to the map and then rebuild it.

SAVE MAP

Save the map as yourName_exercise12.ut2

RUN MAP AND VIEW IN WIREFRAME TO SEE ZONE PORTALLING IN ACTION

When the map is running bring up the console and enter rmode 1 to enter wireframe mode. Looking around you should be able to see parts of the map pop in and out based on the zone portals you have created.

To go back to normal rendering mode bring up the console again and enter rmode 7.

UNDERSTANDING ANTIPORTALS

One of the most powerful tools you have at your disposal to affect performance and make sure you are squeezing as much performance as possible out of the Unreal engine is antiportals.

Antiportals are a good example of something you may have to do in a game to specifically author performance. Most performance-related things are built to be automated, like zone-based occlusion, for example. However, sometimes it's necessary to take charge of the performance of the engine.

AUTHORING ANTIPORTALS

Antiportals are authored in the same way that zone portals are authored. The only difference is the type of brush you select in the Create Special dialog box.

EXERCISE 13 — EXAMINING ANTIPORTALS AND THEIR EFFECTS

The goal of this exercise is to allow the students to see what effect antiportals have on visible objects.

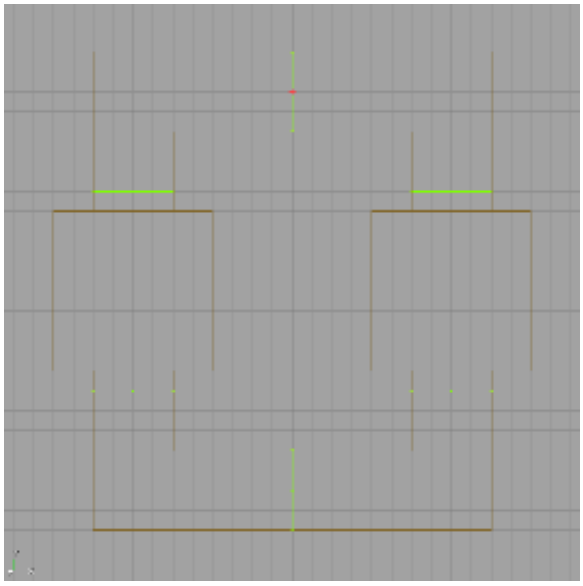
Open the exercise13example.ut2 map

RUN THE MAP

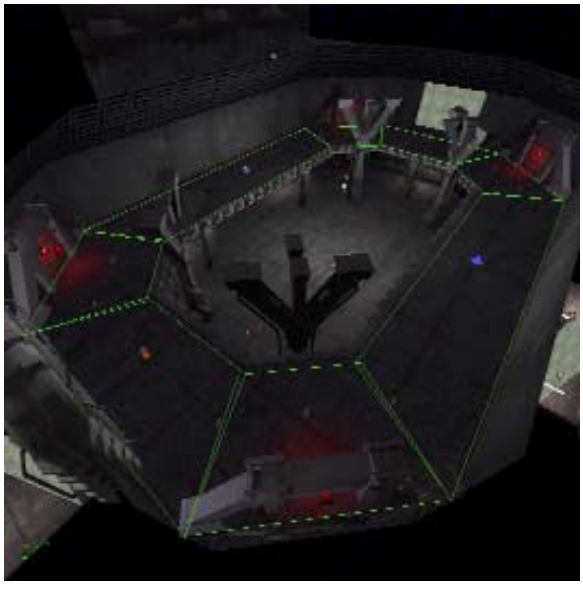
Examine the effects of the antiportals in wireframe mode

Open the console and enter rmode 1 to enable wireframe rendering.

When running under the walkways you will notice that static meshes and other actors are clipped from view above the walkway.



Add the other indicated Zone Portals to the map and then build it



Add the other indicated Zone Portals to the map and then build it

CREATING COMPLEX SYSTEMS

The goal of this topic is to show the student some of the other types of content that can be authored in Unreal. This curriculum doesn't include them as they are beyond the scope of what we are trying to teach, though if you wanted to adapt them into your class, the reading material should give you enough information to do that with.

EVERY GAME IS GOING TO HAVE ITS OWN UNIQUE COMPLEXITIES

Games cannot be created these days without having complex systems that define how things work within them. It's a necessity of the sophistication of today's gamer and what they demand. Every game is going to have its own unique set of systems that define it. A comprehensive breakdown of what these systems might entail would be exhaustive and for that matter impossible given the numerous possible systems there are.

That being the case we are still going to talk about a few of the complex systems in Unreal to give the student an idea of what they may have to tackle out there in a real level design environment.

DESCRIBE SYSTEMS IN UNREAL

In particular we are going to let the student get a taste of both scripted sequences and emitters. They won't spend a lot of class time working on them per se as both subjects are beyond the scope of what we want to cover in this curriculum, but we'll push them in the right direction.

Understanding how to author these is important, and often a role in and of itself.

The important thing is that the student get a taste of what is involved and just how complex the systems can be. Along these same lines it might be a good idea to give the students optional assignments based on both matinee authoring and terrain authoring. But these are both at the teacher's discretion based on time and interest.

