IMGD 1001 - The Game Development Process: 3D Art

by

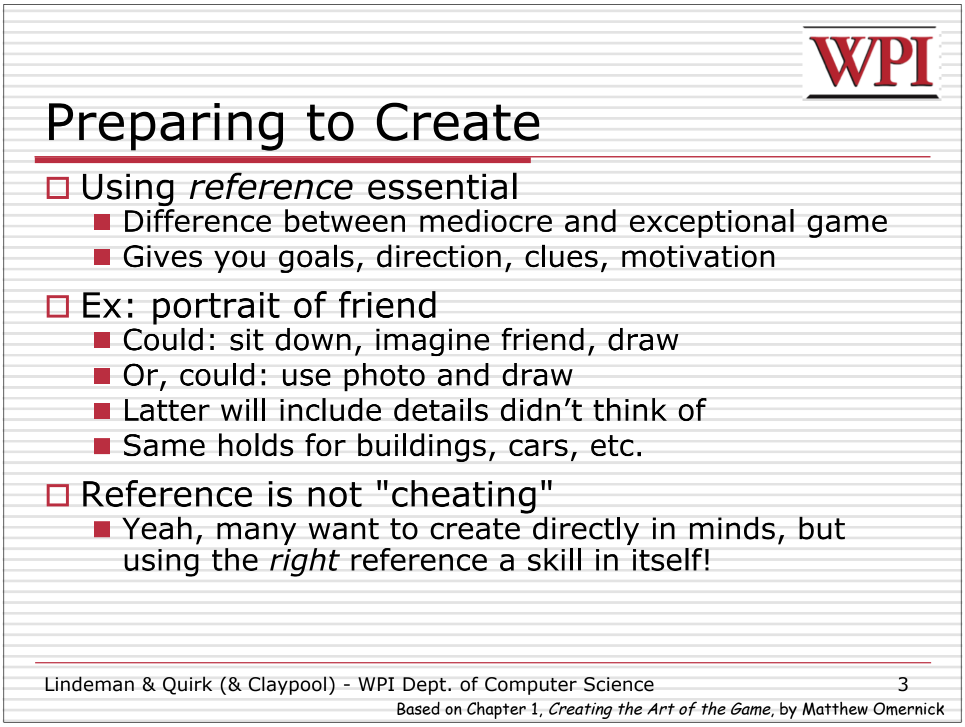
Robert W. Lindeman (gogo@wpi.edu)

Kent Quirk (kent_quirk@cognitoy.com)

(with lots of input from Mark Claypool!)

3D Art: Outline

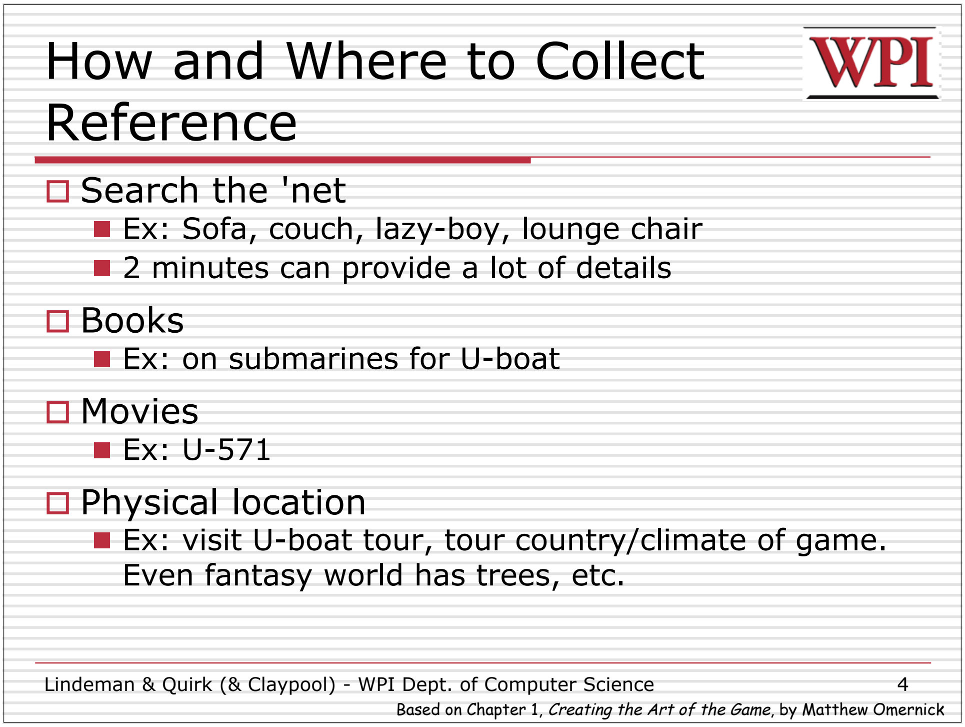
- □ Preparing to Create
- □ Modeling Theory
  - Example
- □ Texturing
- □ Lighting



Preparing to Create

- Using *reference* essential
  - Difference between mediocre and exceptional game
  - Gives you goals, direction, clues, motivation
- Ex: portrait of friend
  - Could: sit down, imagine friend, draw
  - Or, could: use photo and draw
  - Latter will include details didn’t think of
  - Same holds for buildings, cars, etc.
- Reference is not "cheating"
  - Yeah, many want to create directly in minds, but using the *right* reference a skill in itself!

Based on Chapter 1, *Creating the Art of the Game*, by Matthew Omernick



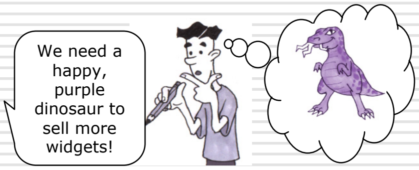
How and Where to Collect Reference

- Search the 'net
  - Ex: Sofa, couch, lazy-boy, lounge chair
  - 2 minutes can provide a lot of details
- Books
  - Ex: on submarines for U-boat
- Movies
  - Ex: U-571
- Physical location
  - Ex: visit U-boat tour, tour country/climate of game. Even fantasy world has trees, etc.

Based on Chapter 1, *Creating the Art of the Game*, by Matthew Omernick

Concept Art

- □ Pre-visualize art for communication, color, inspiration
  - Establish "look and feel", like storyboarding for film
  - Saves time and money since iterate before rendering
- □ Even if company has concept artists, digital artists should still do their own
  - Remember, computer is just another tool
  - Figure drawing helps understand shape and line
    - □ And often required for portfolio!



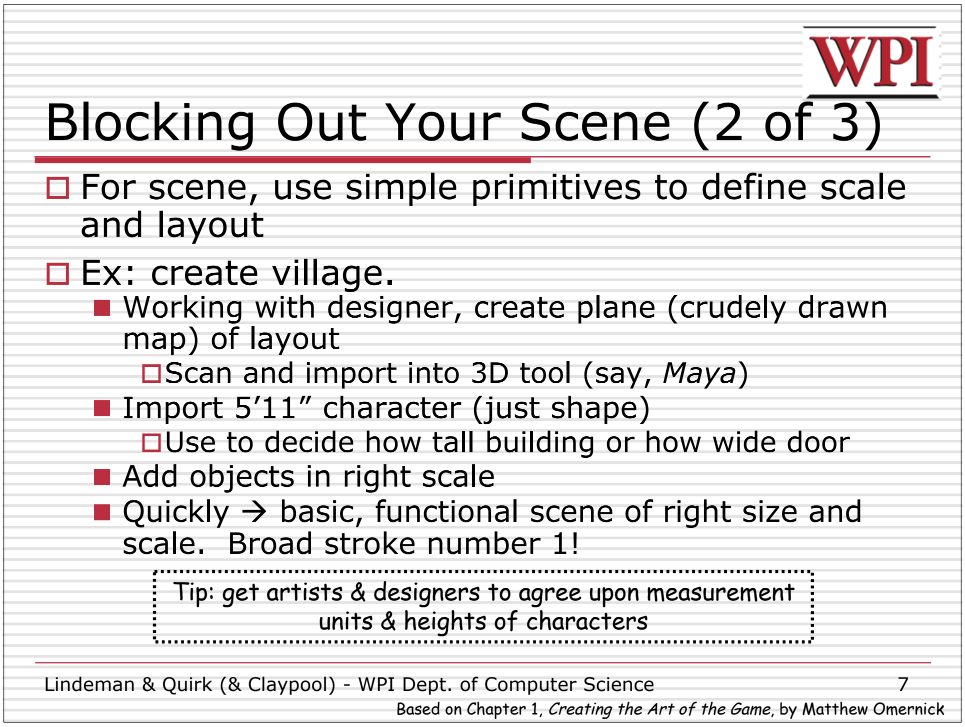
- Paolo Piselli

Based on Chapter 1, *Creating the Art of the Game*, by Matthew Omernick

Blocking Out Your Scene (1 of 3)

- □ Mantra
  - "Broad strokes, then add detail"
- □ Ex: painting mountain scene.
  - Start with blue sky, define brown mountains, lake. Finer brush for trees, clouds. Finer for rocks, birds in sky …
  - Start in corner. Paint all details and move over. You'd go crazy! Would be skewed.
- □ Ex: animating a character.
  - Set two keyframes, point A and point B. Get speed right, basic idea. Add frames for up and down. Then legs and arms swing …
  - When done, smooth walk

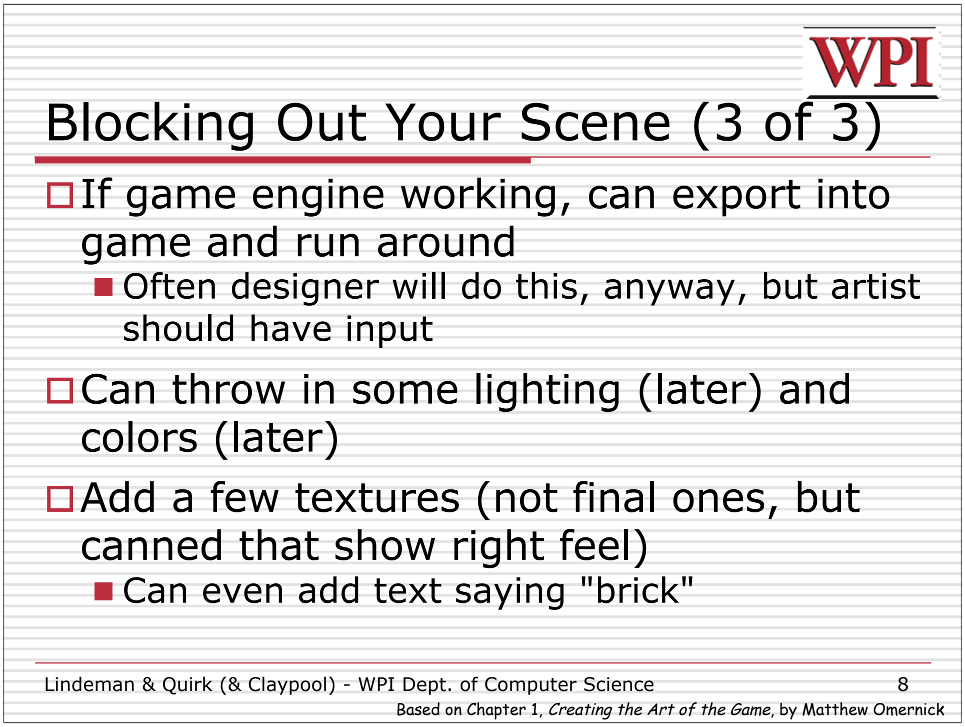
Based on Chapter 1, *Creating the Art of the Game*, by Matthew Omernick



Blocking Out Your Scene (2 of 3)

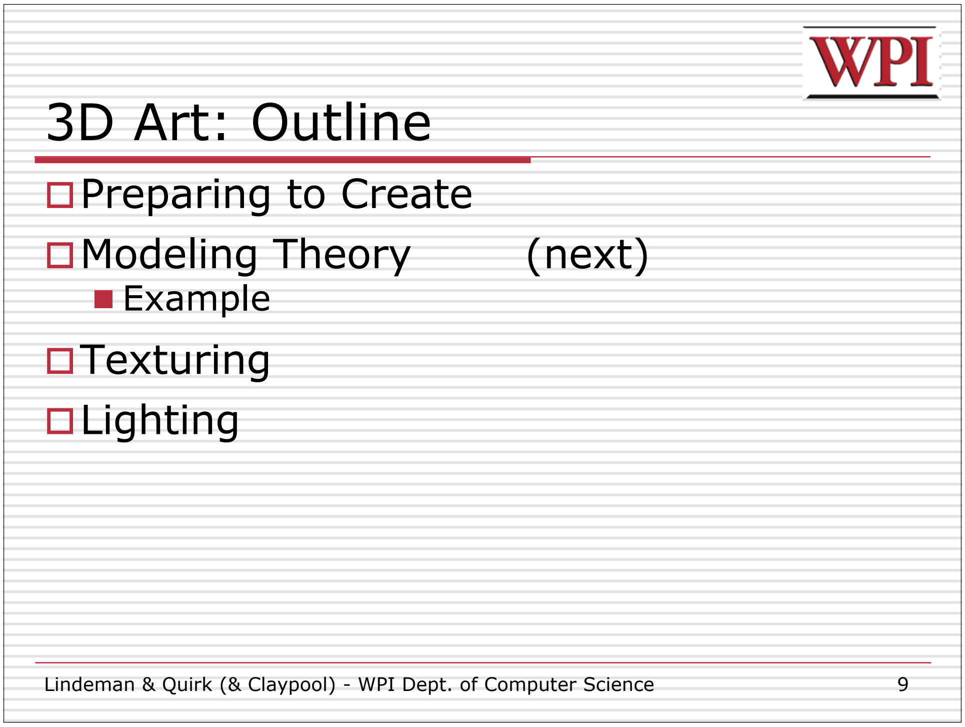
- □ For scene, use simple primitives to define scale and layout
- □ Ex: create village.
  - ■ Working with designer, create plane (crudely drawn map) of layout
    - □Scan and import into 3D tool (say, *Maya*)
  - ■ Import 5’11” character (just shape)
    - □Use to decide how tall building or how wide door
  - ■ Add objects in right scale
  - ■ Quickly → basic, functional scene of right size and scale. Broad stroke number 1!

