

# INFT3960 – Game Production

**Week 11**

**Module 11.1**

**Graphics Pipeline**

# Course Overview

Lec	Start Week	Modules	Topics	Assignments
1	3 Aug	Mod 1.1, 1.2	Course Overview, Design Process	
2	10 Aug	Mod 2.1, 2.2, 2.3, 2.4	Unity3D Introduction, Introduction C#, Variables and Components, Hello World	
3	17 Aug	Mod 3.1, 3.2, 3.3	Booleans, Loops, Lists and Arrays	Assign 1 21 Aug, 11:00 pm
4	24 Aug	Mod 4.1, 4.2	Functions and Parameters, Debugging	
5	31 Aug	Mod 5.1, 5.2	Classes, Object Oriented	
6	7 Sep	Mod 6.1, 6.2, 6.3	Agile Processes, Risks and Prototypes, Testing	
7	14 Sep	Mod 7.1, 7.2	Puzzles, Guiding the Player	Assign 2 18 Sep, 11:00 pm
8	21 Sep	Mod 8.1	Game Physics	
9	12 Sep	Mod 9.1	AI for Games	
10	19 Oct	Mod 10.1, 10.2	Game Interface, Storytelling in Games	
11	26 Oct	Mod 11.1, 11.2	Graphics Pipeline, Animation in Games	Assign 3 1 Nov, 11:00pm
12	2 Nov	Mod 12.1, 12.2	Networked Games, Course Review	