MATINEE

Simple description and pointer to online tutorials – beyond scope of this class

READING - http://udn.epicgames.com/pub/Content/MatineeTutorial/

Below is a link to demo content that showcases the matinee system and what it can do.

http://udn.epicgames.com/pub/Content/ExampleMapsEPIC/#CIN_Dolly_Example_Map

TERRAIN

Simple description and pointer to online tutorials – beyond scope of this class

READING - http://udn.epicgames.com/pub/Content/TerrainTutorial/

EMITTERS

READING - http://udn.epicgames.com/pub/Content/EmittersTutorial/

EXPLANATION OF EMITTERS

Emitters are another example of an extremely complex system that can be created within Unreal. Digging into these is FAR beyond the scope of the class we're outlining here. However, the above tutorial has everything in it you need to know to author these systems so should you want to add it to the course it would be possible.

I would highly recommend you stick to the simpler elements of emitter creation such as making a basic particle emitter. Again, these are huge and among the more complicated game systems you will ever encounter.

SCRIPTED SEQUENCES

READING

http://udn.epicgames.com/pub/Content/ScriptedSequenceTutorial/

http://udn.epicgames.com/pub/Content/ScriptedSequenceActions/

http://udn.epicgames.com/pub/Content/AIControllers/

EXPLANATION OF SCRIPTED SEQUENCES

Scripted sequences are used to create complex interactions and chains of events. Say for example you wanted a chain reaction of events to happen after the player triggered an actor. It would be very cumbersome to have to set up multiple triggers and go back and forth between them and edit them.

However, a scripted sequence allows you to trigger a series of events from one triggered object. It also allows you to insert pauses within the events, which is necessary for many complex sequences.

EXAMPLE OF HOW THEY PROVIDE COMPLEX INTERACTION IN GAME

Let's look at an example. Say you want a door to open very dramatically when the player triggers it. First you want steam vents to kick on for a minute and a steam sound to play. Then several seconds later you want latching bars to slide out of in front of the door. Then finally a second or two after that you want the door to slowly slide open.

Normally this type of sequence would be very tricky to execute, but with a scripted sequence keying all of the movement and pauses and whatnot it is a fairly easy process that involves setting up only one controlling object.

CONTENT PRODUCTION - ASSET CREATION AND INTEGRATION

The goal of this topic is to educate the student on what an asset is and give them a high level idea of how an asset gets into a game. This topic also addresses the importance of an elegant and well-defined tools pipeline.

DEFINITION OF ASSET

Assets are what define a game visually and functionally. Assets are what are built, coded, tweaked, and finally placed in a game level. In other words, the definition of an asset is somewhat ambiguous. A good rule of thumb is to think of an asset as being anything that you build to put into a game. Whatever it is. Usually to clarify, people will refer to things as texture assets, or static mesh assets, or sound asset, or... you get the picture.

DEFINITION OF CONTENT

Content is another important term to understand. While there is a bit of overlap between the definition of assets and content you can usually think of content as referring to a group of graphic or sound assets. Content is usually artistic in nature whereas assets can be artistic or functional.

DIFFERENT WAYS ASSETS ARE PRODUCED

Assets are produced in many different ways depending on what type of assets they are. Textures get made in 2D apps like Adobe Photoshop, models get made in 3D apps like Alias|Wavefront Maya, code is written in development environments like Microsoft Visual Studio .net and so on. However, just making these assets is not the only task the person creating them has to do to get them into the game. In fact, it's usually only part of the series of tasks required to get them into a game. These series of tasks are known as a production pipeline. Understanding the production pipeline for a game is key to making that game successful. As a matter of fact, many games that fail can directly link the failure to a lack of definition or education of the team regarding the production pipeline.

IMPORTANCE OF SMOOTH PIPELINE

Defining an elegant and well-integrated production pipeline before production ever begins is crucial to the smooth execution of the production portion of a game project. Have games ever been shipped without an elegant, well-integrated, or even finished production pipeline? Of course they have. However, anyone who worked on those projects could tell you unequivocally that the project would have run smoother and the resulting product would have been higher quality had they put the appropriate amount of work up front into designing and fully implementing their production pipeline.

HOW A SMOOTH PIPELINE HELPS CONTENT PRODUCERS

The reason for this is simple. A well-designed production pipeline enables the people creating content to do it in as simple a way as possible, with as few steps as possible, with the fastest turnaround times possible. The longer an artist has to wait to see the results of what they just created in the engine the more time is wasted on the project as a whole.

EXAMPLE OF TIME WASTED BY BAD PIPELINE AND TOOLS

If you figure that an artist needs to preview their work in engine at least twenty times a day, and each of those times takes a minute and a half as opposed to fifteen or even thirty seconds you can say that 20 minutes a day is being wasted per artist. This seems trivial at first. But when you compound that by, say, ten artists on a team by, say, 275 days for a production (and that's a relatively short one) then you have a grand total of 916 hours wasted over the course of a project. That's almost 6 man months. And those are conservative estimates across the board.

You can see how this type of an effect can snowball. And then when you factor in employee morale and wasted work over mistakes that inevitably happen the more steps are involved in a process... You get the picture.

This was simply to demonstrate the importance of a good pipeline, largely that is not in your realm of responsibility as an artist. Not to say you can't stress how important it is to the people in charge but usually you're not the one making these decisions. However, you are the one who has to work in whatever pipeline is decided upon for a project. As such it is important for you to understand exactly how to work in that pipeline as efficiently as possible.

EXAMINING A SAMPLE ART PRODUCTION PIPELINE

The goal of this topic is to show an example pipeline and define what steps exist in it and why.