Tip: get artists & designers to agree upon measurement units & heights of characters



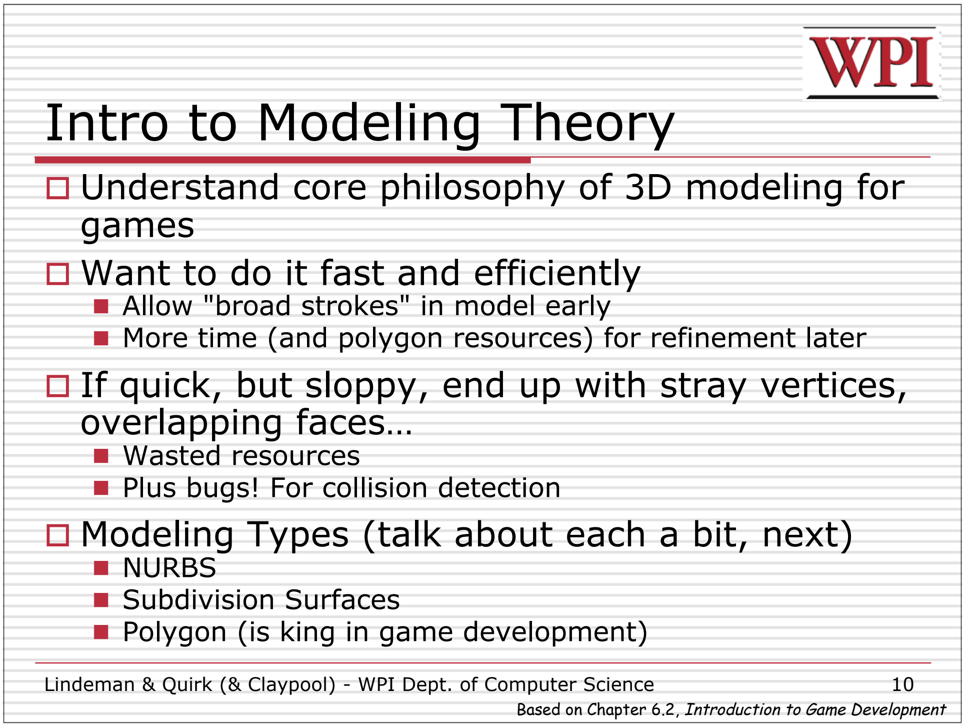
Blocking Out Your Scene (3 of 3)

- □If game engine working, can export into game and run around
  - ■ Often designer will do this, anyway, but artist should have input
- □Can throw in some lighting (later) and colors (later)
- □Add a few textures (not final ones, but canned that show right feel)
  - ■ Can even add text saying "brick"



3D Art: Outline

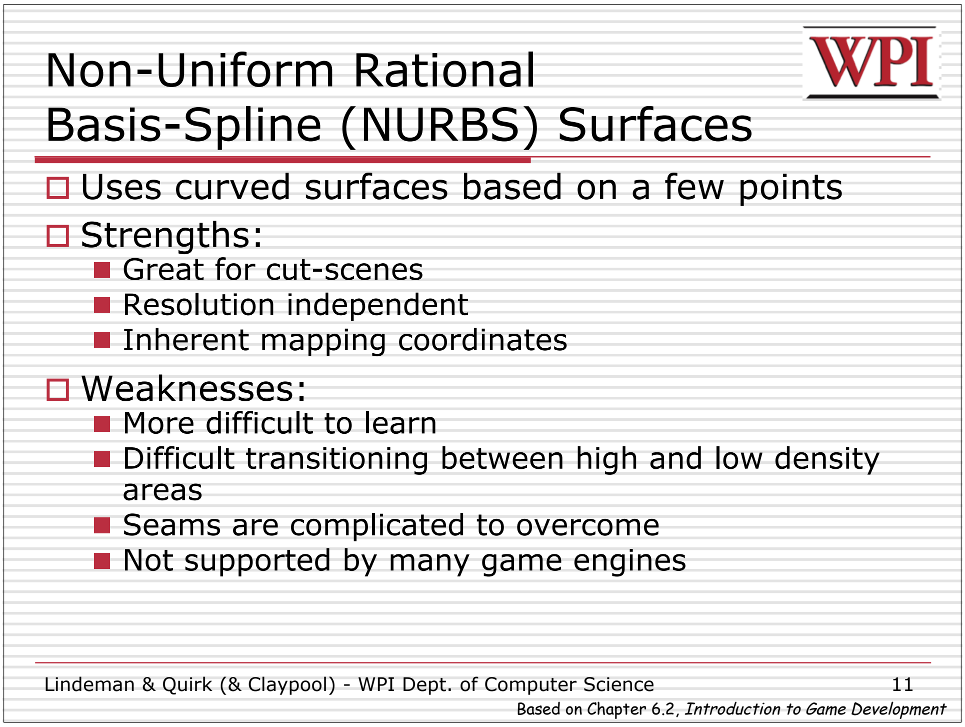
- Preparing to Create
- Modeling Theory (next)
  - Example
- Texturing
- Lighting



Intro to Modeling Theory

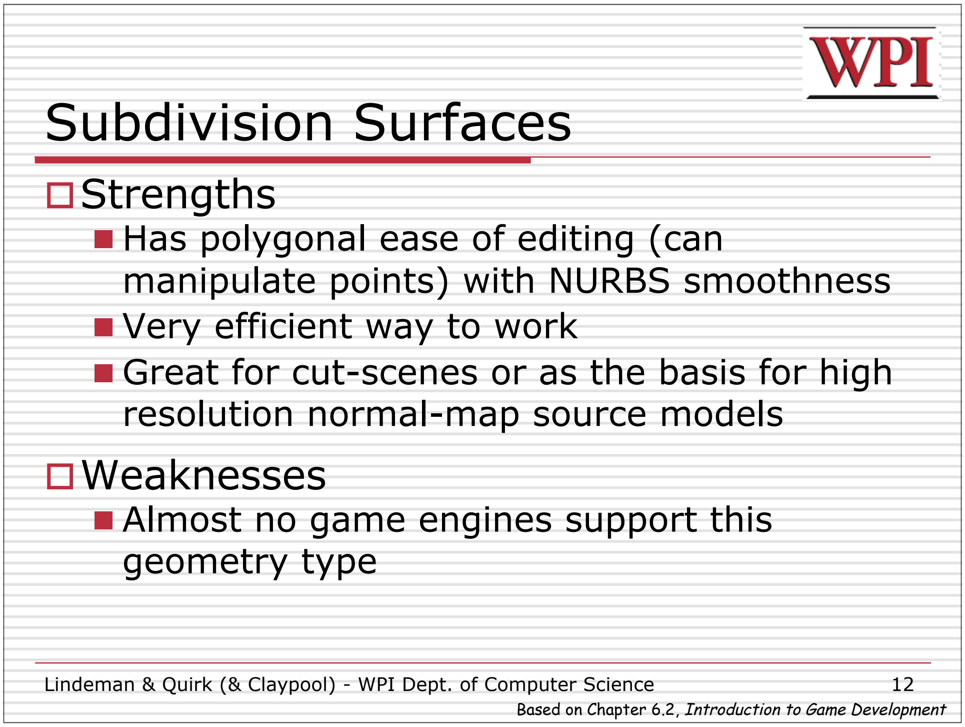
- Understand core philosophy of 3D modeling for games
- Want to do it fast and efficiently
  - Allow "broad strokes" in model early
  - More time (and polygon resources) for refinement later
- If quick, but sloppy, end up with stray vertices, overlapping faces…
  - Wasted resources
  - Plus bugs! For collision detection
- Modeling Types (talk about each a bit, next)
  - NURBS
  - Subdivision Surfaces
  - Polygon (is king in game development)

Based on Chapter 6.2, *Introduction to Game Development*



Non-Uniform Rational Basis-Spline (NURBS) Surfaces

- Uses curved surfaces based on a few points
- Strengths:
  - Great for cut-scenes
  - Resolution independent
  - Inherent mapping coordinates
- Weaknesses:
  - More difficult to learn
  - Difficult transitioning between high and low density areas
  - Seams are complicated to overcome
  - Not supported by many game engines



Subdivision Surfaces

- Strengths
  - Has polygonal ease of editing (can manipulate points) with NURBS smoothness
  - Very efficient way to work
  - Great for cut-scenes or as the basis for high resolution normal-map source models
- Weaknesses
  - Almost no game engines support this geometry type

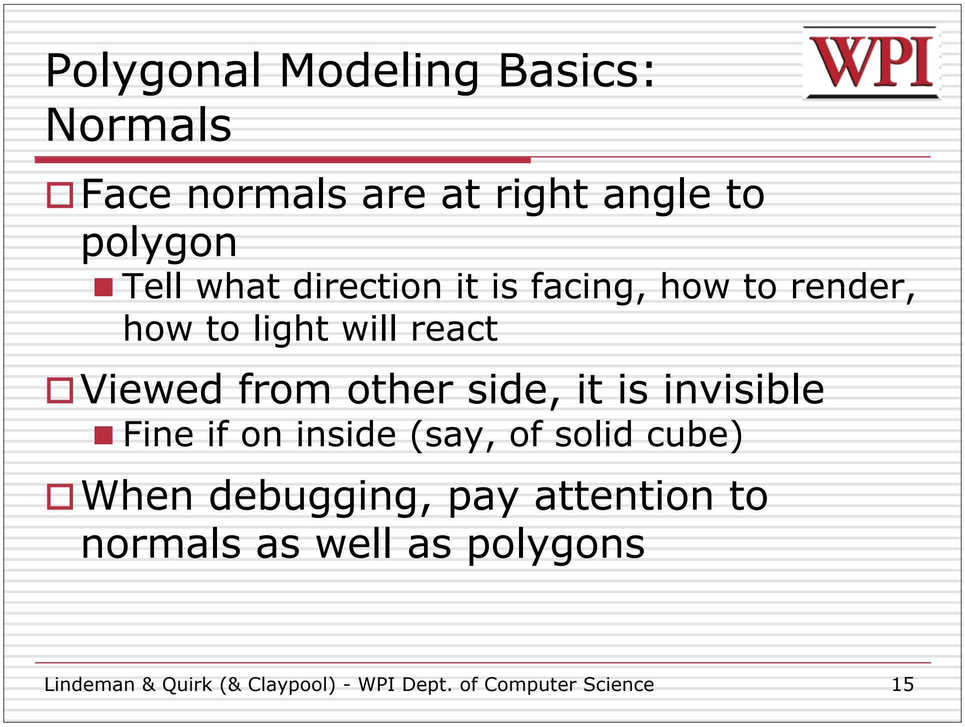
Polygonal

- Strengths:
  - Very straightforward, easy to troubleshoot, easy to modify
  - Supported by all 3D game engines
- Weaknesses:
  - A technical process
    - Constantly manipulating topology
  - Faceting
    - Rough around the edges
  - Fixed Resolution
    - Unless level of detail models are created
- Polygons preferred since most used (talked about rest of section)
  - By polygons, we mean *triangles*
  - Face may have triangles that share vertices (Ex: square down middle)
  - Software may hide shared edge for cleaner look

Based on Chapter 6.2, *Introduction to Game Development*

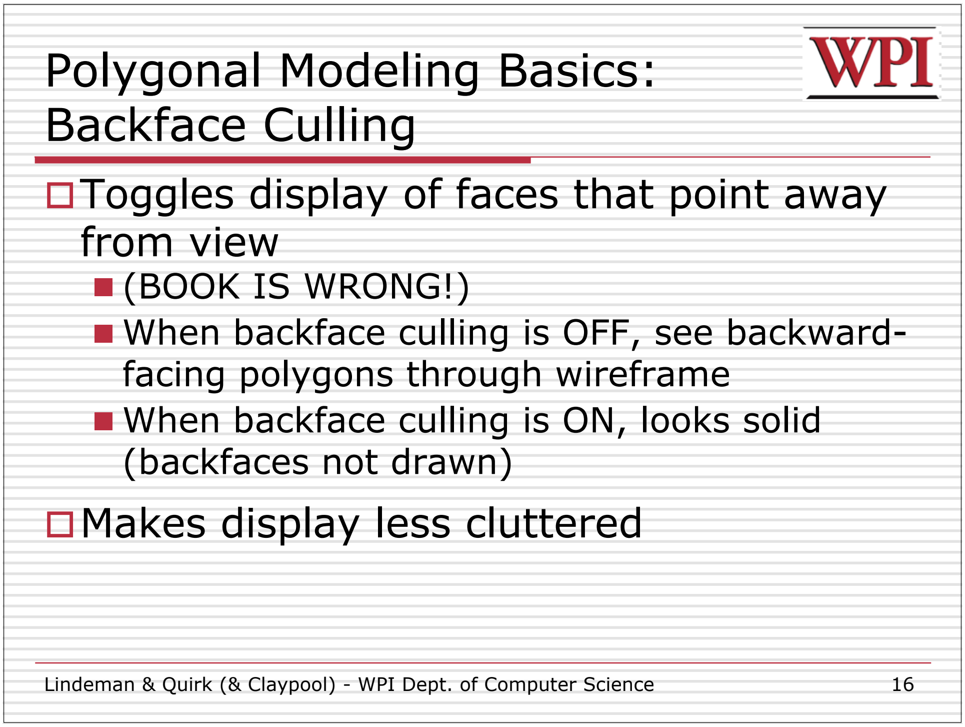
Polygonal Modeling Basics: Primitives

- *Primitives* are basic shapes
- Most 3D packages have same primitives:
  - Sphere, Cube, Cylinder, Plane
  - Use for "broad strokes"
- Concentrate on primitives within object
  - Ex: human body (ovals for shoulders, cylinders for legs, sphere for head…)
- *Components* are parts that make up primitive
  - Ex: vertices, edges, triangles, faces, elements
  - Similar across all packages, but terminology can vary
- *Transformation* allows moving, rotating, scaling object or component



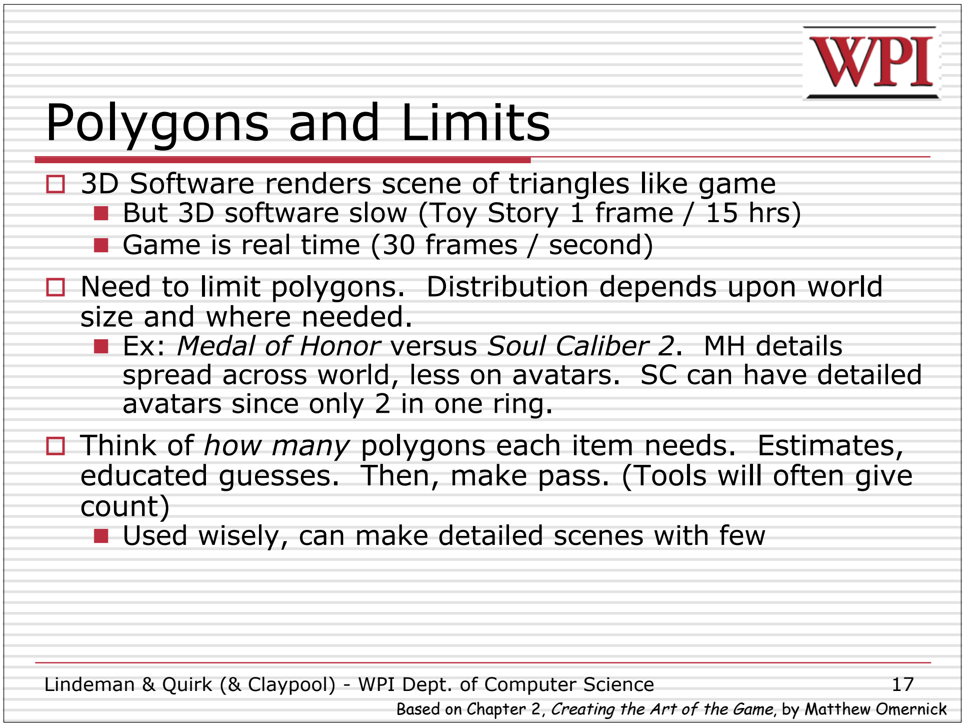
Polygonal Modeling Basics: Normals

- Face normals are at right angle to polygon
  - Tell what direction it is facing, how to render, how to light will react
- Viewed from other side, it is invisible
  - Fine if on inside (say, of solid cube)
- When debugging, pay attention to normals as well as polygons