## Course Details

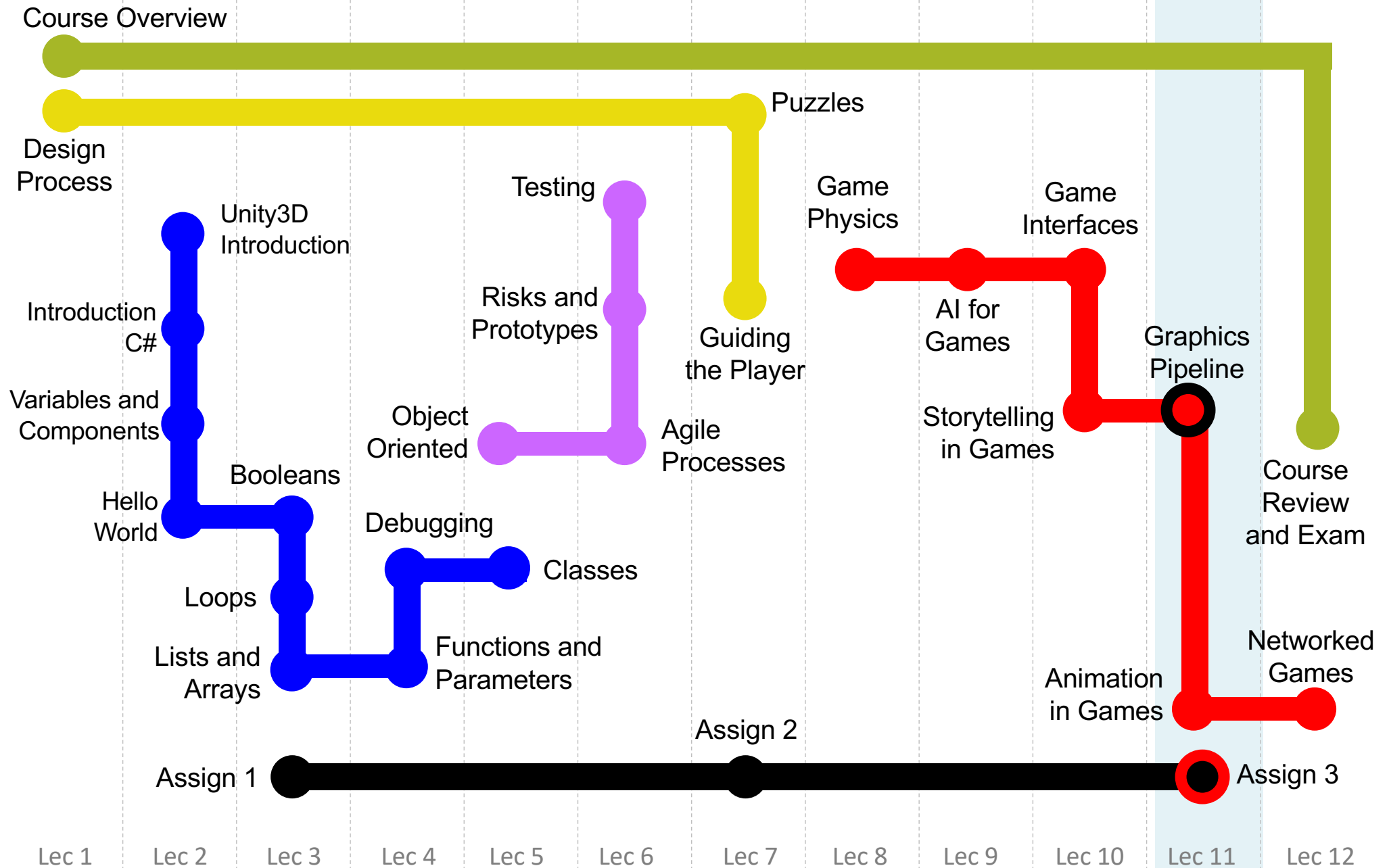
## Game Design

## Unity 3D and C#


## Development Process

## Core Game Concepts

## Assignments



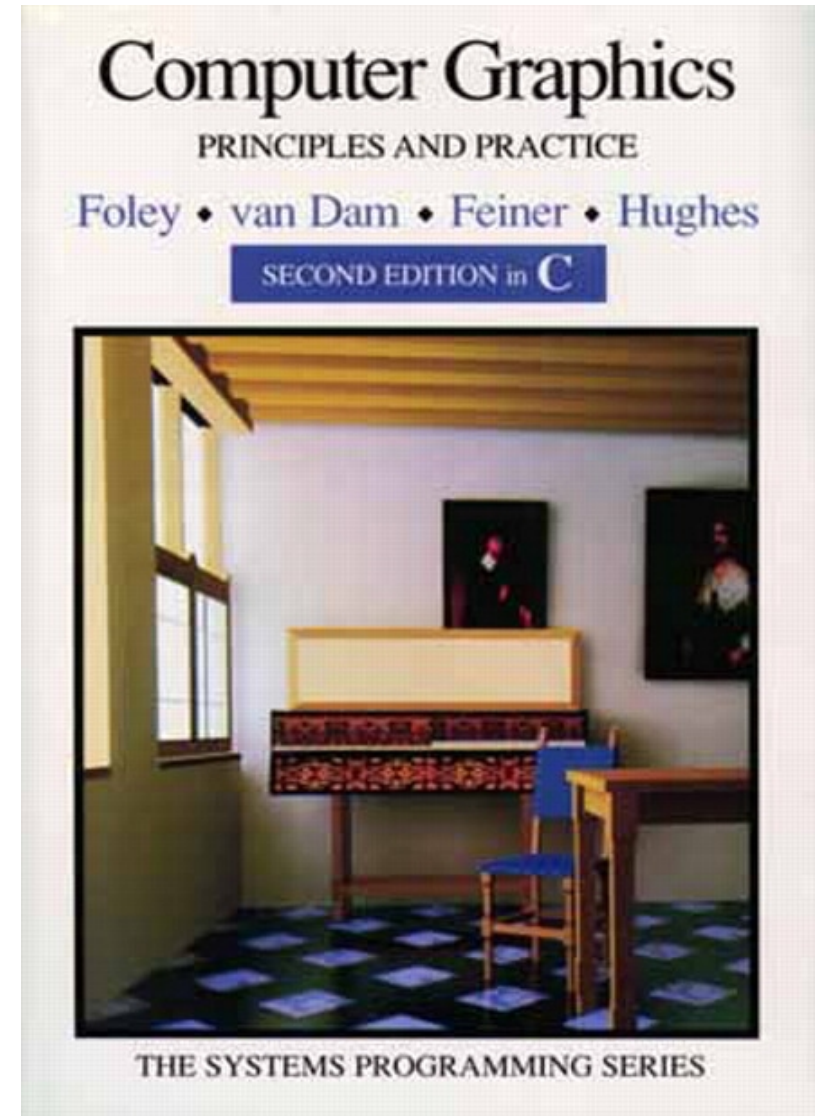
# Graphics Pipeline – Topics

- 
- The Graphics Pipeline
  - Model Transformation
  - Lighting
  - View Transformation
  - Projection Transformation
  - Clipping
  - Texturing
  - Rasterisation
  - Display

# Useful Reference

Foley, van Dam, Feiner, Hughes  
Computer Graphics: Principles and  
Practice in C (1996). Addison-  
Wesley Professional

A fundamental and very technical text  
book on the foundations of computer  
graphics - often referred to as the  
encyclopaedia of computer graphics.



# Game Engine

Some jobs for the Game Engine...

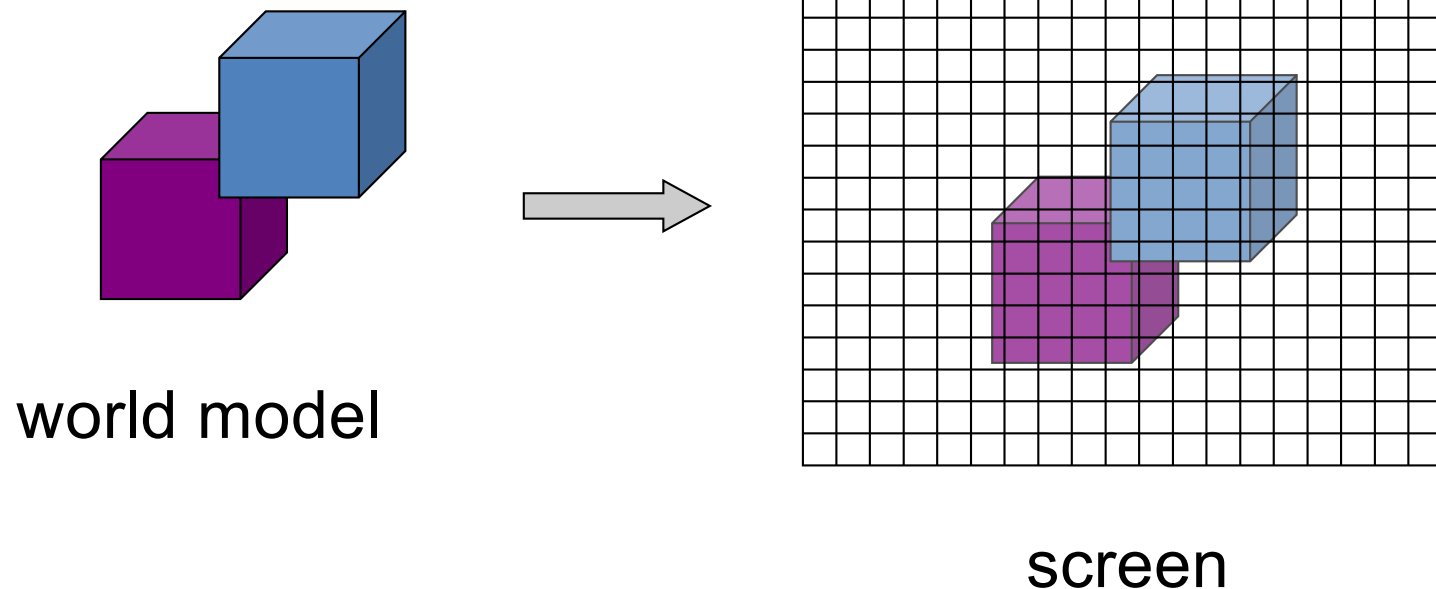
<b>Rendering Graphics</b> Sound Rendering Calculating Physics Collision Detection	User Interaction Event Processing Managing the World	Storytelling Animation Artificial Intelligence Networking
--	--	--

The key steps for graphics rendering are implemented in the **graphics pipeline**.

