Assignment 1 – Concepts and Principles of 3D

Unit 66: 3D Environment

Assessor: Tracey Clarke

Dean Smith - 30153732
10/03/2010
Contents
Introduction.................................................................................................................. 3

Part 1 - The application and use of 3D within computer games .................................. 4
  Pre-Production ......................................................................................................... 4
  Production .............................................................................................................. 7
  Post Production ...................................................................................................... 9

Part 2 - Displaying 3D models within 3D development software ............................... 10

Part 3 - Geometric theory of 3D and 3D mesh construction ..................................... 15
  Additional Info ........................................................................................................ 20

Part 4 - Constraints of producing 3D models for computer games ............................ 23

Bibliography.............................................................................................................. 26
Introduction

This document is to show that I understand the basic concepts and principles of 3D design within the computer games industry. I will achieve this by explaining the subject across 4 separate areas.

These are:

- The application and use of 3D within computer games
- Displaying 3D models within 3D development software
- Geometric theory of 3D and 3D mesh construction
- Constraints of producing 3D models for computer games
Part 1 - The application and use of 3D within computer games

In contemporary 3D games design, there are many more features to an environment that require more technical expertise than has been seen on any previous generation platform. Elements ranging across small objects such as weapons and debris, vehicles and forms for characters and entire landscapes, which contain geological formations and architecture, must be constructed in correlation within current video game constraints. These constraints include tri and quad point polygons within geometrics, correct polygonal scale and single mesh construction. The aim is to remain consistent with the industry trend of implementing as much realism within a game as possible which sees the media type conforming to more simulation than video game stereotypes.

As within many media driven industries the implementation of any 3D development falls within the following 3 development categories:

- Pre-Production
- Production
- Post Production

Pre-Production

Purpose

The intention of the pre-production phase is to develop the structure and plan for a project with all factors analysed. This would include key areas such as the target audience, time restriction, cost and feasibility. An example of this would be the planning of a children’s, Christmas holiday themed production required to be developed before December 25th. The majority of these would be found through Primary and Secondary Research techniques.

With these details considered, the development team would begin construction of an ‘Initial Design Proposal’ to display and consider the ideas and pre-visualization concepts of characters, settings, themes and narrative from Games Designers as well as concept artists. This would be developed as a document, verbal pitch, or combination of the two, for the purposes of advertising the project to a potential publisher or higher authority. The designs included within this document would be similar to a portfolio layout spread containing relevant information and annotation. Within this phase of development, the artists and designers would make use of a variety of methods in which to create and plan elements within the project. Aside from multiple skill styles with regards to manual artwork such as pen, pencil and colouring media, the employees would also utilize various CAD applications.
to create and render images and/or animatics. Examples of these applications could include Adobe Photoshop or Fireworks.

An artist in this category would begin by producing rough sketch plans, which include annotations as to the purpose of the concept, and begin to develop the design further, adhering to strict deadline schedules. An artist may also be required to produce high quality 3D rendered images for this. Examples of this can be found overleaf.
The image to the left suggests possible ideas for a project environment. In this case the work has been created for a 3D animation production. The spread contains artwork not only art styling for the images but also the intended mood. A technical plan for the environment is also present.

This image suggests multiple designs for the animations lead character within a variety of styles. A short description as to the purposes of the character is also present alongside rough sketchwork.

The series of images contained within this spread detail a Moodboard for the animation. Using highly detailed concept work, the series displays the intended scenes with regard to colour style and shade. These colour ranges are also simplified beneath each of the images.
This would then lead to a fully comprehensive Game Design Proposal, a more detailed document which contains the intended level designs, plotlines, character designs, vehicles, weaponry, user interfaces and any techniques and restraints that are to be abided by in order to achieve the desired project. The document is also intended to allow the publisher to gain insight into how the game is intended to be played and who and what the game is to be designed for.

In addition, basic tests may be applied during Pre-Production that may be considered such as 'white box testing' for the idea and a possible prototype to be include with the 'Game Design Proposal. These are the blank slate designs for the program that the developers perform to test qualities of the game such as physics, gravity and playability. Test models may also be produced along with basic layouts of the intended environments to view scale and proportion.

Production

Following on from the aforementioned concepts and initial IDPs (Initial Design Proposals), the development team would then begin the actual physical production of the project, a process whereby all of the intended game content is to be produced, including any imbedded source code. The phase is divided into sections of development covering Design, Programming, Level Creation, Artwork, Audio, and Testing. When considering 3D production, it is the Design, Programming and Level Creation that are the most important stages of the production.