Polygonal Modeling Basics: Backface Culling

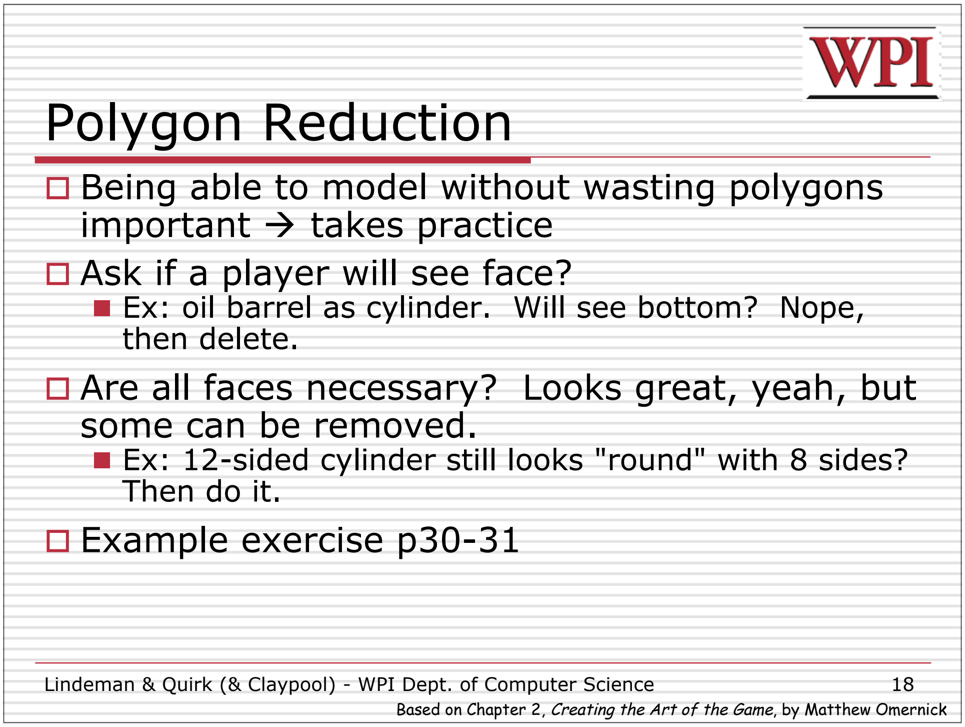
- Toggles display of faces that point away from view
  - (BOOK IS WRONG!)
  - When backface culling is OFF, see backward-facing polygons through wireframe
  - When backface culling is ON, looks solid (backfaces not drawn)
- Makes display less cluttered



Polygons and Limits

- □ 3D Software renders scene of triangles like game
  - But 3D software slow (Toy Story 1 frame / 15 hrs)
  - Game is real time (30 frames / second)
- □ Need to limit polygons. Distribution depends upon world size and where needed.
  - Ex: *Medal of Honor* versus *Soul Caliber 2*. MH details spread across world, less on avatars. SC can have detailed avatars since only 2 in one ring.
- □ Think of *how many* polygons each item needs. Estimates, educated guesses. Then, make pass. (Tools will often give count)
  - Used wisely, can make detailed scenes with few

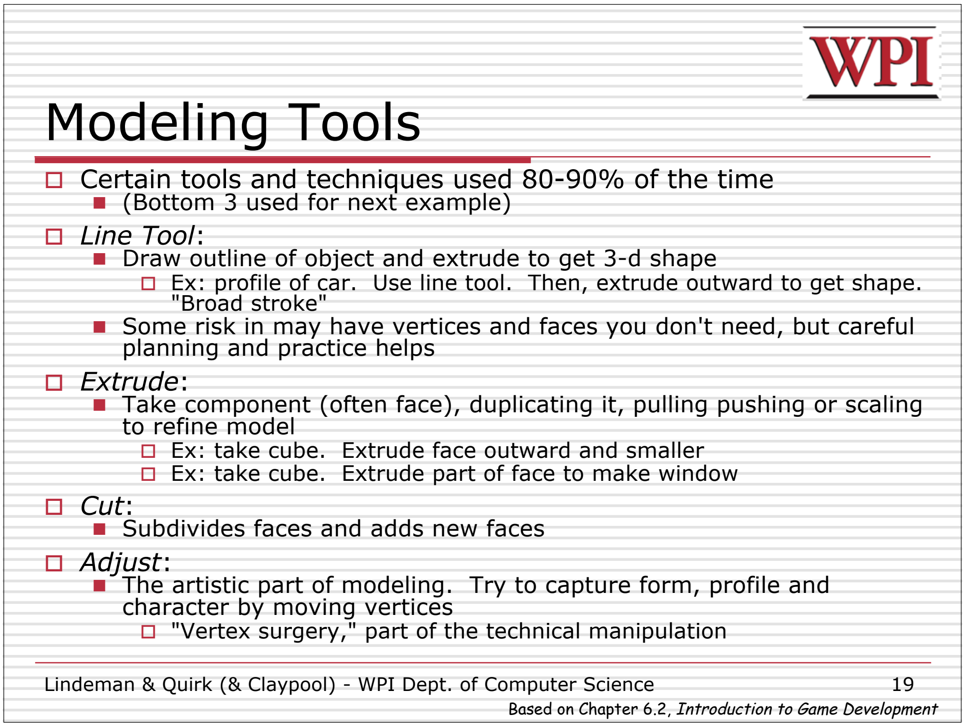
Based on Chapter 2, *Creating the Art of the Game*, by Matthew Omernick



Polygon Reduction

- □ Being able to model without wasting polygons important → takes practice
- □ Ask if a player will see face?
  - Ex: oil barrel as cylinder. Will see bottom? Nope, then delete.
- □ Are all faces necessary? Looks great, yeah, but some can be removed.
  - Ex: 12-sided cylinder still looks "round" with 8 sides? Then do it.
- □ Example exercise p30-31

Based on Chapter 2, *Creating the Art of the Game*, by Matthew Omernick

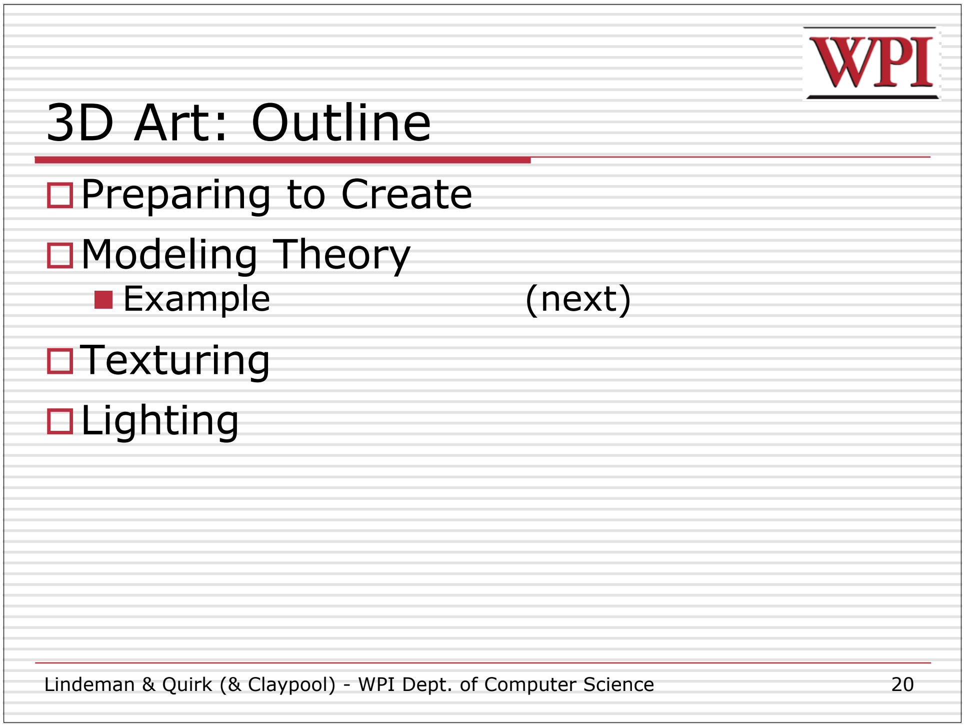


Modeling Tools

- Certain tools and techniques used 80-90% of the time
  - (Bottom 3 used for next example)
- *Line Tool*:
  - Draw outline of object and extrude to get 3-d shape
    - Ex: profile of car. Use line tool. Then, extrude outward to get shape. "Broad stroke"
  - Some risk in may have vertices and faces you don't need, but careful planning and practice helps
- *Extrude*:
  - Take component (often face), duplicating it, pulling pushing or scaling to refine model
    - Ex: take cube. Extrude face outward and smaller
    - Ex: take cube. Extrude part of face to make window
- *Cut*:
  - Subdivides faces and adds new faces
- *Adjust*:
  - The artistic part of modeling. Try to capture form, profile and character by moving vertices
    - "Vertex surgery," part of the technical manipulation

Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 19

Based on Chapter 6.2, *Introduction to Game Development*



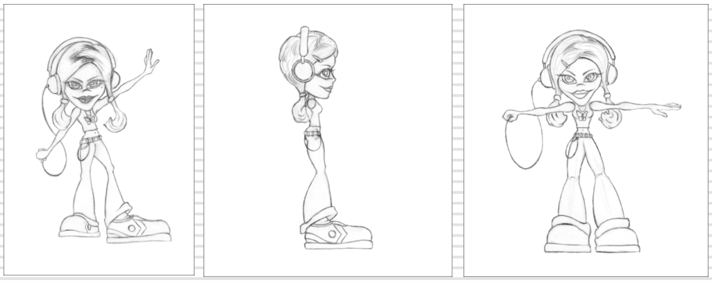
3D Art: Outline

- Preparing to Create
- Modeling Theory
  - Example (next)
- Texturing
- Lighting

Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 20

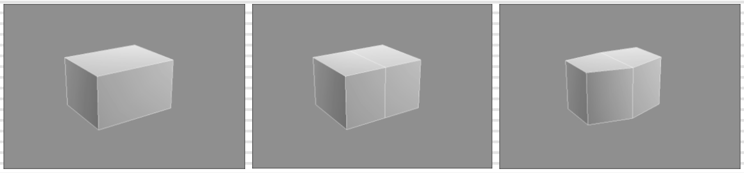
Box Modeling: Reference

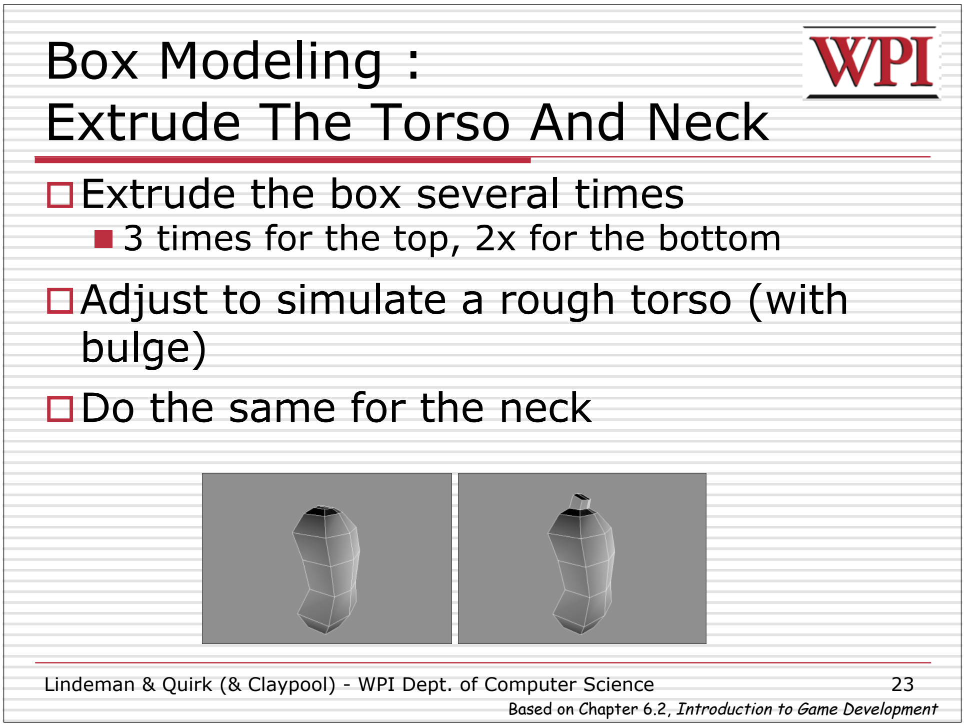
- □ Decide on polygon limits
- □ Posed and turnaround sketches of a character
  - ■ Can often be imported into 3D tool



Box Modeling: Start With A Box

- □ Begin with a box
- □ Cut it in half
- □ Approximate the torso shape
- □ Cut it in half (will do half well, then mirror)





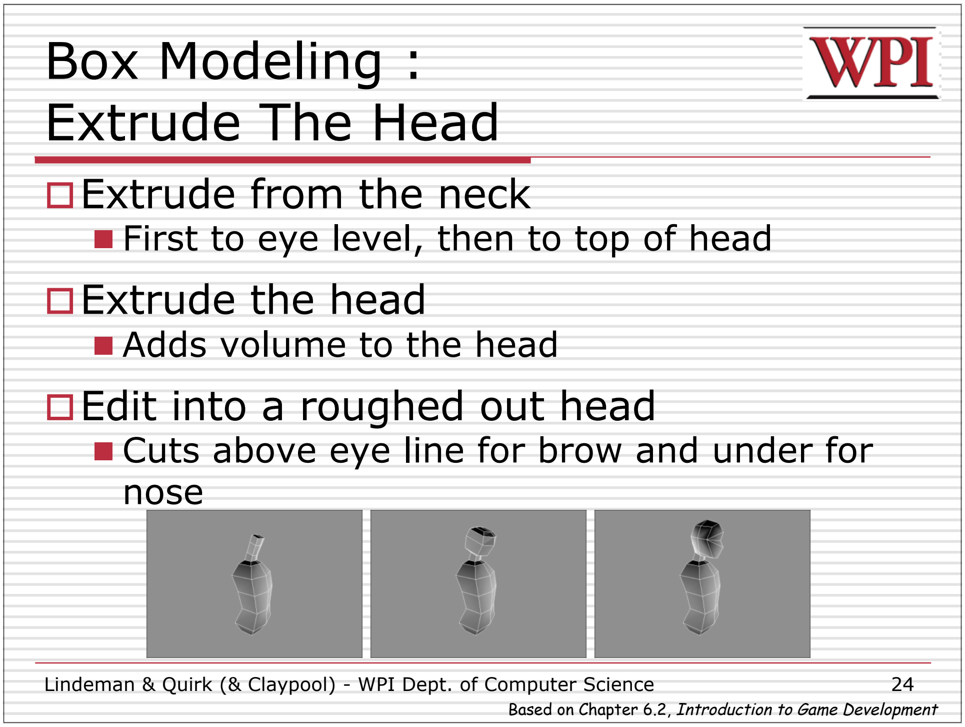
Box Modeling : Extrude The Torso And Neck

WPI

- Extrude the box several times
  - 3 times for the top, 2x for the bottom
- Adjust to simulate a rough torso (with bulge)
- Do the same for the neck

Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 23

Based on Chapter 6.2, *Introduction to Game Development*



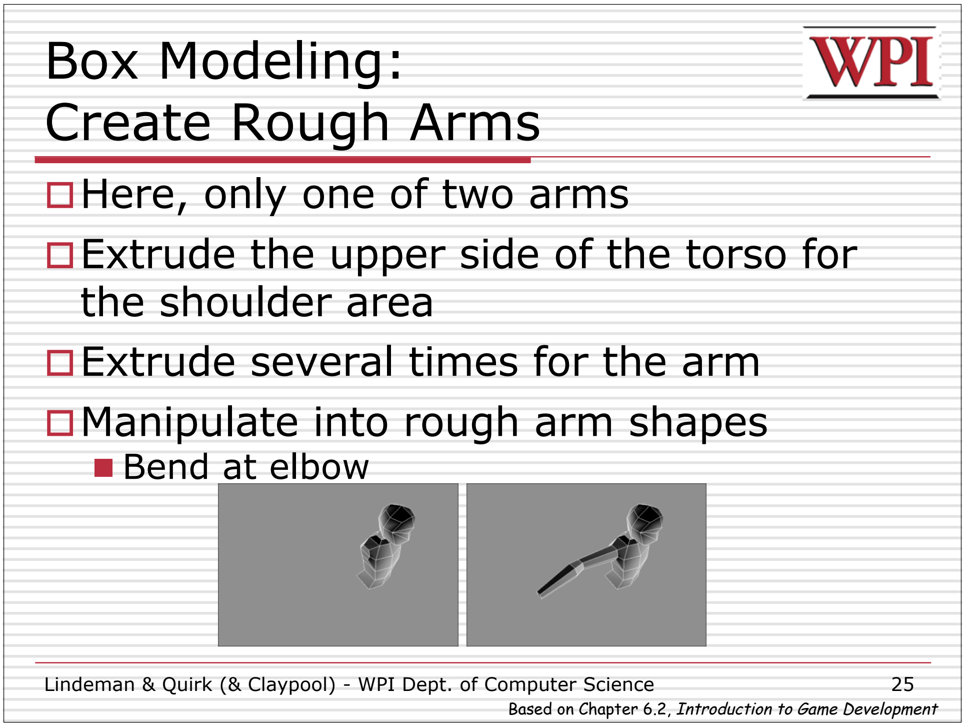
Box Modeling : Extrude The Head

WPI

- Extrude from the neck
  - First to eye level, then to top of head
- Extrude the head
  - Adds volume to the head
- Edit into a roughed out head
  - Cuts above eye line for brow and under for nose

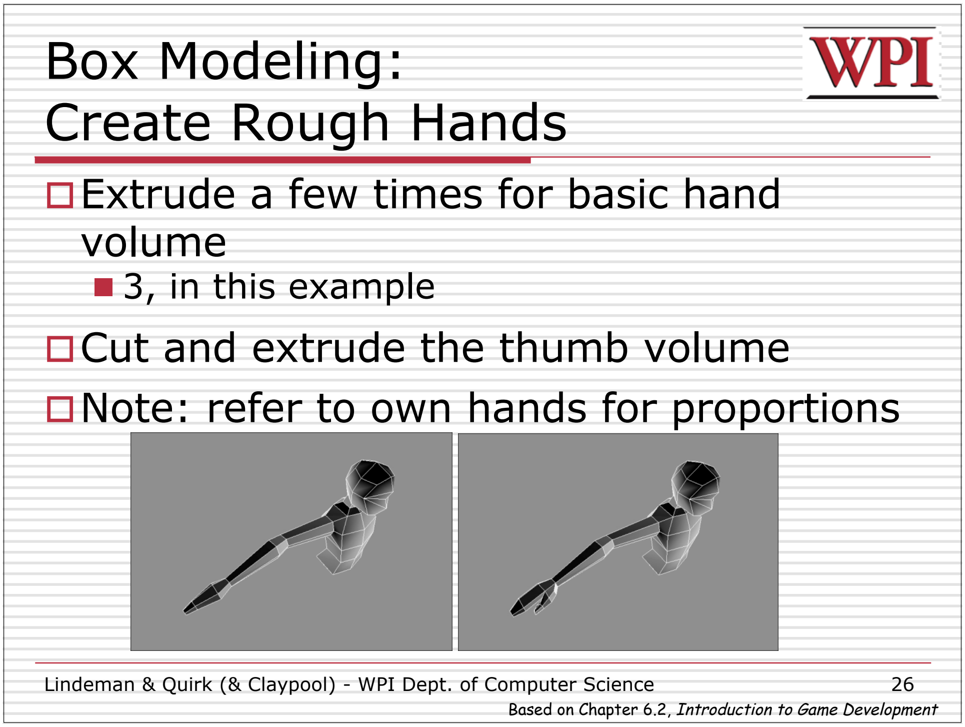
Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 24

Based on Chapter 6.2, *Introduction to Game Development*



Box Modeling: Create Rough Arms

- □ Here, only one of two arms
- □ Extrude the upper side of the torso for the shoulder area
- □ Extrude several times for the arm
- □ Manipulate into rough arm shapes
  - ■ Bend at elbow

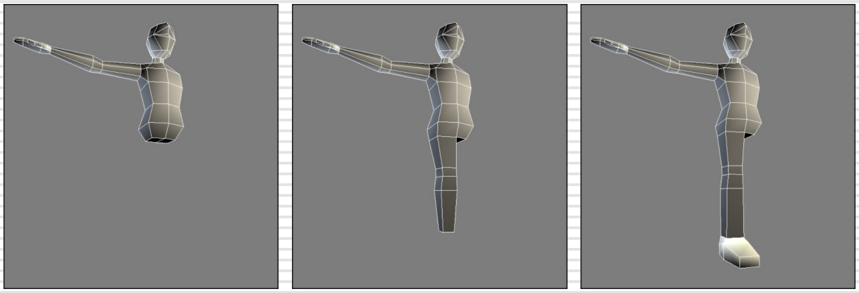


Box Modeling: Create Rough Hands

- □ Extrude a few times for basic hand volume
  - ■ 3, in this example
- □ Cut and extrude the thumb volume
- □ Note: refer to own hands for proportions

Box Modeling: Time For Legs

- □ Again, create only one of the legs
- □ Extrude and edit
- □ Extrude feet forward from stump



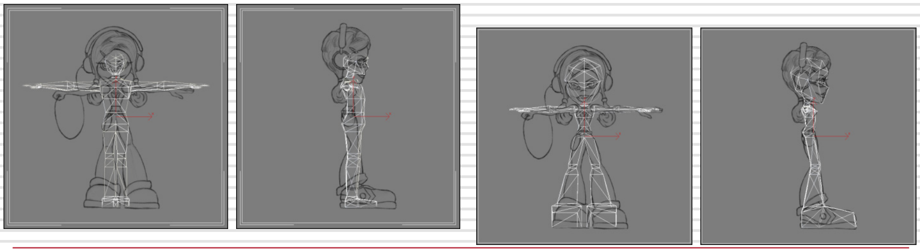
Box Modeling: Mirror

- □ Delete half of the model
- □ Mirror the other side
- □ Attach and weld the seam

Next up, refining the model!

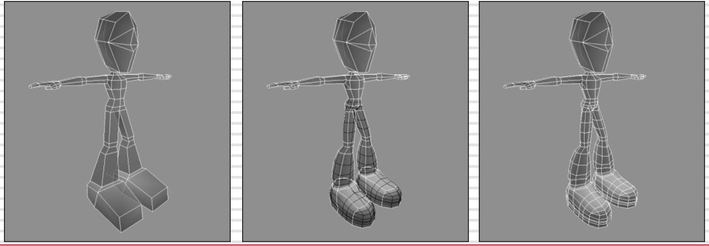
Box Modeling: Proportions Match

- □ Bring the turnaround sketches into the viewport on a textured plane
- □ Manipulate until the model matches up
  - Important things: head right size, extremity lengths, eye level.



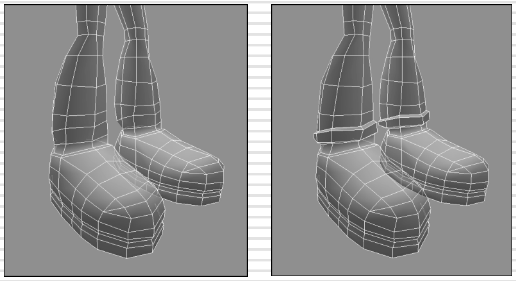
Box Modeling : Begin Adding Detail

- □ Square legs and shoes are especially prominent
  - Will look blocky in game engine
- □ Add a few more segments to support more curvature for the legs



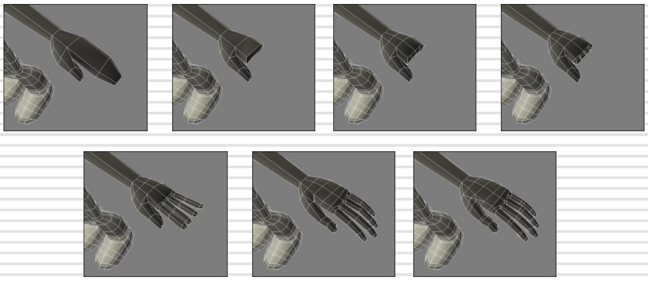
Box Modeling : Cuffs

- □ Cut faces into feet to get curvature, adjust as necessary
  - Constantly compare to sketches
- □ A simple extrude to create the cuffs of the pants



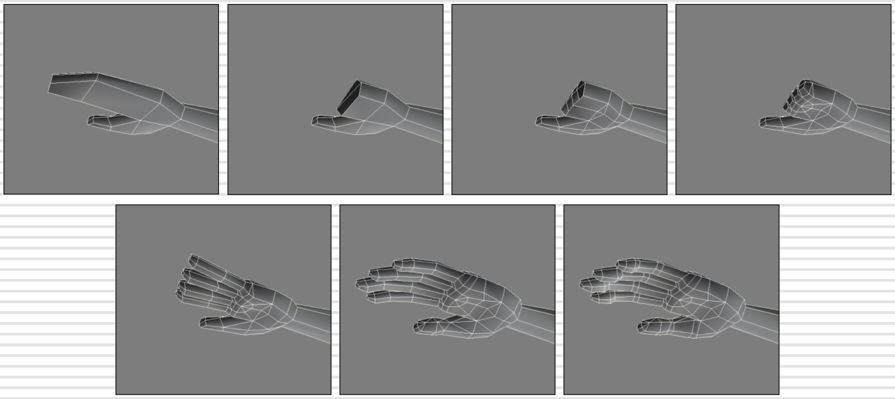
Box Modeling: Gimme A Hand (1 of 2)

- □ Often the most difficult
  - Use own hand constantly for proportions
    - □ Slightly curved, so natural, middle finger higher
- □ Cut where fingers begin
- □ Extrude outward for 4 fingers
- □ Fingers will need joints if animated



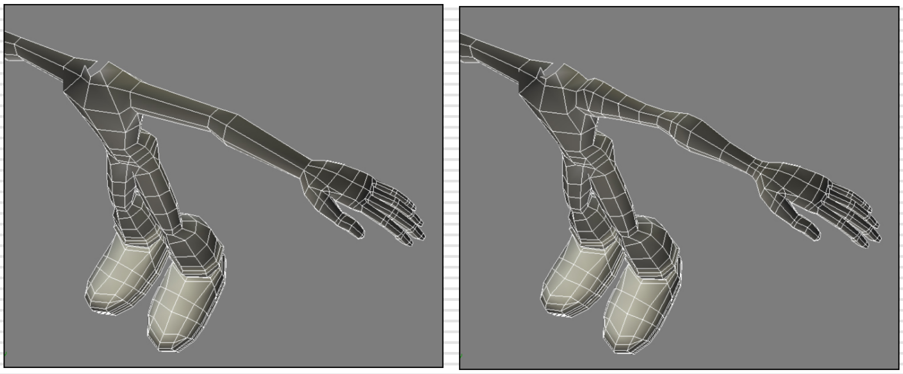
Box Modeling : Gimme A Hand (2 of 2)

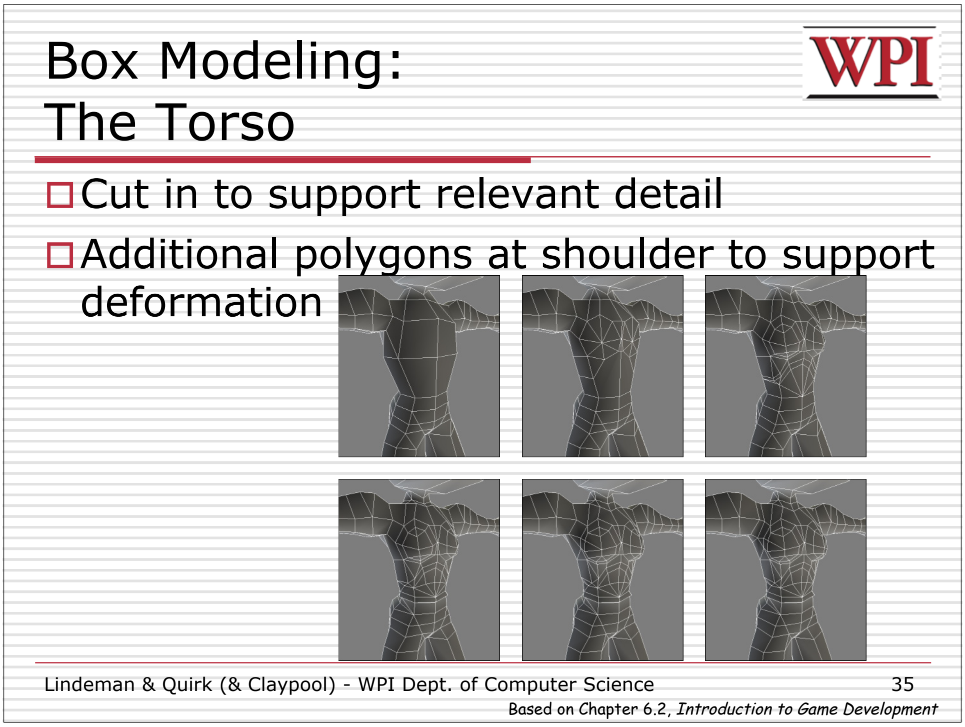
- Same buildup, but from underneath



Box Modeling: More Arm Please

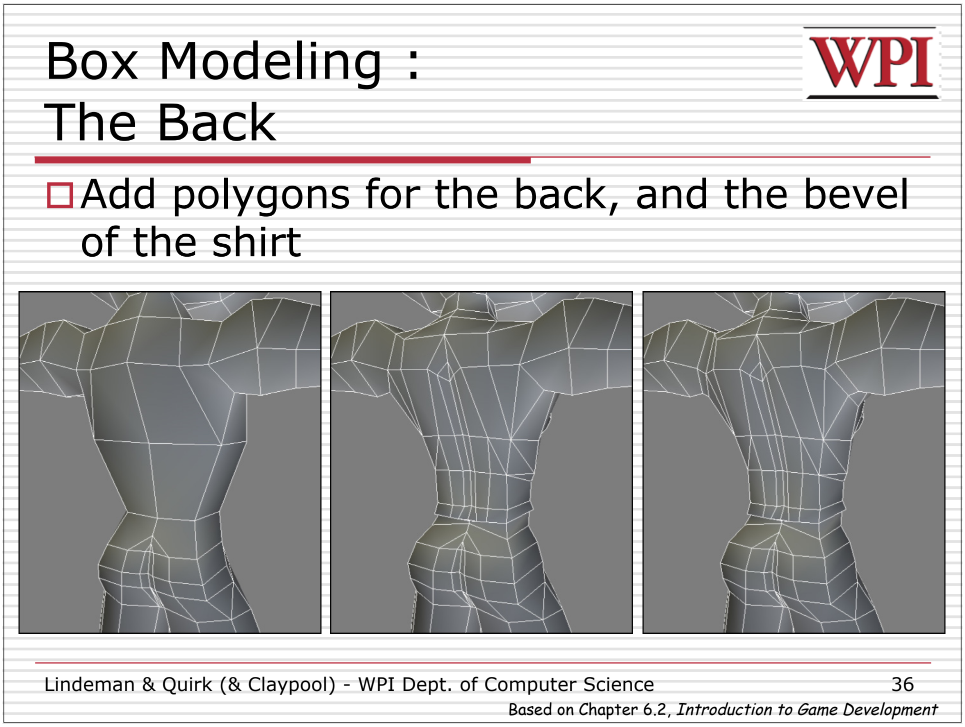
- Cut in more detail for the arm, and manipulate for better form and curvature