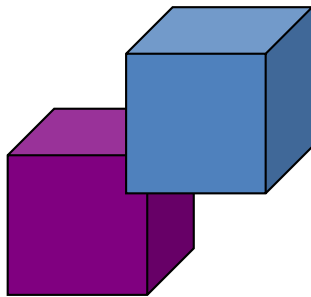
You don't need to understand this pipeline in detail - but it should help you understand how and why game worlds are modeled in the way they are.

# Rendering Graphics

Creating a digital (raster) image from a model



# Rendering Graphics



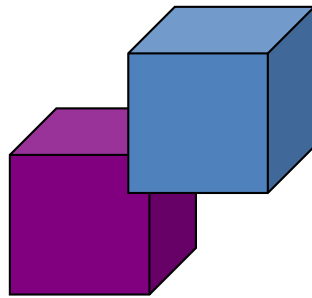
world model

The model contains a description of objects and properties such as geometry, transformations, lighting, shading and the viewpoint (camera).

The model is often stored as a "scene graph".

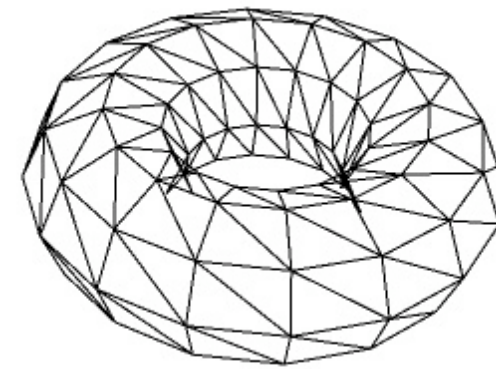
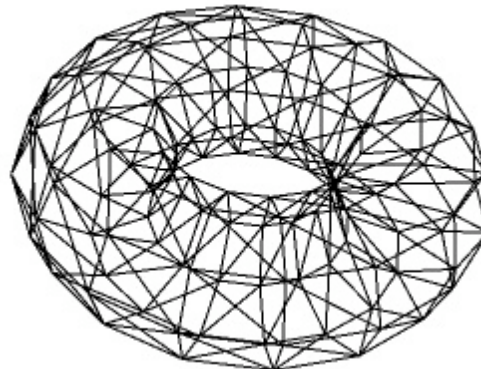