Game Design

The overall production lead usually manages this segment of the production utilizing a high level of expertise. Here the game rules are developed alongside any interactions of more advanced models and their representation within the game environment. The designers may manually aid the production team within the 3D development software to ensure that the correct content is achieved, such as Newtek LightWave Layout or AutoDesk’s 3DS Max. An example of this would be ensuring that the animation team meets the required criterion of motion in order to meet the game themes. Designers must also be familiar with any proposed game engines such as UDK as software within this category aid in level development.

The HUD and menu layouts for a game would also be designed and incorporated here along with network messages.

Themed HUD displaying tactical information for Command & Conquer 4
**Programming**

3D objects and locations may require a high amount of source coding to deliver various effects from the models such as AI. Other examples may include the use of Machinima cut screens that divert the view of a 3D level, for example, from a users control to highlight specific points of interest. Games such as Tomb Raider on the Playstation made heavy use of this technique to guide users. Various programming languages have been developed specifically for this purpose such as DarkBASIC.

**Level Creation**

This phase can prove very time consuming and costly to the project as attempting to construct functional levels for use within intended games engines, as well as any planned sequences is very difficult. The environments and plot points may constantly need to be tweaked, edited or even redesigned in order to match the specifications of the game-play or compatibility with AI. Usually created in 3D development software such as AutoDesk Maya or 3DSMax, an initial draft of the level geometry is produced and tested with the other elements of the production. This would usually incorporate features which may become redundant as the development process reaches later stages. Game engines such as UDK contain features which aid in level creation as well as planned level events.

**Artwork**

An area that deals mainly with the artistic elements required for a video game project such as Packaging designs, User Manuals, title menu art styling and in game textures to be used on 3D models. Many of the Pre-Production concept works are used in this area for presentation however with regards to the texture process, various maps may also be produced such as Diffuse, Normal and Ambient Occlusion. These maps provide 3D geometry with fine detail such as light and depth management and are produced using a combination of software. Such software may be XNormal used to generate the maps from textures created in application such as Adobe Photoshop or Fireworks. This process is familiarly known as ‘baking’ a model.
Testing

The testing phases review the designs and functionality of the game and ensure all 3D elements are working correctly. These stages are:

- **Alpha testing** – Preliminary testing that ensure all functionality and game specifics are working correctly. Coding is still amended at this phase.

- **Code Freeze** – This phase sees the development team correcting bugs only. Programming code ceases at this stage.

- **Beta Testing** – A finished draft of the game is produced and tested continually by testers for bugs and glitches.

- **Code Release** – These tests are performed to ensure the game meets the correct standards by Publishers and other authoritative bodies.

Post Production

This final phase of production sees the development team producing elements such as teaser trailers, game demo releases and heavily rendered screenshots for advertisement purposes, although some smaller teaser projects may be released early to enhance anticipation and hype for the game. Preparation for this phase begins during production with these teasers generated to be seen as the release of the game approaches.
Part 2 - Displaying 3D models within 3D development software

Contemporary 3D development software usually features various methods of displaying a 3D mesh which aid the designer in developing a model. These range from viewports, perspective fields of vision and shading to specularity and luminous UV mapping. The following examples used to detail these methods will be based upon those as seen in Newtek LightWave 3D 9.6

When we first initiate the LightWave modelling application, we are presented with four viewports. These viewports are editable, by pressing ‘D’ for display options, to display a number of specific views. The default Layout upon opening the application displays 3 Orthogonal Views (above the model, left of the model, behind the model) and one Perspective View.

There two key differences between the orthogonal views and perspective view, these are:

- All views can be manipulated to view the model from a different position and scale however the Perspective view also allows the user to rotate around the model.

- The orthogonal views display the wire mesh of the model whereas the perspective view can display a number of useful visual effects for different methods of modelling. Overleaf are some descriptions of these effects:
<table>
<thead>
<tr>
<th><strong>Wireframe</strong></th>
<th>The most basic skeletal structure of a model, displaying edges only. Similar to blueprints, this model is vision permeable through polygons.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Colour Wireframe</strong></td>
<td>Similar to Wireframe, this effect allows the user to custom define the colour of the mesh in display options. Also vision permeable through polygons.</td>
</tr>
<tr>
<td><strong>Hidden Line</strong></td>
<td>A variation of Colour Wireframe with the added exception that edges behind the foremost view of the model are not displayed as though the model. A non-vision permeable solid, although the polygons remain un-displayed.</td>
</tr>
<tr>
<td><strong>Sketch</strong></td>
<td>Sketch gives colour to the polygons of the model. A non-vision permeable view.</td>
</tr>
<tr>
<td><strong>Wireframe Shade</strong></td>
<td>This view displays polygons as though there would be a source of light affecting the model. A non-vision permeable view of the wireframe is also visible. The light is calculated by the angle of the each polygon towards the source.</td>
</tr>
<tr>
<td><strong>Flat Shade</strong></td>
<td>Similar to Wireframe Shade except without the wireframe. Also angle calculated with regards to the light source.</td>
</tr>
<tr>
<td><strong>Smooth Shade</strong></td>
<td>A variation of flat shade that allows the light to be rendered against the entire model.</td>
</tr>
</tbody>
</table>
**Weight Shade**