Box Modeling: The Torso

- □ Cut in to support relevant detail
- □ Additional polygons at shoulder to support deformation

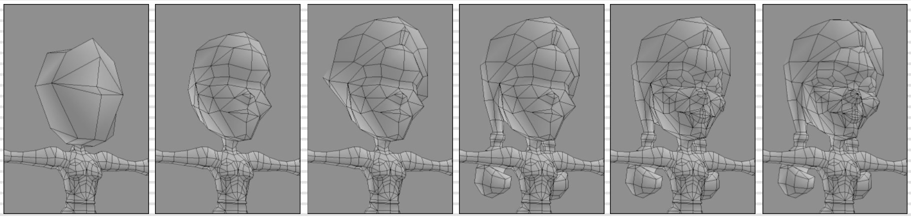


Box Modeling : The Back

- □ Add polygons for the back, and the bevel of the shirt

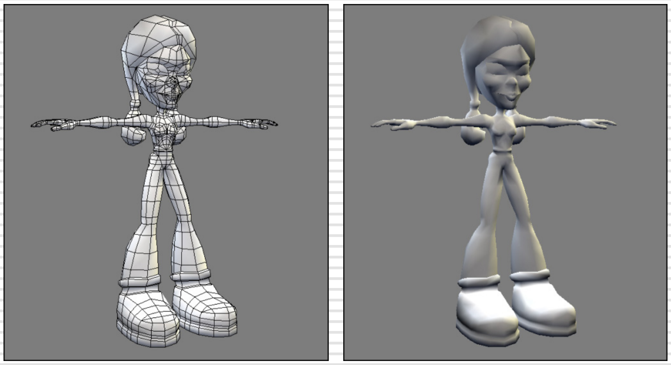
Box Modeling : The Face, Head And Hair

- Phases
  - Major structures: brow, eye, cheekbone, mouth nose hairline
  - Extrude volume for hair
  - Adjust bottom for extruding ponytails
  - Cut in polygons around eyes, mouth, nose
  - Once done, add some asymmetry (part off center)



Box Modeling: Done

- The completed model in wireframe and shaded



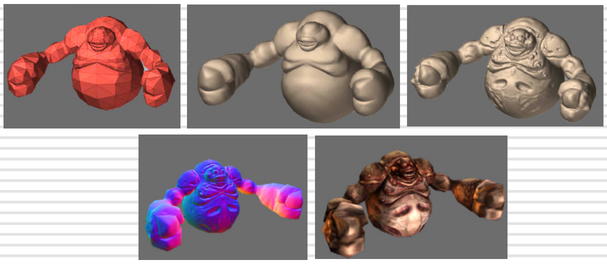
Images courtesy of WildTangent, modeled by David Johnson.

Box Modeling: Summary

- Done for character, but can apply to other things
- General idea:
  - Start with box, cylinder or other primitive
  - Extrude, Cut, Adjust…
  - Get topology, proportions right
  - Once happy, refine until details complete

Other Modeling Techniques: 3D Sculpting

- A low resolution model can be sculpted into a very detailed mesh
- This can be used in game via normal maps
  - (Calculate lighting on each pixel, gives illusion of more polygons of fidelity)



Images courtesy of Pixolgic.

Based on Chapter 6.2, *Introduction to Game Development*

Other Modeling Techniques: Reverse Engineering

- Real world objects or sculptures can be scanned or digitized
- This may not save time because of complicated polygon cleanup, but will ensure high fidelity

Image courtesy of FARO Technologies, Inc.

Other Modeling Techniques: BSP

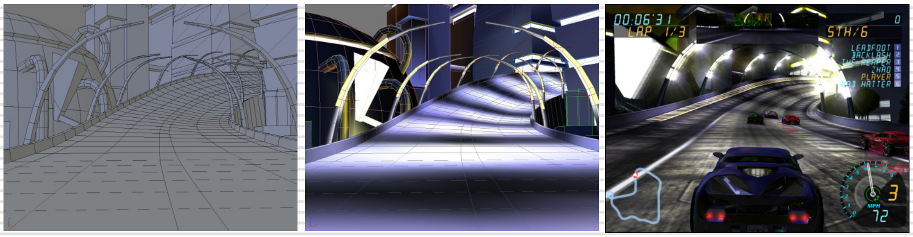
- BSP stands for *Binary Space Partition*
- A coding term that is also method for organizing data
- Like cutting away a mineshaft
  - Start inside solid room
  - Cut away chunks with primitives
- Satisfying since can make space quickly
- BSP Editors come with many games like *Quake*, *Unreal* and *Half-Life*

Low Poly Modeling (1 of 3)

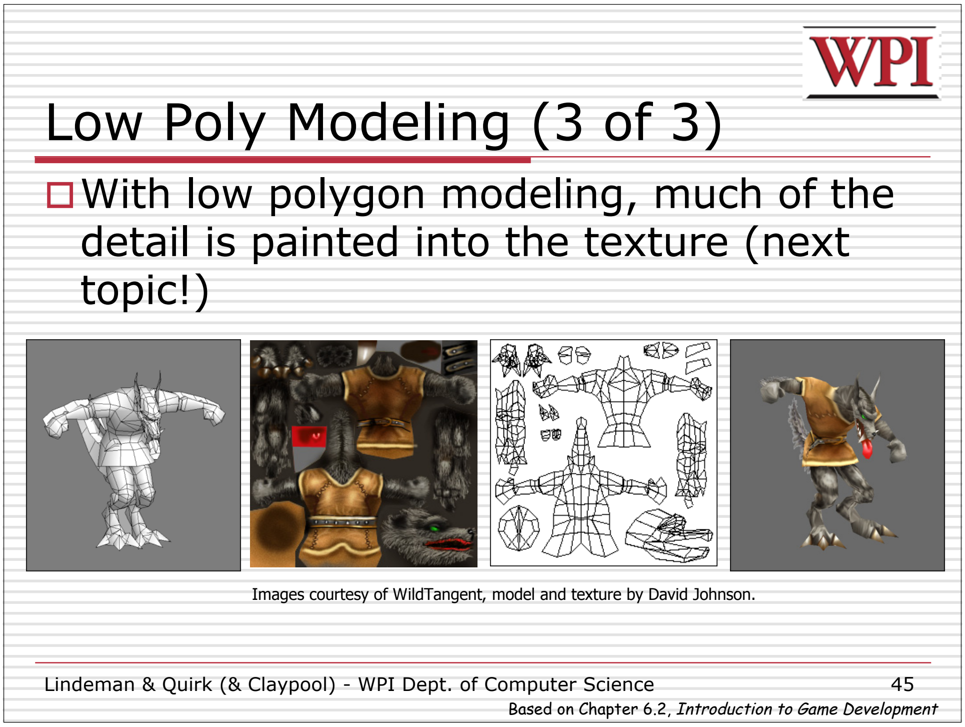
- □ Again, too many polygons results in lower frame rates
- □ To keep frame rates consistent, use level-of-detail (LOD) meshes
  - ■ Multiple versions of object, progressively lower levels
- □ When far away, use low level
  - ■ Assume more objects in Field of View
- □ When close, use higher level
  - ■ Assume fewer objects in Field of View

Low Poly Modeling (2 of 3)

- □ For entire level (ie- map with environment), entire polygon count matters
  - ■ Impacts amount of memory needed
- □ But only visible polygons rendered
  - ■ Rest are "culled" and not computed



Images courtesy of WildTangent

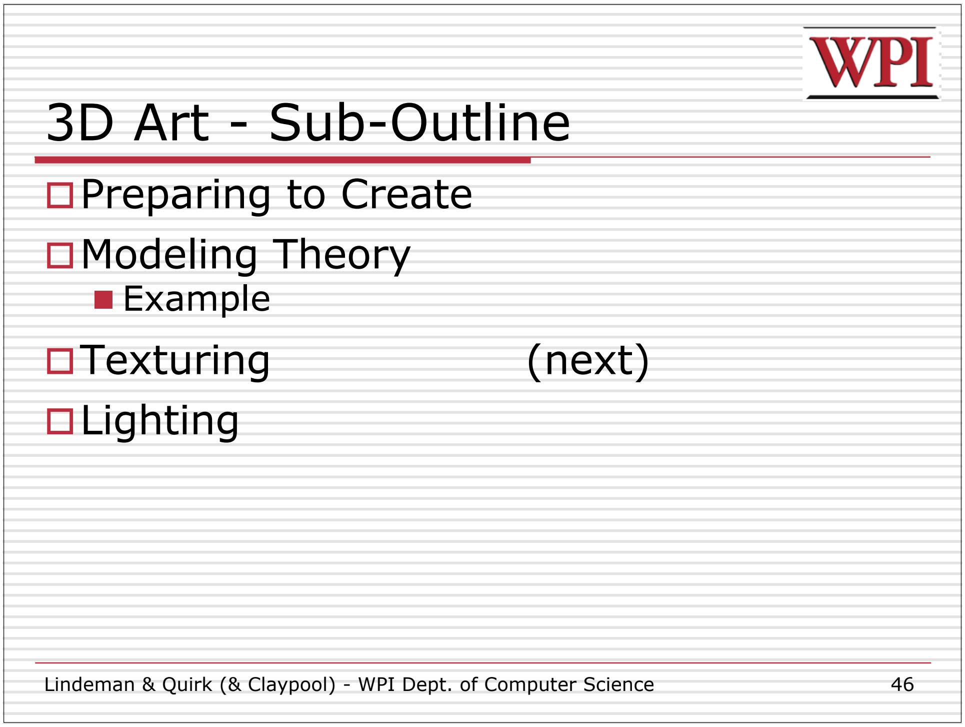


Low Poly Modeling (3 of 3)

- □ With low polygon modeling, much of the detail is painted into the texture (next topic!)

Images courtesy of WildTangent, model and texture by David Johnson.

Based on Chapter 6.2, *Introduction to Game Development*



3D Art - Sub-Outline

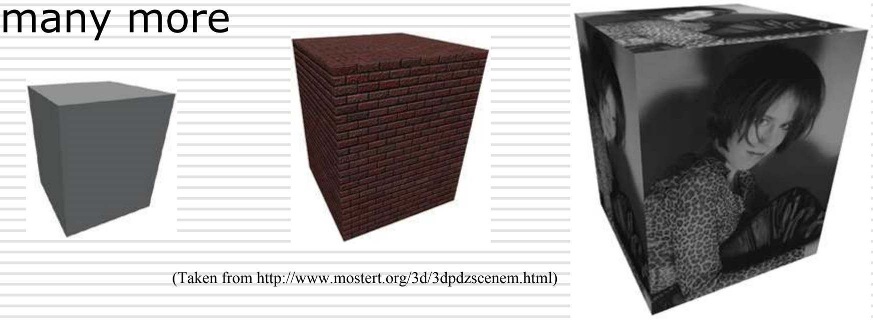
- □ Preparing to Create
- □ Modeling Theory
  - ■ Example
- □ Texturing (next)
- □ Lighting

Texturing

- Motivation
  - Games rely heavily for realism
  - Important to compensate for low geometry
  - Challenging, yet rewarding
- Distinction between texture and shader
  - *Shader* – define surface property of object – how shiny, bumpy, how light effects
  - *Texture* – bitmap plugged into shader that defines image we want to appear on object

Detail in Texture

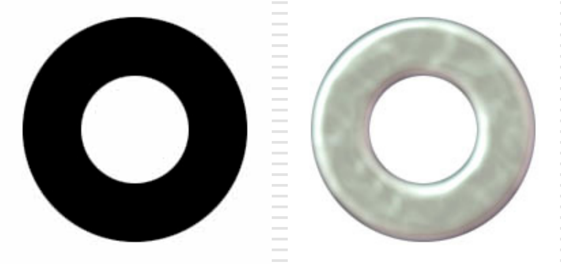
- Add depth, lines, etc. without polygons
- Box is 12 polygons, bricks would take many more



(Taken from http://www.mostert.org/3d/3dpdzscenem.html)

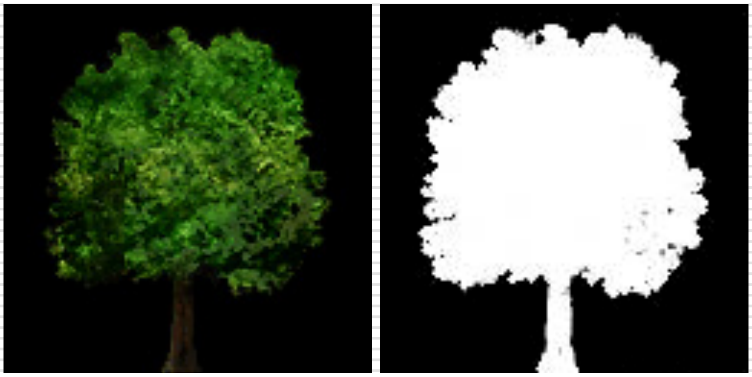
A Brief Word on Alpha Channels

- □ The embedded extra 8 bits of 32-bit image
  - 24 bits gives true color, 2^{24} ~ 16 million colors)
- □ Use for:
  - *Transparency*
  - *Reflection*
  - *Bump maps*



Alpha Channel - Transparency

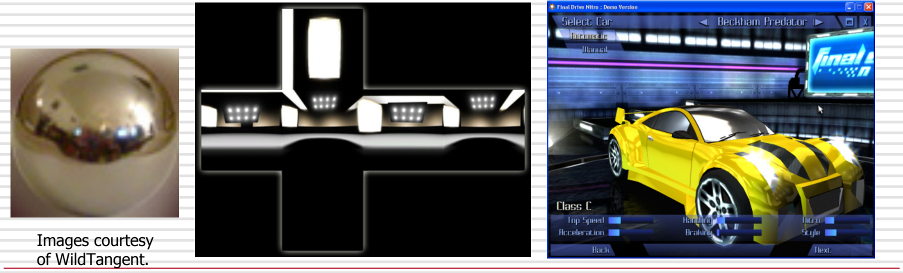
- □ Used to create transparency
- □ White means opaque, black means transparent, grey are values of opacity (semi-transparency)



Images courtesy of WildTangent.

Alpha Channel - Reflection

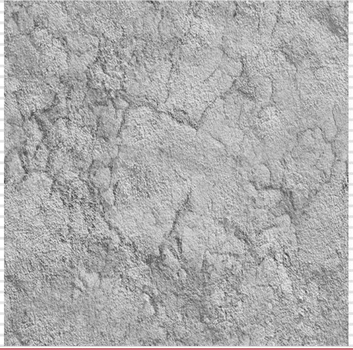
- □ Define what areas reflect light most – human face shiny where oil, water ripples
- □ Three common types of reflection
  - Camera projected – always the same, but can be unrealistic
  - Cubemap – 6 sides, but predefined
  - Dynamic – sides computed on the fly



Images courtesy of WildTangent.