# Rendering Graphics

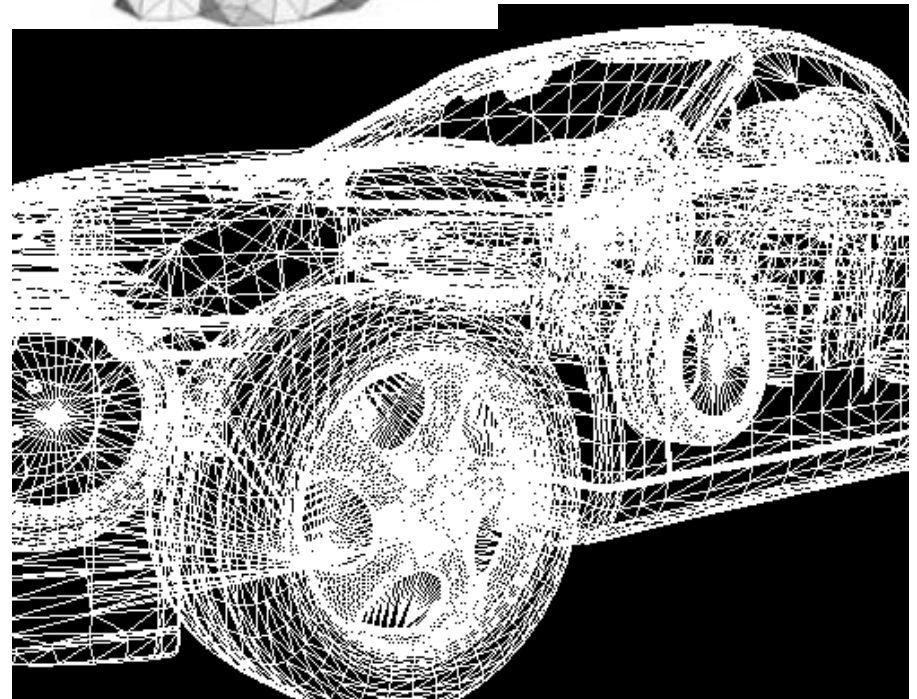
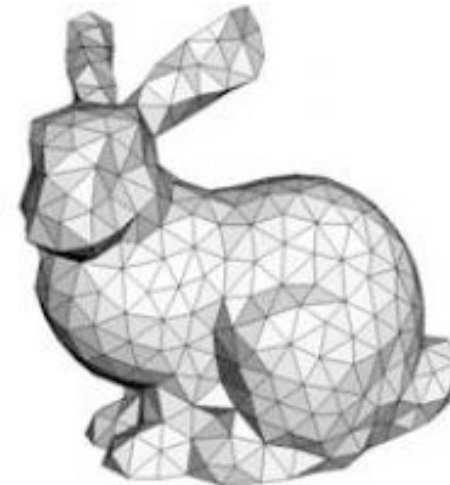
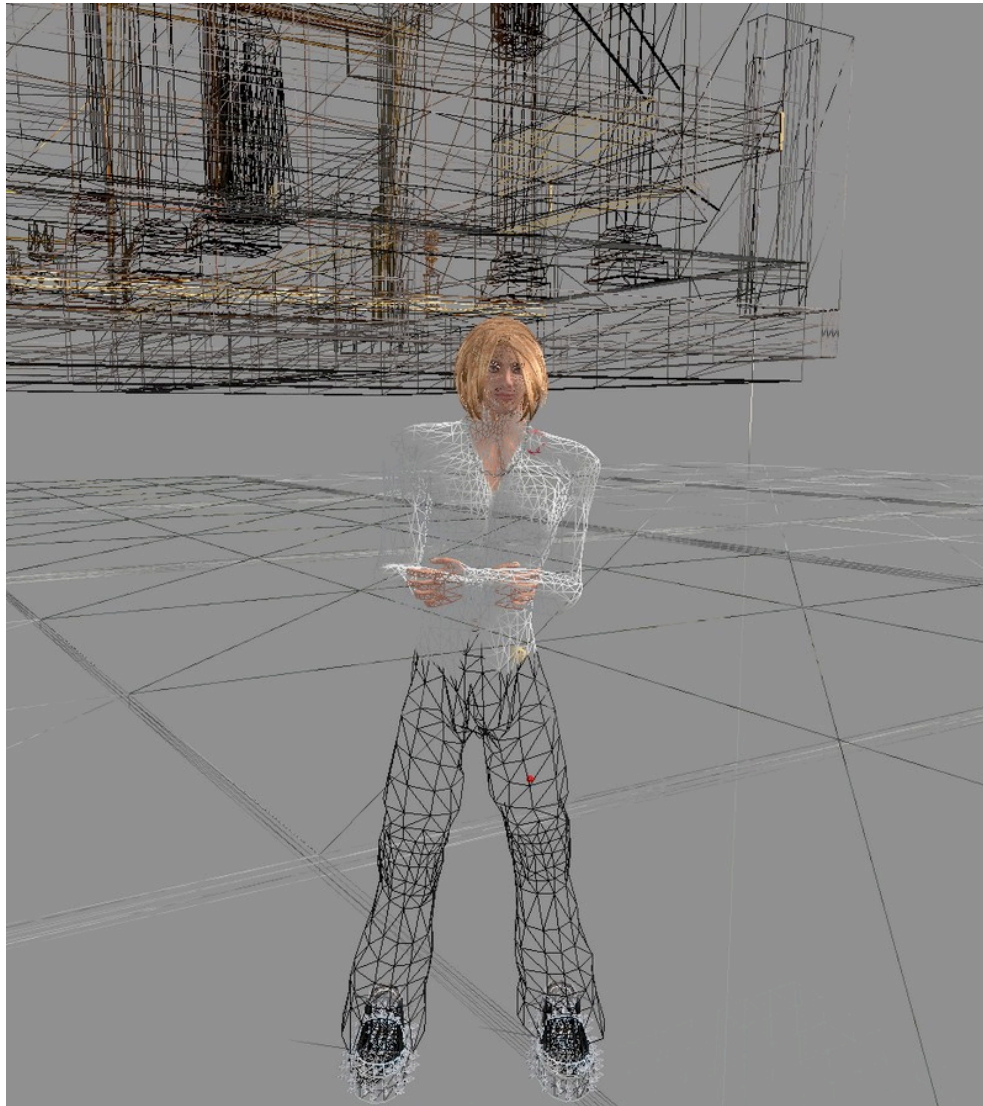


world model

Geometric primitives		other parts
2D or 3D	points lines and line segments planes circles and ellipses <b>triangles and other polygons</b> spline curves, ...	<i>text</i> <i>images and sprites</i> <i>animations</i>
3D	cube cylinder sphere pyramid cone, ....	



