# Topic 2

Graphics pipeline

How to turn data into pictures

All the models are vertices which join up to make polys which join to make the shape

Viewing

This is mapping the vertices from different coordinate systems

Moving data into different spaces to see how it looks different

Types of projection

Orthographic

There is no depth

Used for the different view options

Simple to implement not realistic

Perspective

Makes things look further away

All calculated relative to the point of view

When using this it is realistic as it makes objects that are further away appear smaller

Clipping window/ viewing volume

Anything outside of a clipping window will not be rendered

Frustum

Anything to far or to close will be ignored

Frustum clipping/ culling

Removing unseen elements for optimisation

Only process and render what is in the frustum and viewable

Allows for faster rendering

**Coordinate systems**

Process of taking data and turning it into a picture

Things in the scene need to be described relative to a space and coordinate system

Matrix transformations

Coordinates are transformed between each coordinate systems

**Local coordinate space**

Where the object is defined local to an object

Allows for an object to be defined and reused

**World/global coordinate space**

All objects are transformed into one coordinate system

**Lighting and camera**

Instances of objects can be positioned in world space

Instance will use the same data with its own position, rotation and scale

**View space**

Associated with the camera and defined by the normal

Various camera parameters defined by position, view FRUSTUM(clipping/ culling occurs here)

Also back surface removal

**3D screen space**

Maps the view frustum into a uniform space usually 1 by 1 by 1

Prepares the shape to be displayed on a screen

**Rasterization** is determining what pixels are shown and rendered

Hidden surface removal and determines what is at the front

**Display space**

Coverts 3D screen space into coordinates for the screen

Makes it a 2D image to be displayed on the screen

The movement of the data through the graphics pipeline is what takes the 3D objects into a 2D picture on the screen.

**Graphics system standards**

There are standardised ways to implement graphics in this way

More public access

GKS 1984 graphics Kernel system

PHIGS programmer’s hierarchical interactive graphic standard

GL graphics library

Open GL is open source for everyone to use (Microsoft products)

Direct 3D render graphics can also be used for audio

In view and screen reference system x if left and right, y is up and down, and z is forward and backward

World reference system x is left and right, y is forward and backward, and z is up and down

# Topic 5

Polys are flat sides

There are ways to make shapes look smoother without using more polys

Using the light it can bounce off the object at certain angles to make an object appear smoother

This is calculated with the angle and the normal

**Lamberts cosine**

When facing the light 100% brightness

When facing away 0% brightness

Calculated the colour of the pixel using the normal

This is the cheapest and best way to smooth an object however it is not always ideal

**Normal maps**

This is used to work out lighting

Normal are used to calculate what angle the light is hitting the surface at

Normal maps are used to add detail or fake depth without increasing the poly count

Pixels represent xyz instead of RGB and discerns how light reflects off it

**Smoothest solution**

You can use curvilinear quadrilaterals

When the edges of the poly are cubic curves

Using cubic equations, you can create accurately smooth curves

There are 2 common ways of representing curves

**Parametric**

A function per axis with a parameter passed to the function

**Surfaces**

2 parameters allow for various curves to combine into a surface#

Turning the curve into a 3D surface

**Implicit**

When the vertices are not directly or explicitly provided

Instead, the points are implied

**Polynomials**

Polynomials means lots of numbers

Combining polynomials, you get a curve

These curves can be combined to get a new curve

Cubic curves are more controllable

Polynomials can be blended to make cubic curves which can be combined to make the final shape more controllable

**Bernstein cubic curves**

Change the weight (impact) a curve has on the final shape

**Bezier curves**

They can be combined to make more complex shapes

They can also be subdividing

Surface

Multiple Beziers create a surface with more control and a smoother surface

Still high poly

Also used to create smooth camera paths

More ways of generating smooth curves and surfaces:

**NURBS**

**Non-uniform Rational B-spline**

Allows for creation and manipulation of smooth topology

Can change the weighting of each point

**Spline**

A curve made up of multiple polynomial curves that have been combined

**b-spline**

a flexible spline with lots of control points

split into ranges referred to as Knots

there can be sub curves inside the ranges (Knots)

all about combining curves to have better control

this takes multiple equations for curves and combines them to make one large one

sub-curves can overlap

ranges can be specified using Knot vectors

Non-Uniform is when the knots are changed to change the final curve

**Rational**

Rational is the ration of 2 spline functions

A second spline function us used to determine which curve has the bigger influence using the weighting.

**Beziers vs NURBS**

Beziers cannot be perfect circles whereas NURBS can make perfectly smooth surfaces.