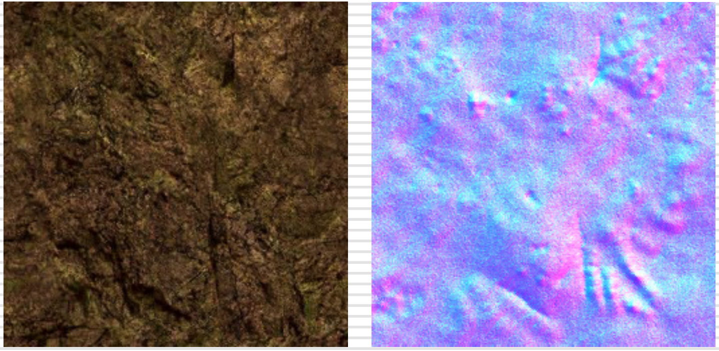
Alpha Channel - Bump Map

- □ Use to create illusion of varying heights
- □ Light is protrusion, dark is recession
- □ Tweaks each pixel based on grayscale value



Alpha Channel - Normal map

- □ A variant of bump mapping
- □ Uses color images (RGB) instead of grayscale
  - ■ Still tweak each pixel

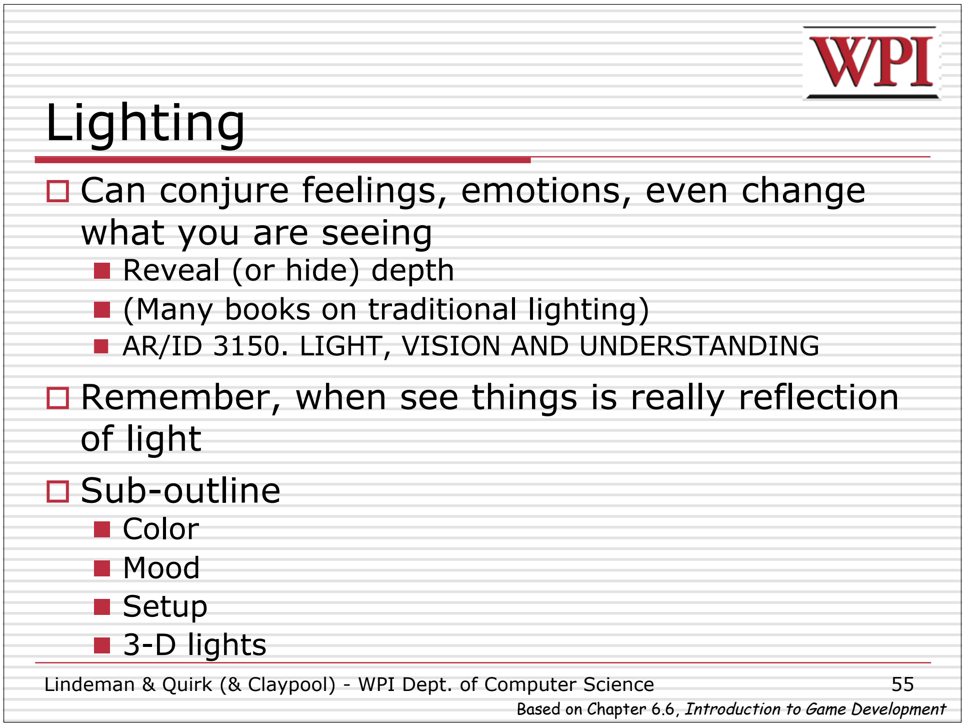


Images courtesy of WildTangent.

Based on Chapter 6.5, *Introduction to Game Development*

3D Art – Sub-Outline

- □ Preparing to Create
- □ Modeling Theory
  - ■ Example
- □ Texturing
- □ Lighting (next)

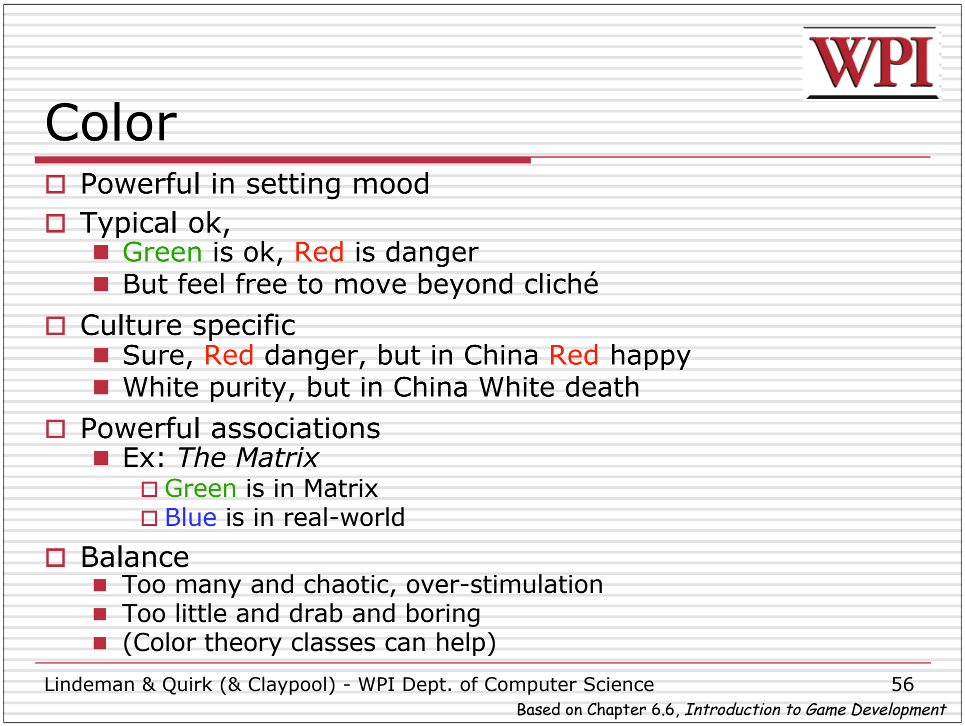


Lighting

- Can conjure feelings, emotions, even change what you are seeing
  - Reveal (or hide) depth
  - (Many books on traditional lighting)
  - AR/ID 3150. LIGHT, VISION AND UNDERSTANDING
- Remember, when see things is really reflection of light
- Sub-outline
  - Color
  - Mood
  - Setup
  - 3-D lights

Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 55

Based on Chapter 6.6, *Introduction to Game Development*



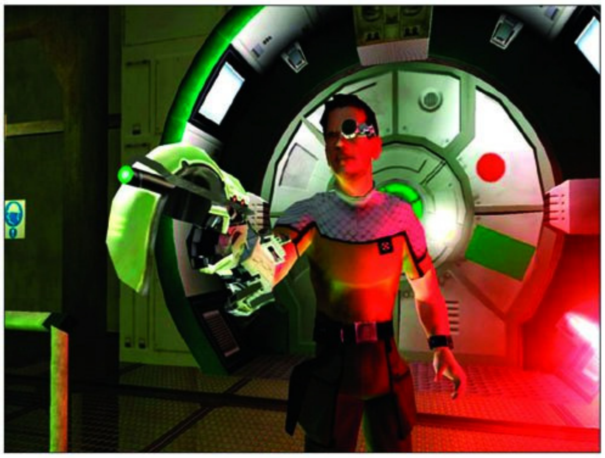
Color

- Powerful in setting mood
- Typical ok,
  - Green is ok, Red is danger
  - But feel free to move beyond cliché
- Culture specific
  - Sure, Red danger, but in China Red happy
  - White purity, but in China White death
- Powerful associations
  - Ex: *The Matrix*
    - Green is in Matrix
    - Blue is in real-world
- Balance
  - Too many and chaotic, over-stimulation
  - Too little and drab and boring
  - (Color theory classes can help)

Lindeman & Quirk (& Claypool) - WPI Dept. of Computer Science 56

Based on Chapter 6.6, *Introduction to Game Development*

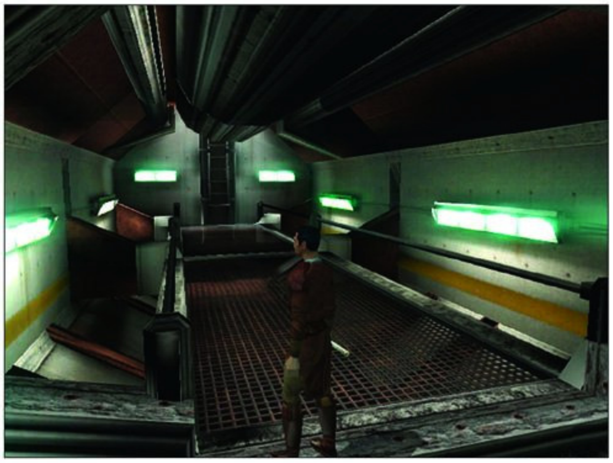
Color Indicates Danger



RTX Red Rock

http://www.informit.com/articles/article.asp?p=174370

Pleasing Colors



Star Wars: Knights of the Old Republic

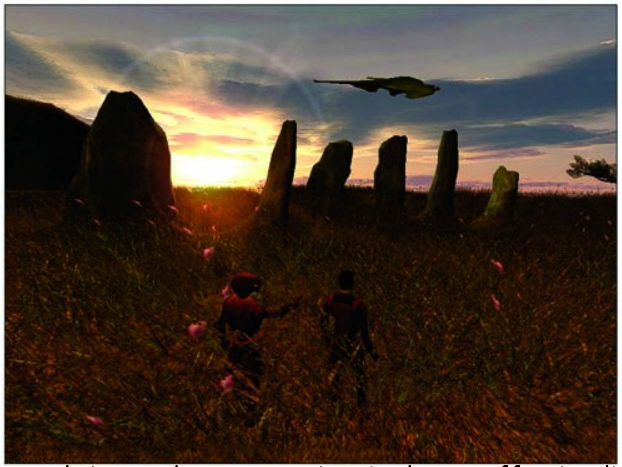
http://www.informit.com/articles/article.asp?p=174370

Mood

- Intensity, direction, angle, number of lights, and shadows all affect mood
- Even humidity, dust, air quality

Based on Chapter 6.6, *Introduction to Game Development*

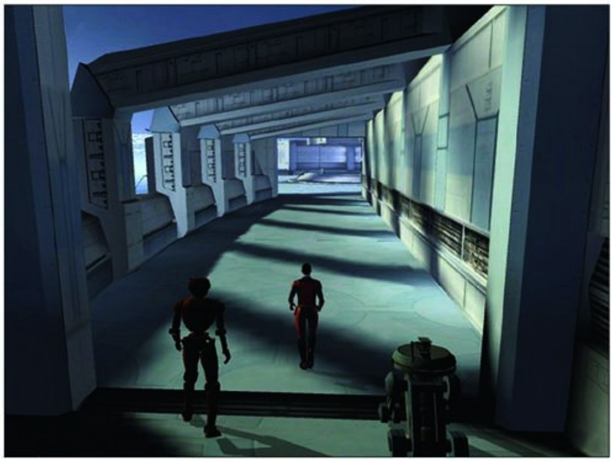
Mood by Lighting Example (1 of 3)



A calming outdoor scene using simple, yet effective, lighting

http://www.informit.com/articles/article.asp?p=174370

Mood by Lighting Example (2 of 3)



Long shadows not only add to the atmosphere, but also help break up repetition

http://www.informit.com/articles/article.asp?p=174370

Mood by Lighting Example (3 of 3)

Light beams and rays give clues as to the humidity, dust, and air quality in a scene

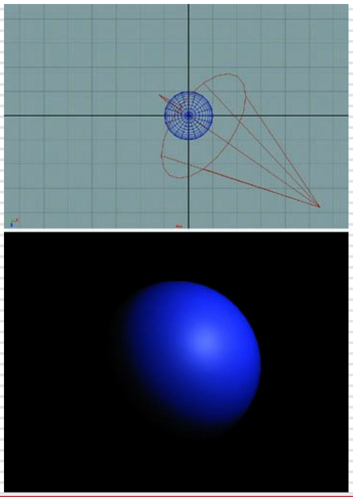
http://www.informit.com/articles/article.asp?p=174370

Lighting Setup (1 of 3)

- □ Traditional lighting
  - ■ Key light, Fill light, Back light
- □ *Key light* – main light source. Most intense and majority. Put at angle to define 3-D forms.

Based on Chapter 6.6, *Introduction to Game Development*

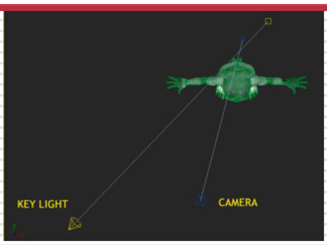
Lighting Setup Example

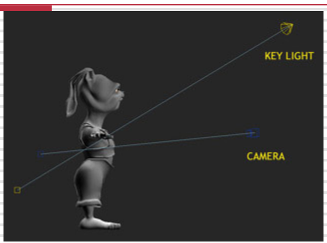


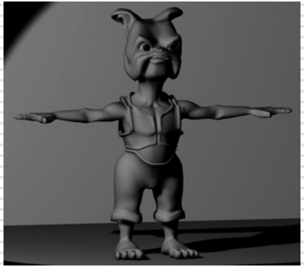
A sphere lit only by a key light positioned at an angle. The detail and form of the sphere are not as clear as if we added another light source.

http://www.informit.com/articles/article.asp?p=174370

Lighting Setup Example







The Key light is placed next to the camera, about 35-45 degree angle to the subject. The angle is determined by what kind of mood that you want the scene to have.

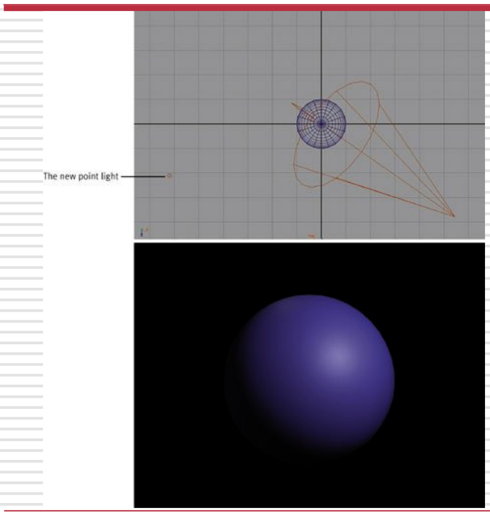
http://www.3dtotal.com/team/Tutorials/Jenns3pt_tut/3ptlighting.asp

Lighting Setup (2 of 3)

- *Fill light* – Brings out some details out of shadow. Works well at angle.