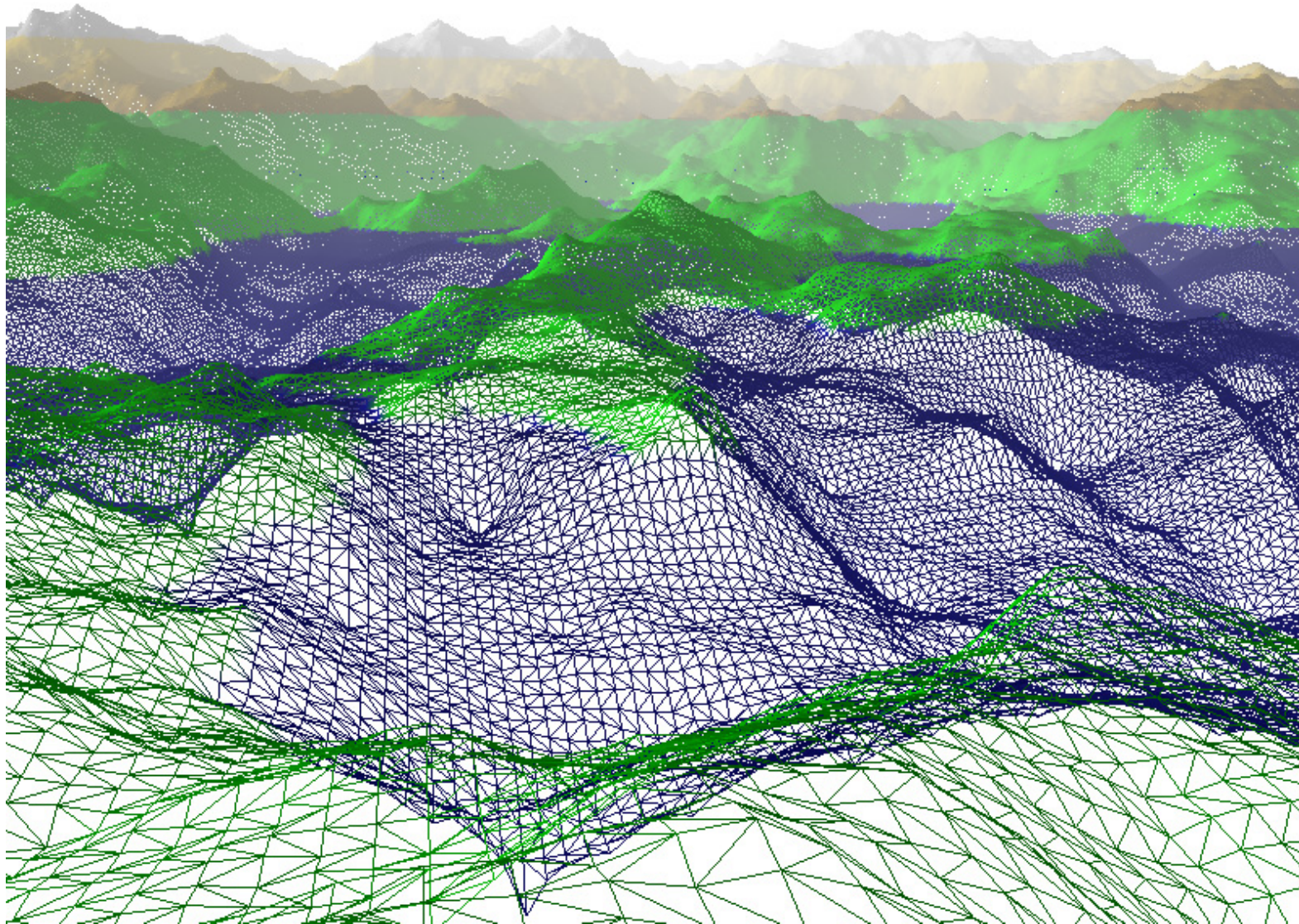
# Triangle Models



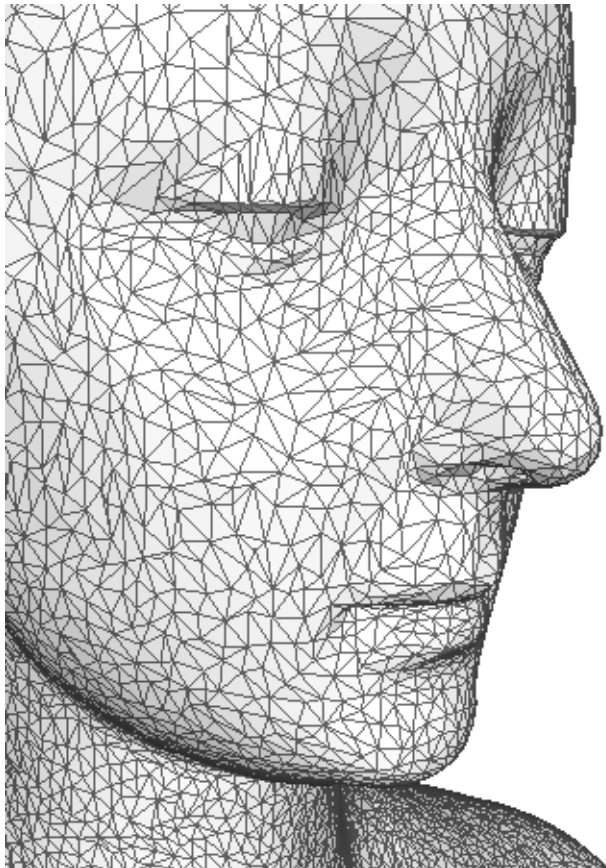


# Triangle Terrain

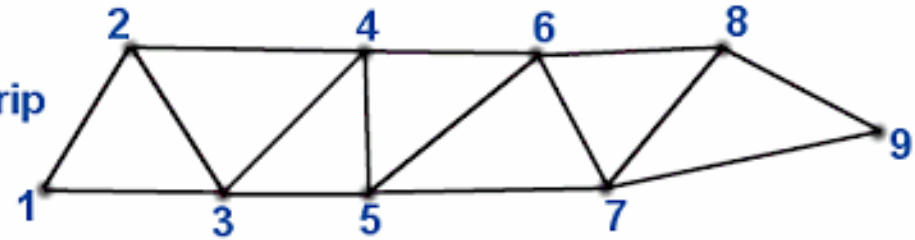
5 fps



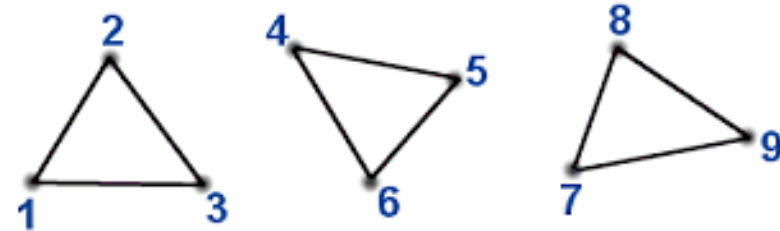
# Triangle Data



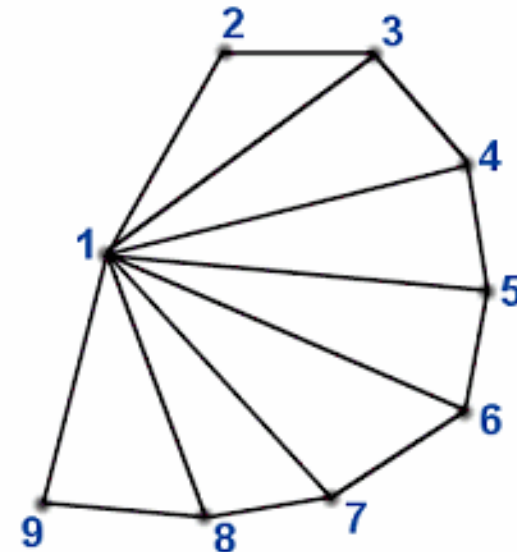
Triangle Strip



Triangle List



Triangle Fan





# Graphics Pipeline

A key to fast rendering is the *"Graphics Pipeline"*.

Dedicated hardware is sometimes used for this and called the GPU (Graphics Processing Unit)

The different steps in the pipeline are implemented as separate processes which can be running in parallel (at the same time).

# Graphics Pipeline

A key to fast rendering is the *"Graphics Pipeline"*.

Dedicated hardware is sometimes used for this and called the GPU (Graphics Processing Unit)

## **GPU vs CPU**

GPU – Graphics Processing Unit