EXAMPLE PIPELINE STRUCTURE

Now we're going to take a look at a sample production pipeline. This entails taking one asset all the way from a twinkle in a designers eye up to the point when it is a finished piece of a game. Note that this is just a sample and is certainly not a be-all end-all description of what a production pipeline should look like.

INITIAL CONCEPT

This is the stage where the designer comes up with the idea of what they want. At this stage it's usually just a vague idea, and for that matter, what's envisioned at this stage is usually changed dramatically by the end of this process as well.

What comes out of this stage is a line item on a list at a design meeting to happen in the near future.

DESIGN

This is a very important step. This is when the representatives of each team sit down, and hash through the nuts and bolts of how the asset is going to work and be made. Design is present to make sure that the functionality is what they need to make the gameplay they are envisioning. Engineering is present to make sure that what is designed is feasible in terms of what the engine is capable of and. Art is present to make sure that what is designed can be made into something aesthetically pleasing. And someone else is present to mediate the whole debacle because these meetings can turn ugly when people passionately hold opposite views on matters. Usually this is the internal producer on the project.

It's possible at this stage that the concept will just not work. This could be for a number of reasons. Whatever the reason may be, it is important to make that call now so that work is not wasted.

What comes out of this is a functional design to be executed on by a designer and if necessary an engineer working in concert.

FUNCTIONAL PROTOTYPE

At this stage both design and engineering work tightly together to flesh out the functionality of the asset. Is it rising to quickly? Is it too small? Is it just not possible? All these and more are questions that have to be answered.

If it is determined in this stage that the functional design cannot be executed then everything goes back to the design phase to revisit the concept and see how it can be better realized.

What comes out of this is a proof of concept. This is a working version of the asset. It's not pretty yet, but it does what it's supposed to.

FINAL CONCEPT

This is the stage where all the functional aspects of the asset are done. Now its time for art to figure out how to dress it appropriately for the setting while respecting the functionality that has been built.

If art has huge issues with the functional implementation of the asset at this point it can ask for design revisions. However, it is usually the case that form serves function in the world of games so this doesn't happen too often

What comes out of this is a final art concept that the art team then uses to build the final art assets out of.

PRODUCTION LEVEL 1

This is the first pass at building the aesthetic of the asset. This stage is important because no matter how much planning went into this asset up until now there hasn't been a truly representational whole until this is done. It is important to note that sound for the object need to be down as well here.

If after this is done, there are problems, be they gameplay, functional, or artist the asset may regress to the appropriate stage. Usually however an asset is well on its way to be finished by now, and sending it back up the pipeline is not done often.

What comes out of this is the first pass at the finished object. All functionality, art, and sound are considered done for the object.

PRODUCTION LEVEL 2

This stage is present to allow any last-minute tweaking to be done on the asset, be it functional or artistic. This is here because it may very well be that once the asset is in place in the level there are artistic elements of it that clash with another. Or perhaps the behavior of it feels a little bit off in relation to the rest of the level. Whatever the case may be, last minute small changes are what are supposed to happen here.

It is important to note that major revisions to the object should not happen. If major changes are required at this point then the process failed in one of the previous steps and it should be determined how and at what point they happened so it doesn't happen again.

What comes out of this is a final shipping asset. It shouldn't be touched at all after it is done with this stage.

FOLLOWING AN ASSET THROUGH THE PREVIOUS ART PIPELINE

The goal of this topic is to look at the life of an asset as it goes through the previously discussed pipeline.

Look at an asset through the previous pipeline

Now let's look at the life of an actual asset through the previously discussed pipeline. Let's take a floating mechanical platform as an example. This is just a generic example, don't spend too much time over the details of it.

INITIAL CONCEPT

This could take the form of anything from "So I was thinking it could be a mechy sort of machiney like floaty thing. Does that make sense?" to "Alright, I got these pictures off this artist's web site. I really like the anti-grav unit dealie on the bottom, and the tubes are really cool too. I don't care for the top much though, can we redo that?"

DESIGN

At a design meeting the Mechanical Lift idea is discussed. Design lays down their idea for it. They want a lift device that gradually accelerates as it starts and then gradually decelerates as it stops. Art feels that it should respond to the weight of the player when they step on top of it. Design thinks this is a cool idea and engineering says it will see what they can do to give it those physics. Art thinks that it should have some glowing ring and streaks trails below it as it moves. No one has any objections to that. In the end it is decided that all of the things that design wanted and that were added on top of the original concept are feasible and the producer puts the asset into the schedule.

FUNCTIONAL PROTOTYPE

Just having a moving object that starts when the player touches it is easy in the engine and the designer makes one pretty quickly. However, the added functionality of having it bob when the player steps on it means that engineering is going to have to build a new kind of object. So this happens first. Once that is built the designer toys with it and tweaks some of the properties that engineering gave them on the object like bounciness and mass until they think they are happy with it. It turns out though that making it Bob has screwed up some of the collision detection that sends trigger events. This means that engineering has to do a bit more work to fix that. Once this is working the designer determines that the initial dimensions of the object are too big for the challenge it is supposed to represent in the game. So they make the primitive art asset that is serving as its mesh a bit smaller. In the end though there is a working lift ready to be made beautiful.

FINAL CONCEPT

The art director begins doing the concept work for the final asset. However, he is concerned at the smaller size that resulted in the functional prototype phase as it means he really doesn't have enough room to build the details that were initially envisioned. After some back and forth with the designer it is determined that the smaller version of the asset is more appropriate for the gameplay and so the art director will just not add some of the details that were initially envisioned. He finishes up some concept work and then it's off to the art team.