Based on Chapter 6.6, *Introduction to Game Development*

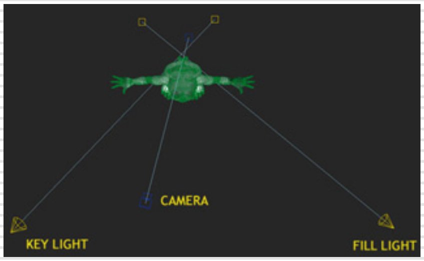
Lighting Setup Example



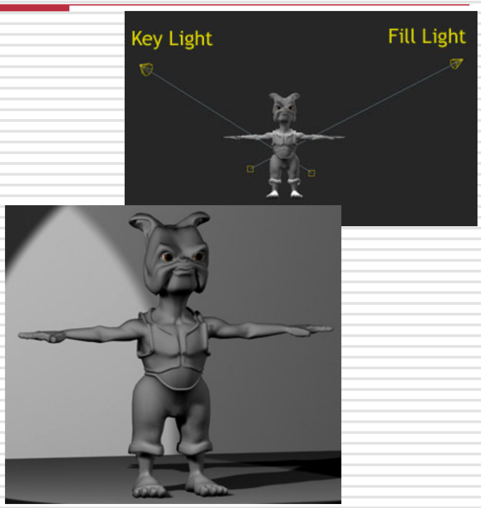
A fill light brings out more form, and softens the shadows from a fill light. Notice the point light has been added to the left of the sphere.

http://www.informit.com/articles/article.asp?p=174370

Lighting Setup Example



Place the Fill Light at a 90 degree angle from the Key Light, usually slightly higher or lower than the Key Light.



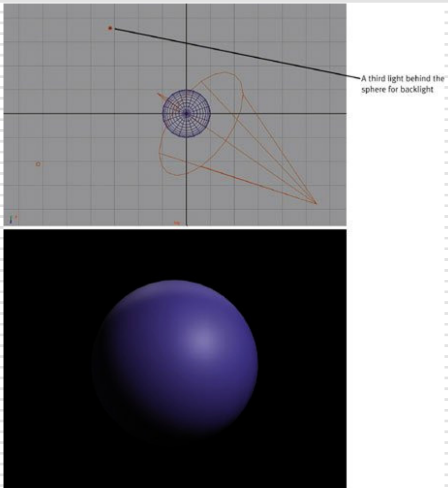
http://www.3dtotal.com/team/Tutorials/Jenns3pt_tut/3ptlighting.asp

Lighting Setup (3 of 3)

- □ *Backlight* – Placed behind and slightly above or below object to help define shape. Highlights edges, pulls away from background.
  - (Also called the *Rim Light* or *Hair Light*)

Based on Chapter 6.6, *Introduction to Game Development*

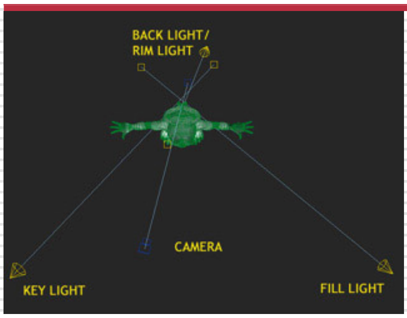
Lighting Setup Example

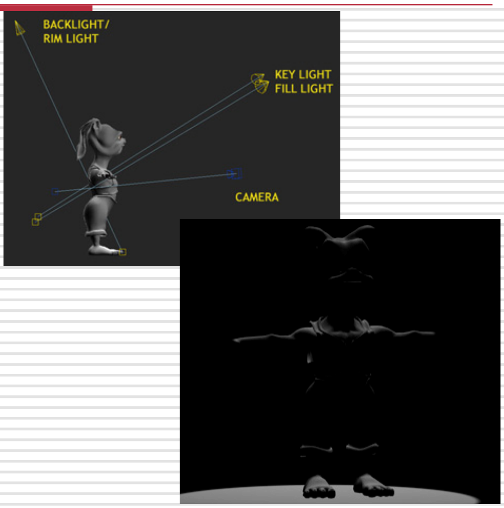


The addition of the third light highlights the edge, helping give the sphere more dimension.

http://www.informit.com/articles/article.asp?p=174370

Lighting Setup Example

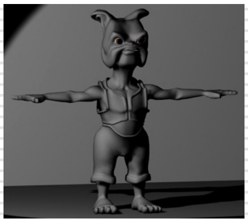


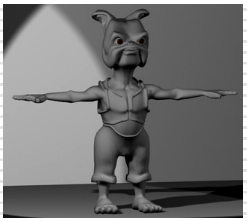


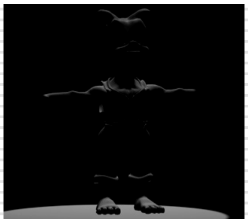
The Back Light is placed directly opposite the camera and behind the subject.

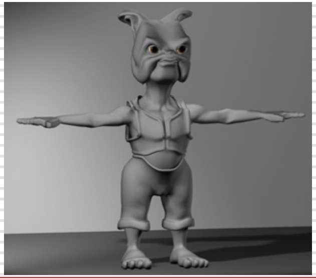
http://www.3dtotal.com/team/Tutorials/Jenns3pt_tut/3ptlighting.asp

Result: Key + Fill + Back Lights







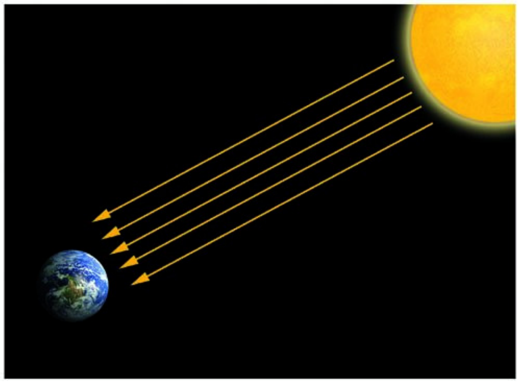


Working with 3D lights (1 of 3)

- □ 3-D lighting different than traditional lighting
  - ■ Start with traditional and modify until you get desired affect (broad strokes)
- □ Tools give different kinds of lights
  - ■ (next)
- □ A few effective practices
  - ■ (after)

Working with 3D lights (2 of 3)

- □ *Directional Lights* – used for sunlight or moonlight. Often as key light. Predictable.

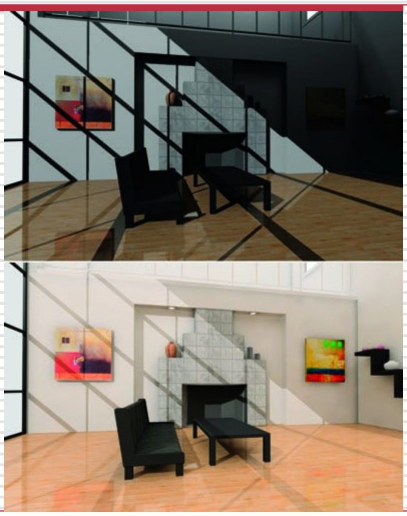


By the time the sun's rays reach the earth, they are nearly parallel to one another.

Working with 3D lights (3 of 3)

- *Ambient Lights* – spread everywhere, equally. Uniform diffuse lights.
  - Can skip by creative placement of the Fill Light, but gives more precise control over illumination
- *Spot Lights* – focus beam on single location. Great control.
- *Point Lights* – single point in all directions. Light bulbs, candles, etc.
- *Background Light* - soften the areas of the background that Key Light doesn't illuminate

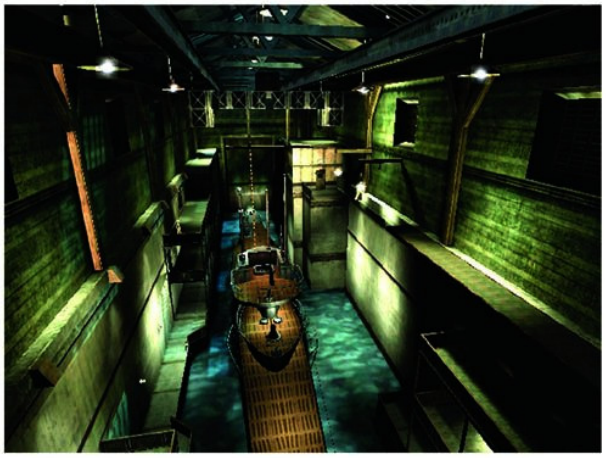
Example of Working with 3D lights



A room lit without radiosity. Bottom The same room with a radiosity solution.

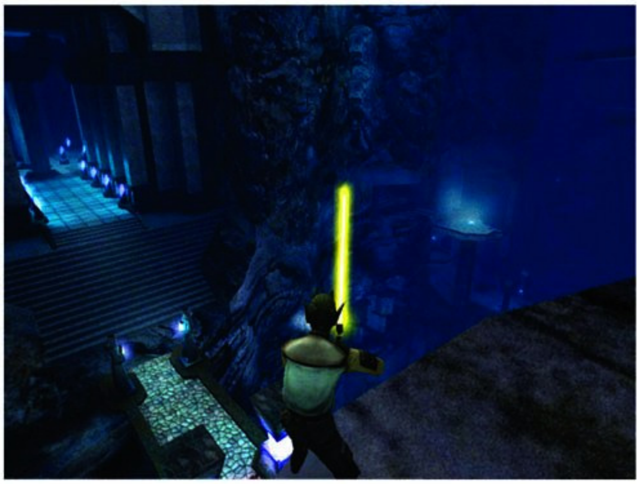
Effective Lighting Practices (1 of 3)

Pools of light
– Don't always try to light evenly.
- *Gives sense of mystery*

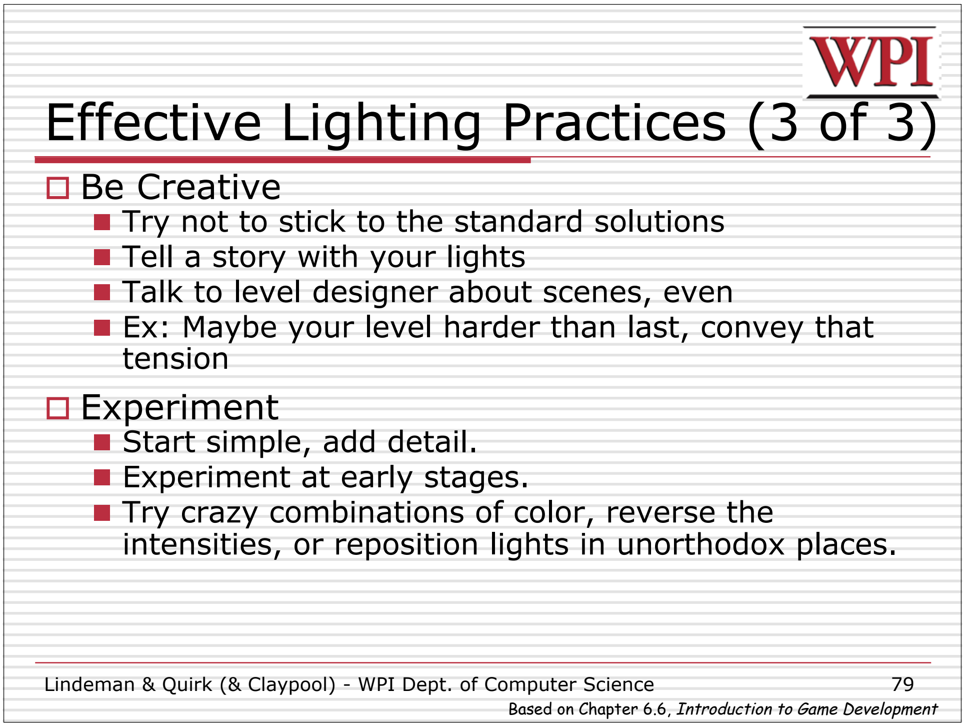


Pools of light in Indiana Jones: The Emperor's Tomb

Effective Lighting Practices (2 of 3)

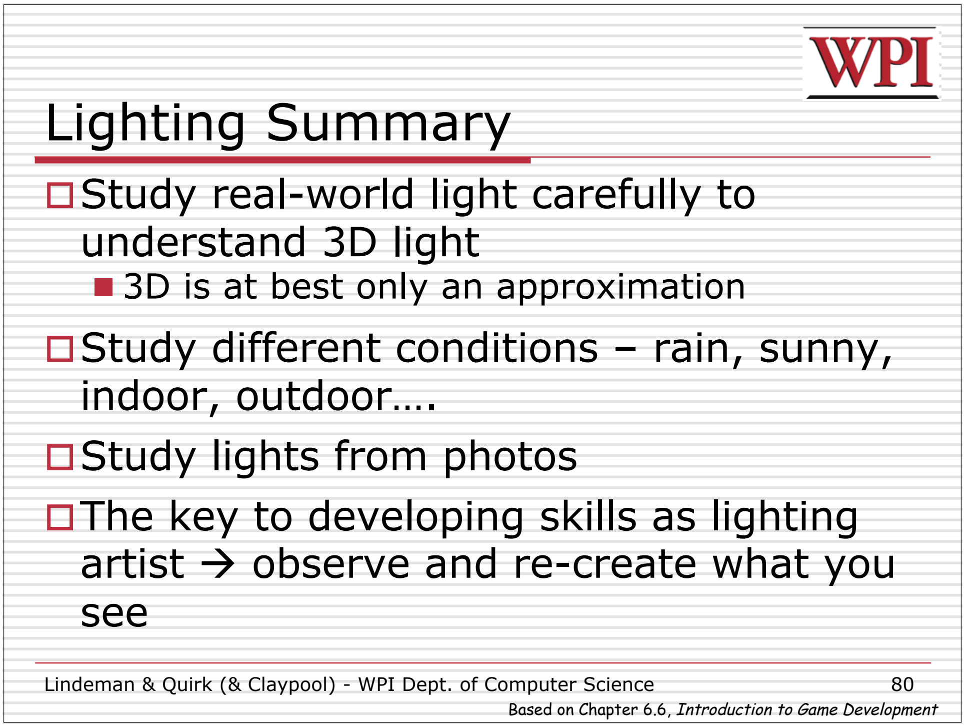


Guide lights –
- Use light to guide the player.
- Helps highlight areas that are accessible and important to the objectives.



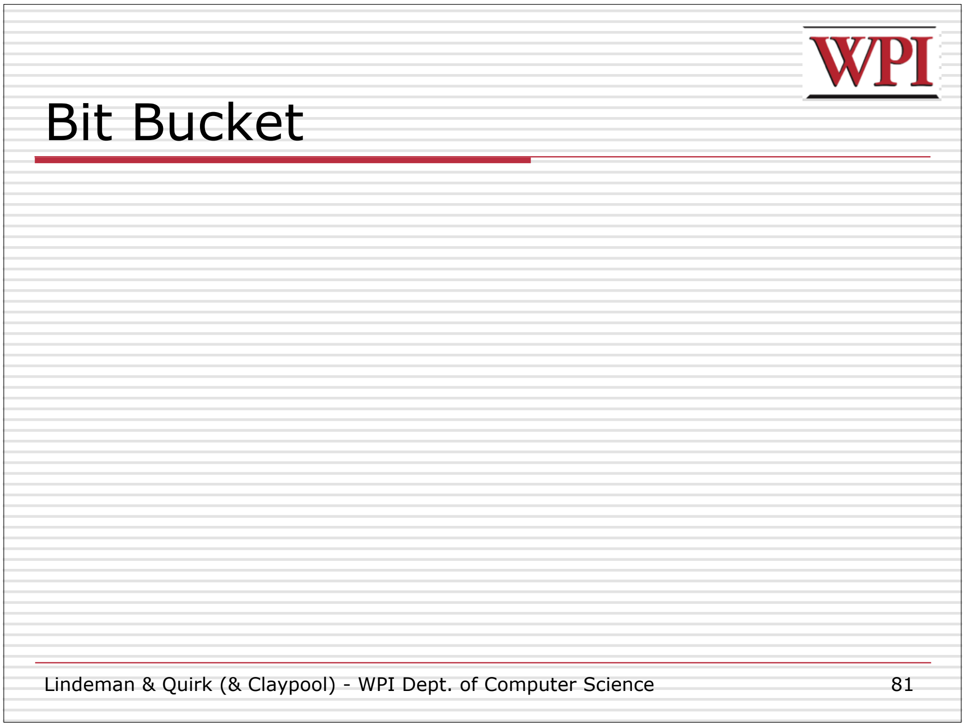
Effective Lighting Practices (3 of 3)

- □ Be Creative
  - ■ Try not to stick to the standard solutions
  - ■ Tell a story with your lights
  - ■ Talk to level designer about scenes, even
  - ■ Ex: Maybe your level harder than last, convey that tension
- □ Experiment
  - ■ Start simple, add detail.
  - ■ Experiment at early stages.
  - ■ Try crazy combinations of color, reverse the intensities, or reposition lights in unorthodox places.