CPU – Central Processing Unit

<https://www.youtube.com/watch?v=ZrJeYFxpUyQ>

ented  
in

# Graphics Pipeline

It's a bit like a factory assembly line - the aim is to keep all the parts of the factory working together to produce the final product - the screen image.



# Parallel Processing



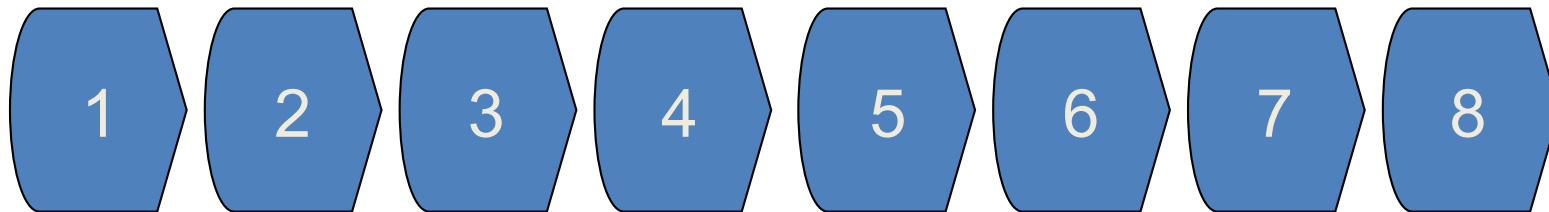
At the beginning of processing only the first steps of the pipeline will be active.



# Parallel Processing



At the beginning of processing only the first steps of the pipeline will be active.

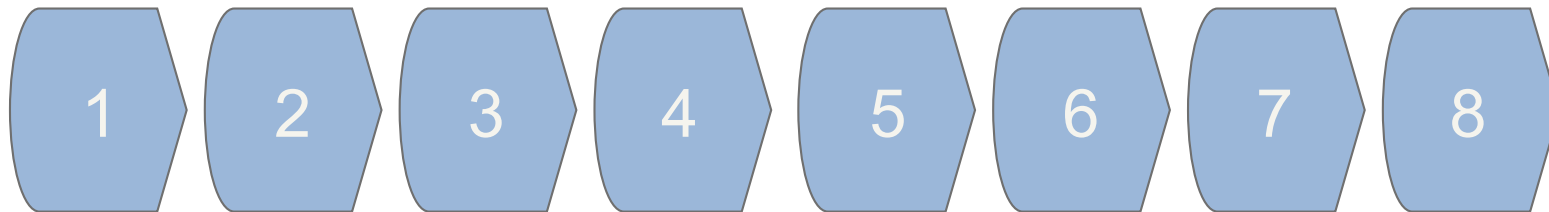


For most of the processing all the steps will be active (working in parallel).

# Parallel Processing



At the beginning of processing only the first steps of the pipeline will be active.



For most of the processing all the steps will be active (working in parallel).