PRODUCTION LEVEL 1

Several things happen in concert here. Art makes the assets necessary to create what the concept illustrates. And sound makes the sounds that begin to bring the object to life. When it's all said and done the object looks like it should ship with the game. All that's left to do know is see how it integrates with the game world.

PRODUCTION LEVEL 2

It turns out that the texture is a bit dark and the object doesn't stand out enough so it is made a bit brighter. Also the sounds are a bit too high-pitched and tinny for something that is moving at such a slow pace so they are taken down a notch. But once this is done the asset looks and sounds at home in the game. Ship it.

THE ROLE OF A PRODUCTION ARTIST

The goal of this topic is to educate the student about the role and responsibilities of a production artist and what they can be expected to do at different phases of the production.

READING

Game Design: The Art and Business of Creating Games p. 167 – p. 174

DESIGN

There are different responsibilities for an artist during the design phase, and in many situations multiple artists share them.

The first of which is simply inspiration. The creative mind of an artist is usually very helpful when dreaming up whatever fantastic items are going to be created in the game.

The second is the reality check. This one is a little trickier and usually falls to a more technically minded artist. What happens here is the artist does a sanity check on the idea in question and tries to identify any problems with it. A problem could be in the form of performance concerns, sustainable production concerns, or just that the idea doesn't fit the theme of the game. Whatever the case may be, having someone who understands and represents these concerns on hand during the design phase is invaluable.

CONCEPT

During the concept phase the artist gets to do what they do best, create art. At this point there is a general idea of what the asset in question is going to be, there might even be simple scratched out sketches of it. But it's the artist job to make a fully realized concept design. There is usually a bit of back and forth between art and design here as design will often have specific things they want to see represented or addressed with the asset. These will most likely be things that affect playability and whatnot.

PRODUCTION LEVEL 1

This is where the artist makes the first fully realized in game version of the asset. The goal here is to have something that is considered to be done. The next level is present to account for changes that might need to be made, but by all accounts the object should be considered to be finished after this.

PRODUCTION LEVEL 2

And finally we have the polish phase. This is critical to have in the schedule as it is where there is room to make all the last minute tweaks and changes that inevitably happen once everything is integrated into the game or into a level.

It should be stressed that at this phase only minor changes should be happening. If major changes need to happen everyone should understand that this takes the whole process back to one of the first two stages and understand the impact that has on the schedule.

THE CONTENT TREE

The goal of this topic is to explain what a content tree is and instill in the student the importance of respecting and maintaining convention in the content tree.

EXPLANATION OF WHAT A CONTENT TREE IS

The content tree is where all of the assets for the game live before they get put into the game. It has all the geometry files, texture files, sound files, etc.

The important thing about a content tree is that it is laid out in a logical way. If at all possible it should be laid out in a way that represents how the assets within it are represented in the game.

For example, let's say the head of your content tree is a folder called \Content\. Underneath \Content\ is where you will store all of the folders that represent the packages in your game. Now let's say in Unreal you have a static mesh package called HumanResistanceTechS, then you should have a folder called \Content\HumanResistanceTechS\. Underneath that, in turn, will be the files that represent the objects or groups of objects in that package.

The reason this is so important is that in a game with hundreds of assets it becomes very important towards the end of the project to be able to find source files for assets very quickly, and without good organization this can become a nightmare.

THE IMPORTANCE OF CONVENTION

The goal of this topic is to instill in the student the importance of naming and storage conventions and why they are critical to a successful production.

INTRO

Over the course of a project hundreds if not thousands of files will be created. Image files, 3D files, files that contain line after line of code, etc. In short order these can become an absolute mess to manage and keep track of. There are two tools that you have to keep these files from becoming an unmanageable nightmare. Naming conventions and storage conventions.

NAMING CONVENTIONS

Naming conventions are important because they allow anyone on the team to understand exactly what to name something when it comes time to do that. It also means that anyone looking for an asset after it is finished will have good idea of what it will be called.

STORAGE CONVENTIONS

Storage conventions are important because when someone needs to find a file in the midst of the thousands that comprise the projects, they can use the rules set down in the storage conventions to quickly figure out its location in a resource tree, which is itself likely comprised of hundreds if not thousands of folders.

PUBLISH CONVENTIONS TO TEAM

The important thing to note here is that all naming and storage conventions should be simply and clearly defined and in general make sense. There are going to be problems unless you have a clearly defined rule structure for how things are named and where they are stored. Before production is underway these should be defined and published to the team in a public and readily-available place. This becomes critical when you have several people working on different elements of the same scene or group of objects in a game production.

FILE REVISIONS

Another very important element of production to iron out beforehand is how file revisions are tracked and stored. That is to say, if someone creates 4 working files over the course of creating a final file where are those 4 files stored and where is the final file stored? How are those files numbered? All these are things that need to be defined so that there is consistency across a project.

FURTHER REINFORCEMENT OF IMPORTANCE

It may seem that this is all very nit-picky and something that surely you shouldn't be concerned with but once you've had to manage an asset tree that is comprised of thousands of files stored in hundreds of folders you'll realize the value of this. The benefits reaped from establishing a system are numerous.

IMPORTING TEXTURES

The goal of this topic is to educate the students how to import texture content in UnrealEd.