---

**Texture**

This shows the model with textures applied. Non-vision permeable.

---

**Textured Wire**

This shows the wire mesh alongside the applied textures. Non-vision permeable.

---

Non-Lightwave effects can now include Phong Shading, a culmination of Ambient, Diffuse and Specular effects, and Gouraud Shading, a method of imitating flat shade but with fewer calculations to be processed through estimating via the number of polygons rather than pixels. These variations are now widely used as Phong shading is considered too processor heavy and Gouraud shading, despite being processor effective, renders the light effect looking too triangulated.

---

**PHONG SHADING**

[Image]

**GOURAUD SHADING**

[Image]
The background behind the models includes a grid to define the scale and proportion; this also features a point of origin to keep the user aware of positioning. A useful feature is also the ability to maximise a view port by clicking a triangle in the upper right corner. The model may also be edited in different layers similar to Adobe Photoshop of which can also be used to house other models efficiently. These are accessed through buttons across the top right of the screen.

The following is the aforementioned Display options window. Here many features regarding the GUI of the application can be adjusted.

The Newtek LightWave 3D package comes in to parts; the aforementioned modelling application and the scene render generation application titled Layout. This sister application also uses view ports that are customizable includes views such as camera and light view.
Within this application the user can adjust the camera and various lighting options, such as dome, spherical and spotlight, in order to produce a high quality render of a textured 3D mesh. The application accounts for advanced lighting of the project including shadows and any effect maps used such as Specularity or Ambient Occlusion although these are only visible correctly when rendered.
Part 3 - Geometric theory of 3D and 3D mesh construction

In modern day 3D mesh construction, we construct models using points, edges and polygons. A point or Vertex is a 1 dimensional element within 3D space, in example this is simply a corner of a 3D shape. An edge is also just and edge of a 3D shape, representing a 2 dimensional element within whereas a polygon is a particular 2D face for a shape utilising all 3 dimensions. The term ‘Polygon’ stands for multiple (poly) Geometric Object Normal meaning multiple edges and points. Polygons are organised into 5 classes. These are:

1. Diagon (A polygon consisting of only 2 points)
2. Triangle (A polygon of three points)
3. Quad Point (A polygon of 4 points)
4. N-Gon (A polygon consisting of a greater number of points than 4)
5. Legacy Polygons (A remnant of a polygon that was intentionally deleted)

A model is generated through meshing a number of these polygon types together in to a 3D mesh however; there are constraints that the videogames industry must abide by in order to allow a 3D model to be compatible with videogames that mean a model may only be able to be generated from Triangles and Quad Point polygons. Another constraint is that a single polygon may not be merged through another.

We use Cartesian (XYZ) coordinates to acknowledge position and direction of 3D models within a 3D production environment and depending upon view port, we see these from multiple angles. In addition we can note the direction a polygon is facing from the single line that sires from the centre when highlighted, this is known as a normal.
Additionally in LightWave, a view port can be customized to display the UV map of a model, which can be used to tweak textures.

Any models can be saved into file types such as OBJ files that allow a model to be transferable between applications retaining the model, UV map and vertex data. However, a model in Light Wave can be saved as a LWO (Light Wave Object) which saves all Light Wave settings applied to the model. A file saved specifically for Light Wave Layout is known as LWS (Light Wave Scene).

To generate a model, the most basic method is to place points at chosen locations across the Cartesian coordinates and turn them into a polygon by highlighting them individually and pressing P. If the points were highlighted in a clockwise manner the normal of the polygon will face as though to the user, if counter-clockwise the polygon will face away and the user will be viewing the back of the polygon. Although despite being the basic form of mesh construction theory is more complicated when in use than using pre-defined basic shapes known as Primitives. There are many forms of Primitive within LightWave but the ones available from the outset are as follows:

- **BOX**
- **BALL**
- **DISC**
- **CONE**
- **CAPSULE**
A primitive is generated by clicking the left mouse button and dragging across one of the Orthogonal View Ports to generate a base (which can be used on its own as a single polygon) and then across another to add the other dimensions, the third axis is generated automatically. The user sets the desired scale and proportion of the Primitive and then confirms its use by tapping Enter. Additionally a Primitive can be divided into a number of polygons before the being confirmed with the use of the arrow keys, the division axis changes depending on which orthogonal view the mouse is hovering over. Here is a divided box.