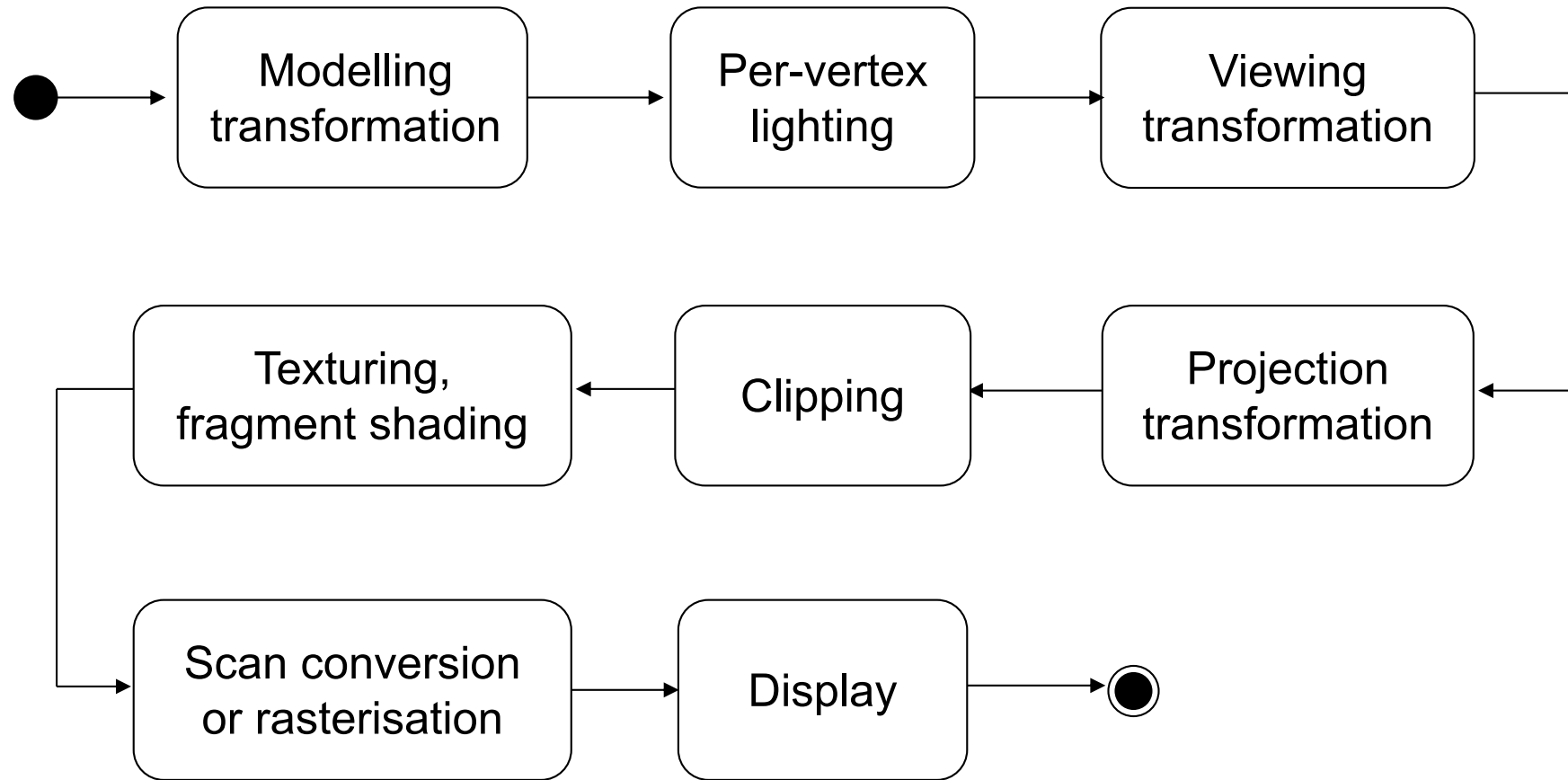
As the processing concludes only the last steps of the pipeline will be active.

# 2D vs 3D

The process of rendering is simpler in 2D graphics (compared to 3D) - especially if the model is mainly bitmaps (pictures) and animations that need to be displayed on the screen in ordered layers.

When vector graphics are used to represent the 2D world model the pipeline is much the same as for 3D graphics (although simplified because only two dimensions are required for calculations)

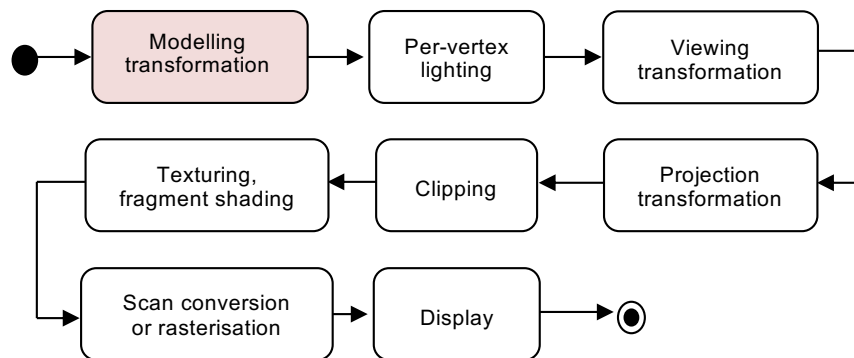
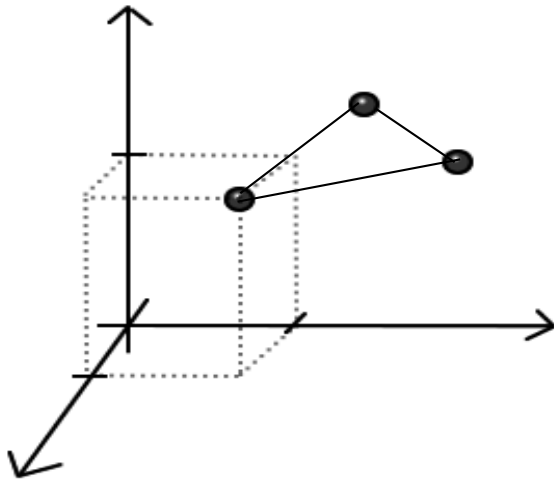
# Fixed Graphics Pipeline



[http://en.wikipedia.org/wiki/Graphics\\_pipeline](http://en.wikipedia.org/wiki/Graphics_pipeline)