READING

http://udn.epicgames.com/pub/Content/UnrealTexturing/

http://udn.epicgames.com/pub/Content/TextureSpecifications/

http://udn.epicgames.com/pub/Content/TextureComparison/

EXERCISE 14 — IMPORTING TEXTURES

OPEN THE TEXTURE BROWSER

In the package pulldown select the ChameleonT package. When importing textures UnrealEd automatically fills in the package entry in the import dialog with the currently selected package.

PERFORM A FILE→IMPORT TEXTURE...

In the menu bar for the Texture Browser perform a File→Import…

FIND THE DESIGNATED FILES

In the ChameleonContent/ChameleonT/ folder select all 4 texture files.

HIT OK ALL

Hitting Ok All will import all 4 of the textures and name them as the file names. This is another reason that proper naming conventions are important.

UNREALED MATERIALS

The goal of this topic is to point students towards reading and demo material about the UnrealEd material system. It's not covered in the body of this curriculum but should you want to adapt it everything you need to do so with is below.

READING - http://udn.epicgames.com/pub/Content/MaterialTutorial/

MATERIAL DEMO MAP

http://udn.epicgames.com/pub/Content/ExampleMapsEPIC/#Materials_Examp

This is a link to a zip file that has demo content for the material system.

5.4 Quizzes and Projects

Final Project

Explanation of project

This project is to allow the students to do two critical things: build a map from scratch all the way through to blowing each other up in it; and create content and see it get into the game they are playing.

Depending on the size of the class, it may be a good idea to split it up into several groups that are working on separate maps and weapons concurrently so that everyone is able to make something.

Critical to the completion of this project is that you give the students a schedule that they have to adhere to. Otherwise, inevitably what happens (ironically just like game development) is that they will spend all of their time in the design and prototyping phases and never complete any of this. The completion of this project is as much a goal as the quality of what they produce. In fact, completion is almost more important than quality. The reason being that the last 10% of any project is the hardest to push through. Establish a quality bar that is attainable in your timeframe and stick to it, don't exceed it.

It should be explicitly understood that the schedule is to be adhered to without fail. If the students want to come back in their off time and fix something, that is acceptable, but it should be outside of scheduled class time.

The focus here is threefold, all of which are equally important: creating content, staying on schedule, and completion. Point out that this is similar to a mini-production, and just like real life if you slip there are consequences. If they stress a bit, all the better.

Building a map

Step 1 — Map Design:

This is where the students will sit down and design an area that they think would be fun to play in. The important thing to keep in mind here is that they will have a somewhat limited amount of time to create this. Use the exercise08.ut2 map as a good gauge of about the right size. A bit bigger perhaps but not too much.

This step is done when the students have a map on paper that they want to build and that seems feasible.

Step 2 — Prototyping:

This is where the students screw around with their ideas and see what works and what doesn't. Any complex behavior they want to include in the maps should also be ironed out here to see if they can make it work.

The goal of this is to iron out any technical details or hindrances to the plan the students have. This is an important step, as it will drive home the importance of working around problems and potentially cutting things that don't work or are too time-consuming.

This step is done when the students know exactly how to do all the things they want to do functionally in their map.

Step 3 — Map Building:

This is where one student per map will begin to integrate the ideas into a solid whole. It is possible to have multiple students working on separate portions of a map and then integrate them together, but be sure that you understand fully how this process of copying from one map and pasting into another works, otherwise you are asking for more headaches than it is worth.

This step is done when the students have a completely functional map that is ready for playtesting as a whole.

STEP 4 — PLAYTESTING:

Once the map is considered complete it is time for play testing. This is really as simple as playing the map. And then playing the map again. And then... you get the idea. The goal here is to identify problems both with functionality and with the general flow and gameplay of the map. Revision will happen again and again in this portion.

It's also worth noting that the prototyping phase should be complete with the weapon before this begins. The reason being that the students are going to be playtesting gameplay against the weapon they have designed. If that design is still in flux than they are trying to hit a moving target and are making more work for themselves than they should. As part of the schedule this task shouldn't begin until that one ends.

This step is done when the students are confident that the map works the way they want it to.

STEP 5 — POLISH:

Once the map exits for playtesting, it's time to polish it. This means going through and tweaking static mesh placement, tweaking light values, etc. It's very important that the students understand exactly how much time they have to do this. This phase can drag on and on because really you can polish something endlessly. Give them a time constraint and work with them to help them organize their time within that.

This step is done when the students meet their desired polish level.

STEP 6 — GOLD:

This is when they are done. It is important to make a big deal out of this. Few games make it this far and few student projects do either.

BUILDING A WEAPON

STEP 1 — WEAPON DESIGN:

This is where the students will design what the weapon is. Keep in mind what the design restrictions are. The specifications and limitations of the chameleon weapon are detailed below in the section titled Chameleon Weapon.

This step is done when they have a concrete functional design to execute on.

STEP 2 — WEAPON PROTOTYPE:

This is where the students play around with the design they created and try and make it a functional reality.

This step is done when the weapon "feels" the way the students want it to.

STEP 3 — WEAPON CONCEPT:

This is where the students will design the look of the elements they are going to build for the chameleon weapon. Namely the weapon model itself, the ammo model, and the projectile model.

This step is done when the students have concept work for all of those elements.

STEP 4 — PRODUCTION LEVEL 1:

This is where the students begin to build all of the elements to create the complete chameleon weapon. It is important to make sure that someone is assigned to finish each and every piece so that they can understand exactly how much work goes into a finished asset, let alone a finished project.