Within Light Wave there are many tools in which the user can manipulate the polygons of which the most common can be activated via the use of keyboards shortcuts such as the Translate or ‘drag’ tool (Hotkey: T) which allows a user to relocate selected geometry. The following reviews more technical tools in greater detail.

**Bevel**

The Bevel tool simply takes a selected polygon and allows a user to drag it out from its original location, re scaling it based on the movement of the mouse in the designated orthogonal viewport, and generating polygons in its trail. By this it is meant that the selected polygon separates from the initial vertices and is manoeuvred to a newly desired size and/or position with the said vertices retaining their original placement, whilst having new polygons generated between gaps. The polygon itself contains copies of these vertices. The effect can be applied to multiple selected polygons with the same effect seen on each. If dragged away from the original any extra polygons face outwards whereas the opposite direction sees these ‘Normals’ facing inwards.

The Bevel tool is useful when an artist intends to generate geometry that increases in and/or decreases in width such as the example below which displays the Bevel tool being used in the creation of simple 3D table from a single disc polygon. The tool additionally functions in ‘Symmetry’ mode and is accessed through the hotkey ‘B’.
Extrude

Similar to the Bevel tool, this feature allows an artist to relocate a polygon leaving additional polygons in the trail in the same fashion. The key differences with this tool however is that the polygon is not re-scaled and the original polygon is left in place with a copy being relocated.

This is useful in mesh construction due to the user quickly being able to extend particular features of a model such as this example showing displaying a sub-divided box being turned into a geometry containing a 90 degree angle. The tool additionally functions in ‘Symmetry’ mode and is accessed through the hotkey ‘B’.

Sketch, Spline Draw and Bezier

These tools allow an artist to create curved edges of points. These have many uses such as various effects when used in conjunction with other tools such as Rail Extrude or the basis of a ‘2 Vertex Poly Chain’, created from extruding the vertices and deleting the generated copies for use with effects such as electrical animation. The Sketch tool simply allows a user to draw out a line which becomes the edge with vertices placed accordingly. The Spline Draw tool allows a user to select a point in 3D space to create a mark location, and select another point of which an edge is shown. If another mark is added the Spline automatically curves towards that location, the edge is created when ‘Enter’ is pressed on the keyboard. The Bezier behaves in a similar manner to the Spline Draw tool with the added exception of including a tool which allows the user to dynamically manipulate the curve in a Bezier styled fashion, also generated when pressing ‘Enter’.
Rail Extrude

The Rail Extrude tool allows a user to take a pre-generated Spline placed in another layer on non-interactive mode, place a given polygon at its source and automatically extrude the geometry across the curve or 'Rail'. This is useful in creating items such as wires. The Hotkey for this effect is Shift+R.

Boolean

In the same manner as Rail Extrude whereby geometry is placed in another layer, the Boolean tool enables to use this as a cutting tool to cut or add in geometry from the highlighted shape. The application automatically maintains the object as a single mesh although this can create issues if modelling for a video game engine. This is useful for cutting areas out of walls to fit the shape of a door for example. The images below display a sphere being used to cut a bowl into a cube.
Basic manipulation tools also include

- Rotate (Y) – Rotates the selection
- Flip (F) – Flips a polygons normal
- Clone (C) – clones the selection
- Cut (Ctrl + X) – Cuts the selection
- Paste (Ctrl + V)– Pastes the selection
- Scale (H) – Rescales the selection
- Triangulate (Shift+T) – divides the selections into 3 vertex polygons
- Merge (M) Merges vertices which share the same Cartesian coordinates
- Polygon (P) – Forms a polygon from selected vertices, manipulated by the order in which they were selected.

Here is a link to a website detailing all tools and shortcuts:


Additional Info

- A user can preload a generated model and construct next to it in order to gain a sense of scale when modelling.
There is a process known as re-topology whereby a pre constructed model from another designer can be used and tweaked to fit a design of custom choice.

A user can add smaller themed models called greebles in random sections of a model in order to create a complex detailed effect. Such examples include models of space vessels or machinery. Below is a Greebled cube from the Star Trek Series constructed in LightWave.