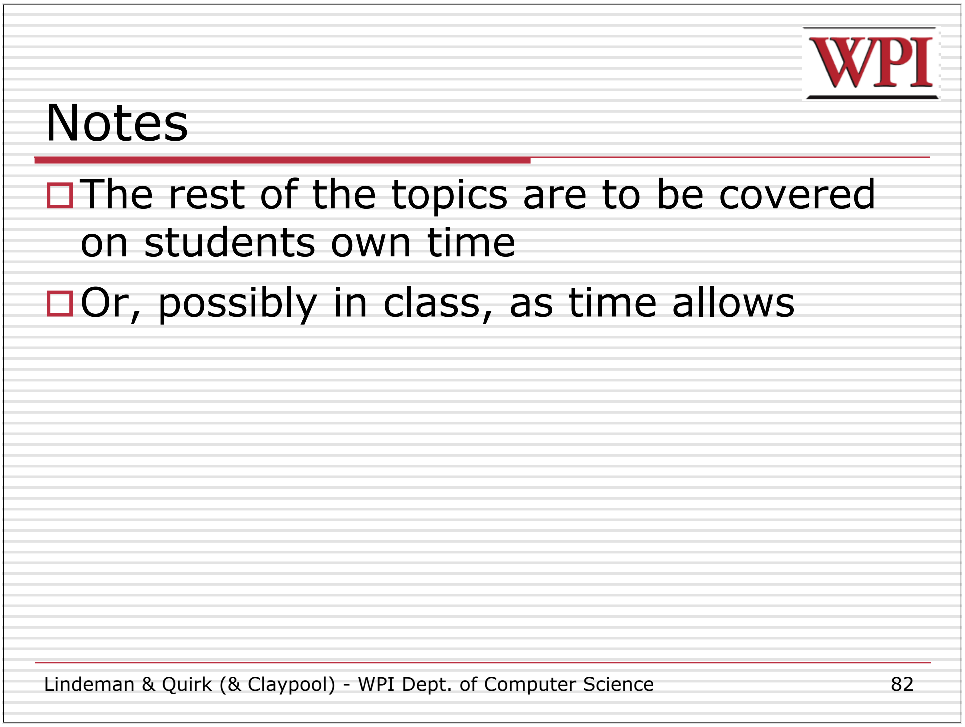


Lighting Summary

- □ Study real-world light carefully to understand 3D light
  - ■ 3D is at best only an approximation
- □ Study different conditions – rain, sunny, indoor, outdoor….
- □ Study lights from photos
- □ The key to developing skills as lighting artist → observe and re-create what you see



Bit Bucket



Notes

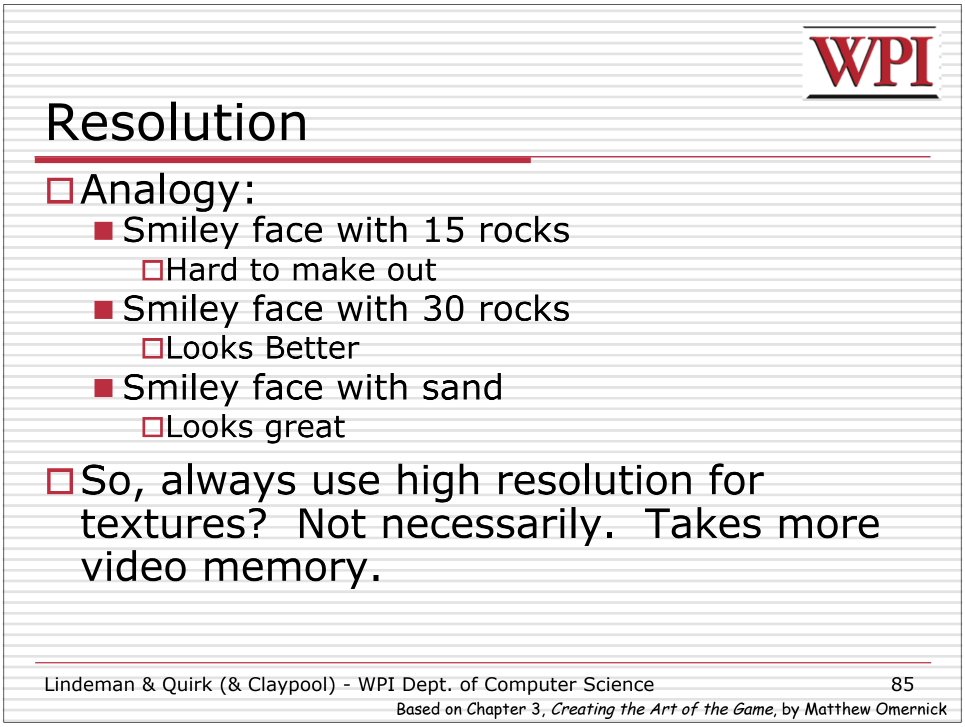
- □ The rest of the topics are to be covered on students own time
- □ Or, possibly in class, as time allows

Make Interesting Textures

- Consider story behind object
- Consider door (contoured, so could do geometry, but cheaper to put picture up)
- Could just take one on Internet and put up
- But can make more believable
  - How old? Who uses it? Repainted? How long ago?
- Add grunge around knob, show nicks at bottom, flecks of color where repainted …

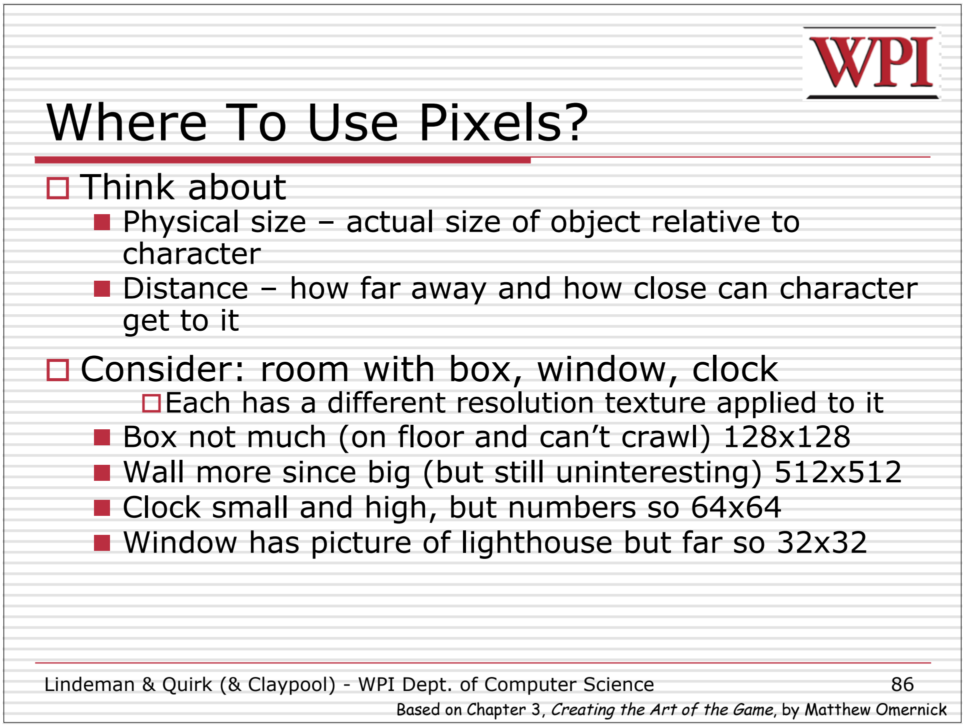
Textures are Their Own Artwork

- Rarely ready to go … spend time in Photoshop massaging, customizing
- Think of each texture as custom artwork
- Before and after page 49
  - Wood → with coffee mug stain, nicks and scratches
  - Window → depth in reflections, uneven opacity
  - Concrete → cracks, discoloration
- Need to be aware if tiled and reused
  - Interesting textures harder to re-use since noticeable



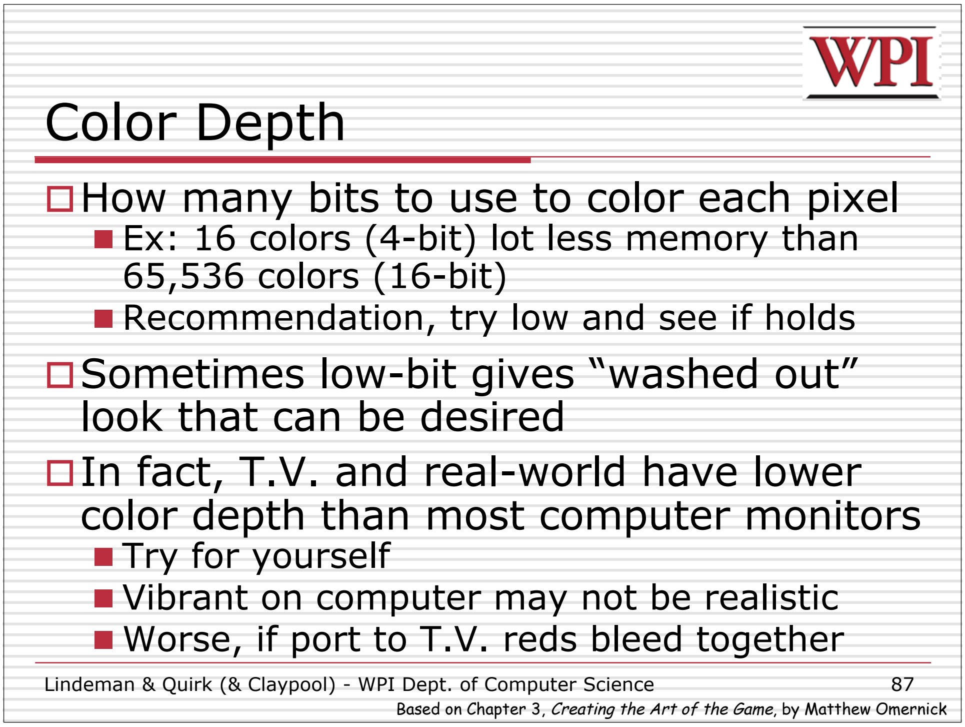
Resolution

- Analogy:
  - Smiley face with 15 rocks
    - Hard to make out
  - Smiley face with 30 rocks
    - Looks Better
  - Smiley face with sand
    - Looks great
- So, always use high resolution for textures? Not necessarily. Takes more video memory.



Where To Use Pixels?

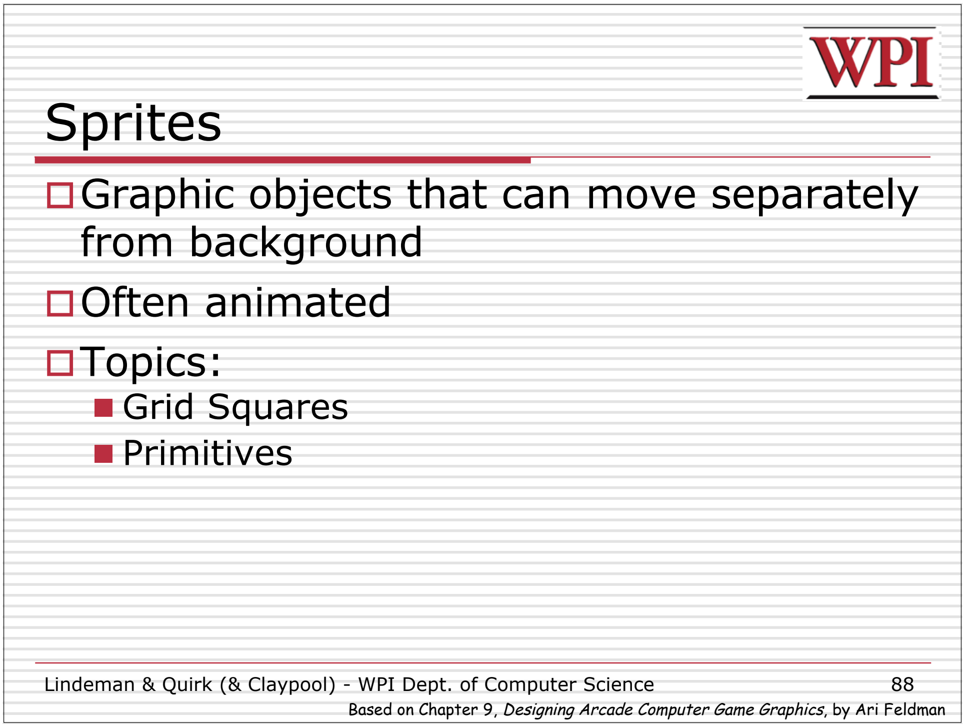
- Think about
  - Physical size – actual size of object relative to character
  - Distance – how far away and how close can character get to it
- Consider: room with box, window, clock
    - Each has a different resolution texture applied to it
  - Box not much (on floor and can't crawl) 128x128
  - Wall more since big (but still uninteresting) 512x512
  - Clock small and high, but numbers so 64x64
  - Window has picture of lighthouse but far so 32x32



Color Depth

- □ How many bits to use to color each pixel
  - ■ Ex: 16 colors (4-bit) lot less memory than 65,536 colors (16-bit)
  - ■ Recommendation, try low and see if holds
- □ Sometimes low-bit gives “washed out” look that can be desired
- □ In fact, T.V. and real-world have lower color depth than most computer monitors
  - ■ Try for yourself
  - ■ Vibrant on computer may not be realistic
  - ■ Worse, if port to T.V. reds bleed together

Based on Chapter 3, *Creating the Art of the Game*, by Matthew Omernick



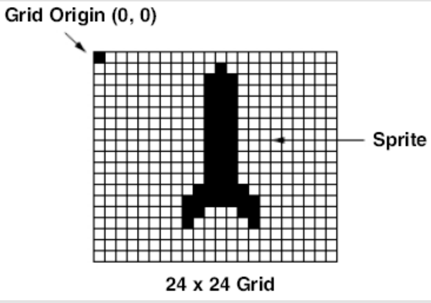
Sprites

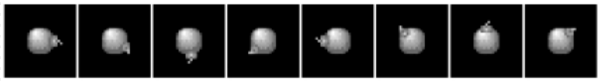
- □ Graphic objects that can move separately from background
- □ Often animated
- □ Topics:
  - ■ Grid Squares
  - ■ Primitives

Based on Chapter 9, *Designing Arcade Computer Game Graphics*, by Ari Feldman

Grid Squares

- □ "Mini-Screen" to depict
  - ■ Individual pixel modifications
- □ Help observe animation progression
- □ (Show Game Maker image editor example)
- □ Strips for tools





http://www.flyingyogi.com/fun/download.cgi?spritelib

Based on Chapter 9, *Designing Arcade Computer Game Graphics*, by Ari Feldman