Emphasize the fact that it is more important to finish all of the elements individually before iterating on the whole. That's what the next step is for.

This step is done when the students have modeled and textured assets for all of the components of the chameleon weapon.

STEP 5 — PRODUCTION LEVEL 2:

This is where the students look at the integrated whole and tweak as necessary.

This step is done when the students are happy with the look and feel of the weapon.

STEP 6 — GOLD:

Same story as above. This is when they are done. It is important to make a big deal out of this. Few games make it this far and few student projects do either.

ROLES AND TEAMS

The students should be broken down into production teams. Each team will consist of key members with important roles. Based on the amount of work to be done it is entirely possible and in fact likely that some students will fill multiple roles. Again much like a real life production.

Producer – This is your job as the instructor. You take on the responsibility of driving the entire production. You figure out how best to have your team execute on the plan they come up with. You should also help them sort out what the team breakdown is.

Game Designer – This role is defining the feel of the chameleon weapon. Given that the design is part of the fun of this exercise everyone should be able to participate that wants to, but one person should be responsible for that task of finalizing the design based on everyone's input.

Level Designer – This role entails both designing and building the map that the students will finally play in.

Artist – This role entails doing concept work, modeling, and texturing the design that the group comes up with.

SCHEDULING, AND TASK ASSIGNMENT

The schedule break down into two concurrent task lists with one overlapping dependency between the two, that being that play testing can't happen in the map building pipeline before the weapon prototyping is finished.

Make sure that the students understand this and understand the dependency between the two as it reflects an actual real life production process where many things in the production are contingent on the completion of many other things.

Once roles and teams are assigned the task breakdown becomes pretty clear based on the steps outlined above. You'll want to schedule your class at a pace that gives appropriate time to spend in each step but that also doesn't allow the students to dally in one area or another. If they finish something to satisfaction have them move on. As far as a scheduling breakdown, the time should work out about like so:

15% Design and concept
30% Prototype
30% Building
15% Polish
10% Fudge Factor

This is the first time we've acknowledged the fudge factor in scheduling. It's always there, everyone knows it should always be there. But you should never rely on it. This is something that should be stressed to the students. Relying on that means your scheduling for that and if it's in your schedule it means there is no room for fudge factor. It's a bit of a paradox, but just pretend like it doesn't exist. You'll use it all up, trust me.

FOOTNOTES AND ERRATA

The Unreal exporters can be funny sometimes so if anything bizarre happens restart the 3D package and re-export.

The Maya exporter tends to have problems working with more than one scene file for skeletal animation in a given session. It's best to restart Maya PLE again for every individual skeletal mesh you need to export.

When exporting content to existing packages make sure you go into UnrealEd first and load the entire package. Otherwise you'll export your information and UnrealEd will create a new package called whatever you have the package listed as. When you go to save that package you'll be saving whatever you just exported over all of the existing data.

WORKING WITH THE CHAMELEON WEAPON

There are several important elements to working with the chameleon weapon. The classes are designed to give the students one place where they can modify all of the elements of the weapon, including the ammo pickup.

Following is a brief user's guide.

To place a chameleon weapon in a level:

- First make sure you have loaded Chameleon.u in the actor browser
- In the actor browser find xPickUpBase→xWeaponBase
- Place one in the level
- Open the properties of the newly place xWeaponBase
- In the xWeaponBase rollout under WeaponType select Class'Chameleon.ChameleonWeapon'

To modify the chameleon weapons base properties:

- Open the Actor Browser
- First make sure you have loaded Chameleon.u in the actor browser
- Uncheck the "Placeable Classes Only?" checkbox
- Uncheck the "Use 'Actor' as Parent?" checkbox
- Find the ChameleonWeaponConfiguration object
- Right-click on it and select Default Properties...
- This will pull up a dialog that has all of the relevant properties for the object.
- NOTE – Sometimes when View→Show Packages is checked on the "Placable Classes Only?" checkbox disappears. This is a bug. Simply turn off View→Show Packages to get the checkbox back again.

To save the ChameleonWeaponConfiguration object:

- Perform a View→Show Packages
- In the panel that appears in the lower portion of the Actor browser check the Chameleon entry
- Perform a File→Save Selected Packages

NOTE – Sometimes when View→Show Packages is checked on the "Placable Classes Only?" checkbox disappears. This is a bug. Simply turn off View→Show Packages to get the checkbox back again.

And here is a list of the entries and their respective functionality

- ChameleonAmmo
    - InitialAmount
        - The amount of ammo in the weapon when you get it
    - MaxAmmo
        - The maximum ammo the weapon can carry
    - PickupAmmo
        - The amount of ammo in an ammo pickup
- ChameleonGeneral
    - AmmoPickupSound
        - The sound played when the ammo is picked up
    - SelectSound
        - The sound played when the weapon is selected
- ChameleonProjectilePrimary
    - AmbientSound
        - The sound that is looped while the projectile is in the air
    - Damage
        - The amount of damage the projectile does with a direct hit
    - DamageRadius
        - The radius from the point of impact that the projectile will cause splash damage in
    - ExplosionDecal
        - The decal left where the projectile strikes
    - ImpactFX
        - The emitter that is spawned when the projectiles strikes
    - Physics
        - What type of physics the projectile uses.
        - PHYS_Flying moves straight forward from the point of spawning at whatever rate the Speed variable is set to.
        - PHYS_Falling flies forward at the rate the Speed variable is set to but also is subject to gravity.
    - Speed
        - The rate in units the projectiles moves per second
    - TravelFX
        - The emitter spawned to make a trail as the projectile moves
- ChameleonProjectileSecondary
    - SAmbientSound
        - The sound that is looped while the projectile is in the air
    - SDamage
        - The amount of damage the projectile does with a direct hit
    - SDamageRadius
        - The radius from the point of impact that the projectile will cause splash damage in
    - SExplosionDecal
        - The decal left where the projectile strikes
    - SImpactFX
        - The emitter that is spawned when the projectiles strikes
    - SPhysics
        - What type of physics the projectile uses.
        - PHYS_Flying moves straight forward from the point of spawning at whatever rate the Speed variable is set to.