There is a process known as transfer mapping by which a user can transfer the image of a high polygonal model over a low polygonal one during a gamer to create detail quickly.

Another aspect to 3D construction involves the use of textures. If editing an environment individual polygons can be textured individually across the vertices but with objects the entire 3D model can be textured with one customized image.

This is achieved by flattening the UV map with applications such as Roadkill whereby a user can cuts across edges.
With this UV map generated a user can use applications such as Xnormal to generate various texture maps such as Ambient Occlusion, Specularity and Wire frame. These maps allow a user to define effects more easily such as glow, 3D illusions within textures and shininess.
Part 4 - Constraints of producing 3D models for computer games

In the modern Videogame industry there are standards and constraints that a 3D artist must abide by in order to ensure that any 3D models are compatible with contemporary computer systems. This is due to the fact that if these ‘Game Legal’ laws are broken, a current computer may struggle to process the design correctly and may result in glitches or image malfunctions. These constraints continue across all aspects of 3D development including not only the modelling process, but also the texture details and animation configurations. The following section details these constraints within the specific areas of the 3D construction process:

**Constraints within Modelling**

A model must not consist of any other polygon other than Triangles and Quad Point Polygons. Any polygon that sustains only 1 or 2 vertices is considered a ‘Diagon’ and any that consist of more a known as N-Gons with the ‘N’ meaning numerous. These types of polygons cause complications within a game engine due to the massive amounts of computation required in order to simulate that face dynamically. An artist must be consistently aware of any sections of a model that fall into these categories as simple primitives such as discs automatically include N-Gon surfaces.

Additionally the term ‘Legacy’ is used when considering any unintended polygon left behind from tools such as Extrude. These additional polygons cause issues also within light and animation aspects.

Another issue with geometry in the games industry are any models which normals cross. No polygon should cross through another when considering the edges of a shape such as this cube to the left. The same also applies to vertices resting on the surface normal or edge of a polygon. This issue creates excessive malfunctions within regards to lighting and character animation.

All aspects of any model unintended for animating must be meshed together in order to create a single whole. This is largely due to a games engine only being able to maintain geometric shapes as individual elements and struggles to process unconnected meshes unless for intended for animating the model, similar to the way that animations software treats layers as individual objects.
There are other restraints that may be company or client specific despite not having an impact on whether the model is computer compatible. This is usually due to rendering power limitations or amount of content that will be put into the game. These include:

- Limitation on amount of polygons available for a model. This is a common restriction as elements within a game must balance the processing power of the given platform.

- Number and scale of models and polygons may not be allowed over or under a certain limit. When modellers and animators work together on joint projects the correct scale must be abided by in order to maintain consistency within a game environment as well as the development team. For example if a character model was twice the size of a building this could create instant issues. When considering Polygons count and individual scale, this may be due to a model being created in such a way that it can easily be manipulated at certain segments.

- File formats must be available in multiple format specifics such as OBJ, X file or LWO. This is often a standard during the development phase as a model may need to be transferred across alternative 3D development software.

- No spider webbing of 3D meshes whereby and over use of triangles are used in order to mesh an object together. Extensive use of edges in close proximity emanating from a given vertex can create light and computer calculation errors during rendering as well as being hard to manage during other aspects of development such as tweaking the models.

- No Concave/Convex polygons known in development as Non-Planar. Every Normal should be a completely flat surface to aid computer computation and lighting.
Texturing is another area in 3D development that requires specific ‘Game Legal’ specifications as their ability to be rendered by a computer is based on their resolution. An image texture or ‘diffuse’ map is only calculated by a computer correctly within a game engine if the numerical value regarding its texture size to the power of 2, such as an 8 x 8, 16 x 16 or 32 x 32 and so on square. The height may differ from the width but must always remain within this standard such as a rectangle at a resolution of 128 x 64 or 128 x 256. The examples below show this in correct scale.
Bibliography

http://fantasyartdesign.com/free-wallpapers/imgs/new/43science-fiction04.jpg

http://www.realitytwist.com/images/illus/compos/Sum_clo_witch.jpg

http://en.wikipedia.org/wiki/Game_development

http://pcmedia.ign.com/pc/image/article/101/1012767/command-conquer-4-20090810112303153_640w.jpg

http://www.episcopalacademy.org/drum/MSWebDesign/INTESAR%20JANJUA%EF%80%A8/Company%20folder/halo-reach2.jpg


http://en.wikipedia.org/wiki/Gouraud_shading

http://wiki.beyondunreal.com/Legacy:Polygon


http://www.downtr.net/154520-udk-unreal-development-kit-2010-build-6383-.html