        - PHYS_Falling flies forward at the rate the Speed variable is set to but also is subject to gravity.
    - SSpeed
        - The rate in units the projectiles moves per second
    - STravelFX
        - The emitter spawned to make a trail as the projectile moves
- ChameleonPrimary
    - AmmoPerFire
        - The amount of ammo used per weapon fire
    - FireRate
        - The rate in seconds after one fire before another can happen
    - FireSound
        - The sound that happens when the weapon is fired
    - FlashEmitter
        - The emitter that is spawned when the weapon is fired
    - ProjSpawnOffset
        - The offset based on the client's local space that the projectile and the FlashEmitter are spawned.
        - X is to the left
        - Y is up
        - Z is toward the clients view
    - ShakeOffsetMag
        - This is the magnitude in units that the screen shake will randomly move in when fired. XYZ as above.
    - ShakeOffsetRate
        - This is how rapidly it will move there in units per second when fired. XYZ as above.
    - ShakeOffsetTime
        - This is the rate at which the view will recover on it own
    - ShakeRotMag
        - This is the magnitude of the rotation that the screen shake will randomly move in when fired. XYZ as above.
    - ShakeRotRate
        - This is how rapidly it will move rotate when fired. XYZ as above.
    - ShakeRotTime
        - This is the rate at which the view will recover on it own
    - SmokeEmitter
        - This is the emitter that will be spawned at the tip of the weapon for cool down after the firing
- ChameleonSecondary
    - SAmmoPerFire
        - The amount of ammo used per weapon fire
    - SFireRate
        - The rate in seconds after one fire before another can happen
    - SFireSound
        - The sound that happens when the weapon is fired
    - SFlashEmitter
        - The emitter that is spawned when the weapon is fired
    - SProjSpawnOffset
        - The offset based on the clients local space that the projectile and the FlashEmitter are spawned.
        - X is to the left
        - Y is up
        - Z is toward the clients view
    - SShakeOffsetMag
        - This is the magnitude in units that the screen shake will randomly move in when fired. XYZ as above.
    - SShakeOffsetRate
        - This is how rapidly it will move there in units per second when fired. XYZ as above.

- o SShakeOffsetTime
  - This is the rate at which the view will recover on it own
- o SShakeRotMag
  - This is the magnitude of the rotation that the screen shake will randomly move in when fired. XYZ as above.
- o SShakeRotRate
  - This is how rapidly it will move rotate when fired. XYZ as above.
- o SShakeRotTime
  - This is the rate at which the view will recover on it own
- o SmokeEmitter
  - This is the emitter that will be spawned at the tip of the weapon for cool down after the firing

BUILDING CONTENT

Making new content for the Chameleon weapon requires that you keep track of a few things.

First, all bone names must remain as they are. There is code that works with those bone names so changing them will break things. Bones of special note will be noted under the various items below.

Second, the original orientation of the object in the base scene file is what you should work with when exporting the items. You can model them in whatever orientation you want. But before export they have to be moved to the location and orientation of the sample files.

Third, the joint orientation on the end joints are important as they dictate which direction the muzzleflash goes. The important information regarding joint direction is listed below.

Finally, be sure you understand how to set up your custom sets and shading groups so that the exporter will work with them.

- ChameleonFirstPerson
  - o 16, 48, 96
  - o Bone_Weapon – This is the base joint of the skeleton.
  - o bone_flashA – This is the joint where the primary weapon fire muzzleflash will appear at. Positive Z is the direction the muzzleflash will go.
  - o Bone_FlashB – This is the joint where secondary weapon fire muzzleflash will appear at. Positive Z is the direction the muzzleflash will go.
- ChameleonThirdPerson
  - o 16, 48, 96
  - o bone_weapon – This is the base joint of the skeleton.
  - o bone_flash – This is the joint where muzzleflashes will appear on the third person model.
- ChameleonPickup
  - o 16, 48, 96
- ChameleonAmmoPickup
  - o 16, 48, 16
- ChameleonProj
  - o 8, 8, 16
- ChameleonProjSecondary
  - o 8, 8, 16
- ChameleonWeaponSkin
- ChameleonAmmoPickupSkin
- ChameleonProjSkin
- ChameleonProjSecondarySkin

NAMING CONTENT

Content should be named as it is already named in the content tree provided. This is critically important for the skeletal meshes and static meshes. They're names are referenced by the Chameleon.u code and if they are changed at all it will break.

Also, groups should never be used with the Chameleon content as this will break the functionality as well.

QUIZ 1

1. List several functionalities of a level editor.

2. If you know how to use one level editor will you know how to use them all?

3. What is the function of the Animation Browser?

4. How should you think about the world when building areas in UnrealEd?

5. What is a package?

6. What types of things are stored in packages?

7. What is an actor?

8. How does class inheritance work?

9. If you create a new instance of the class Pickup, where will that class be placed in the class hierarchy?

10. What are some reasons you would want to create your own actors?

QUIZ 2

1. Why is understanding who makes decisions on a team important?

2. Are a given team members responsibilities and decision making authority hard fast unbreakable rules at all times of the production? Explain.

3. Why is the term level designer still a rather ambiguously defined one in the industry?

4. What is the most reliable way to gauge and monitor performance?

5. Why is it important to define performance budgets before entering production?

6. What is the command to bring up the graphics performance numbers?

7. What is BSP's primary usage in level design in Unreal?

8. What types of lights are available in Unreal?

9. How do shadows work on BSP?

10. How do shadows work on static meshes?

QUIZ 3

1. How do you visualize the zones in the perspective view?

2. What is the purpose of anti portals?

3. How do they differ from zone portals?

4. What part of UnrealEd allows you to import textures?

5. Name at least two systems in UnrealEd not covered in this course.

6. What are some ways that assets are produced?

7. Why is a smooth production pipeline essential to a successful production?

8. Why is a clean Content Tree important for a smooth game production cycle?

9. Along those line why is it important that everyone on the team understand the layout of the content tree?

10. Why are conventions important to a smooth production?

QUIZ 1 ANSWERS

1. Building areas, making complex systems for the engine, creating functionality by wiring objects together. Count this answer as correct if they come up with more than 3 things that a level editor could be used for in games production.
2. No.
3. The animation browser is for editing and previewing skeletal meshes.
4. You should think about the world in UnrealEd as a large chunk of clay or solid object that you carve areas out of.
5. A package is a collection of assets.
6. Textures, skeletal meshes, static meshes, sounds, music, actors, vertex animated meshes.
7. An actor is the parent class of all functional objects that are placeable in the Unreal engine.
8. Class inheritance works by passing functionality down to the children of a given object. In other words, all children have the functionality of their parents all the way up the hierarchy.
9. Underneath the pickup class.
10. Creating your own custom functionality based off of the parents functionality. Having versions with custom settings for easy placement and consistency sake.

QUIZ 2 ANSWERS

1. If no one is in charge of making decisions or the role of decision making is poorly defined then problems can erupt when either there are clashes of opinion and no one has the clear cut authority to make executive calls or decisions just plain don't get made.
2. No. During production crunches for milestones and deliverables it is acceptable and frequently the case that there is a lot of role overlap to make sure that everything that needs to get done does.
3. Primarily because each project has its own set of requirement for how the game and levels are going to be authored. So each one is different.
4. Milliseconds.
5. If you don't know what you're performance budgets are then chances are you're going to exceed them and have a game that doesn't perform up to snuff.
6. stat render.
7. Defining bulk areas. Not detail work.
8. Point and spotlight. There are many types of effects but they are all subsets of point and spotlights.
9. Shadowmapping.
10. Vertex lighting.

QUIZ 3 ANSWERS

1. Clicking the Zone View button in that views toolbar.
2. To cull objects behind them from being rendered.
3. Zone portals define zones, anti portals block visibility.
4. The Texture Browser.
5. Any of the following: Material System, Emitters, Terrain, Matinee.
6. 3D packages such as Maya or 3DSMax. 2D packages such as Photoshop.
7. The smoother the production pipeline the easier it is to get content into the engine and then subsequently iterate on it for quality's sake.
8. Hundreds if not thousands of assets will be created. A clean, well thought out content tree ensures that all of those assets are well organized and easy to find later on.
9. In order for the content tree to stay clean and well organized everyone has to know how to use it.
10. Conventions are important to a project because they ensure that everything is going to be produced and stored in a consistent fashion that allows working with the content later to happen trouble free.

APPENDIX: WEAPON CLASS BREAKDOWN

WEAPON PICKUP

- o Functionality
  - Pickup Mesh
  - Pickup Sound
- o Asset
  - Weapon Pickup Static Mesh
  - Weapon Pickup Texture

AMMO PICKUP

- o Functionality
  - Ammo Mesh
  - Pickup Ammo Amount
  - Max Ammo
- o Assets
  - Ammo Pickup Static Mesh
  - Ammo Pickup Texture

PLAYER WEAPON

- o Weapon Functionality
  - Weapon Mesh
- o Primary Fire Functionality
  - Ammo Class
  - FullAuto - boolean
  - Kick
  - Rate of Fire
  - Projectile Offset
  - Projectile Velocity
  - Projectile Class
  - Fire FX Class
  - Fire FX Position
  - Fire Sound
  - Dry Fire Sound
- o Secondary Fire Functionality
  - Ammo Class
  - Rate of Fire
  - Kick
  - Projectile Offset
  - Projectile Velocity
  - Projectile class
  - Fire FX Class
  - Fire FX Position
  - Fire Sound
  - Dry Fire Sound
- o Assets
  - Player Model Static Mesh
  - Player Model Texture
  - UI Picture Texture
  - Primary Fire FX
  - Alt Fire FX

PROJECTILE

- o Functionality
  - • Projectile Mesh
  - • Travelling FX Class
  - • Travelling FX Position
  - • Impact FX Class
- o Assets
  - • Projectile Static Mesh
  - • Projectile Texture
  - • Travelling FX
  - • Impact FX