# Modeling Transformation

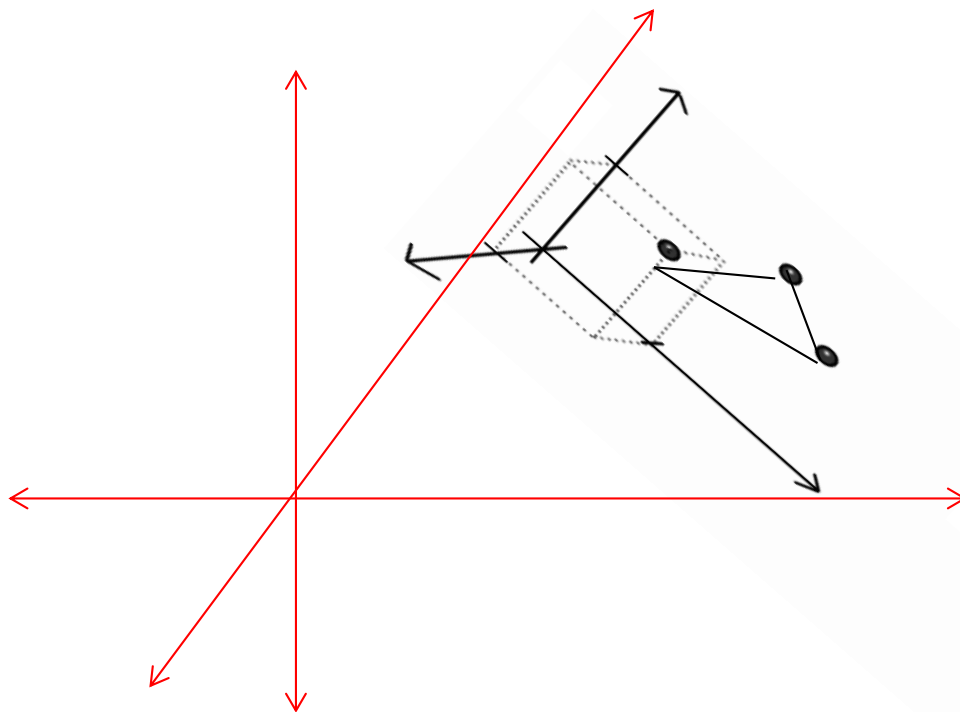
The geometry (2D or 3D) described in *geometric primitives* in their own space (object space)



Geometric primitives	
2D or 3D	points lines and line segments planes circles and ellipses <b>triangles and other polygons</b> spline curves, ...
3D	cube cylinder sphere pyramid cone, ....

# Modeling Transformation

These are transformed into the world space (2D or 3D). This may involve operations such as translation, scaling and rotation.



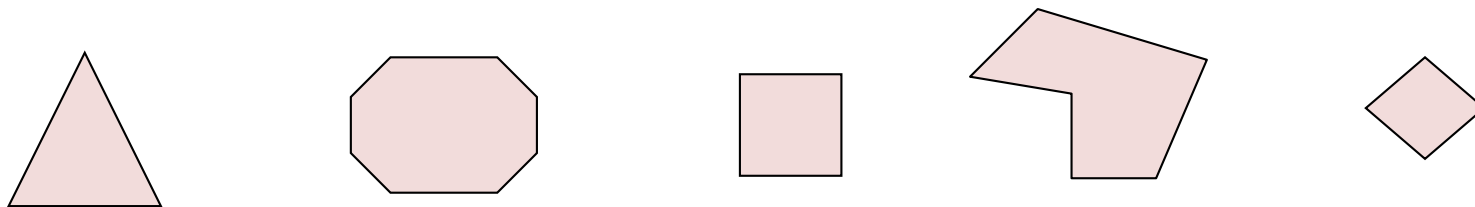
Geometric primitives	
2D or 3D	points lines and line segments planes circles and ellipses <b>triangles and other polygons</b> spline curves, ...
3D	cube cylinder sphere pyramid cone, ....

# Modeling Transformation

Polygons are one of the most important geometric primitives used for modelling (in both 2D and 3D)

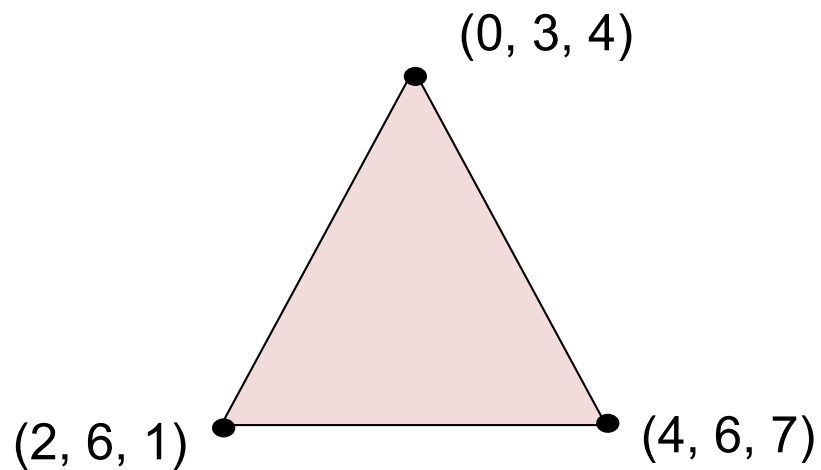
Polygons are enclosed shapes made of straight lines with more than 3 sides. (They lie in a single plane - are flat)

They are one of the most important modelling primitives in computer graphics (especially quadrangles and triangles).



triangles and other polygons

# Modeling Transformation



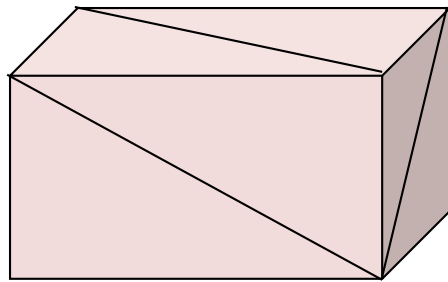
3D triangle  
(a simple polygon)

Polygons can be described by an ordered set of points (vertices).

e.g clockwise - this 3D triangle is described by three vertices (0,3,4), (4,6,7), (2,6,1)



# Modeling Transformation

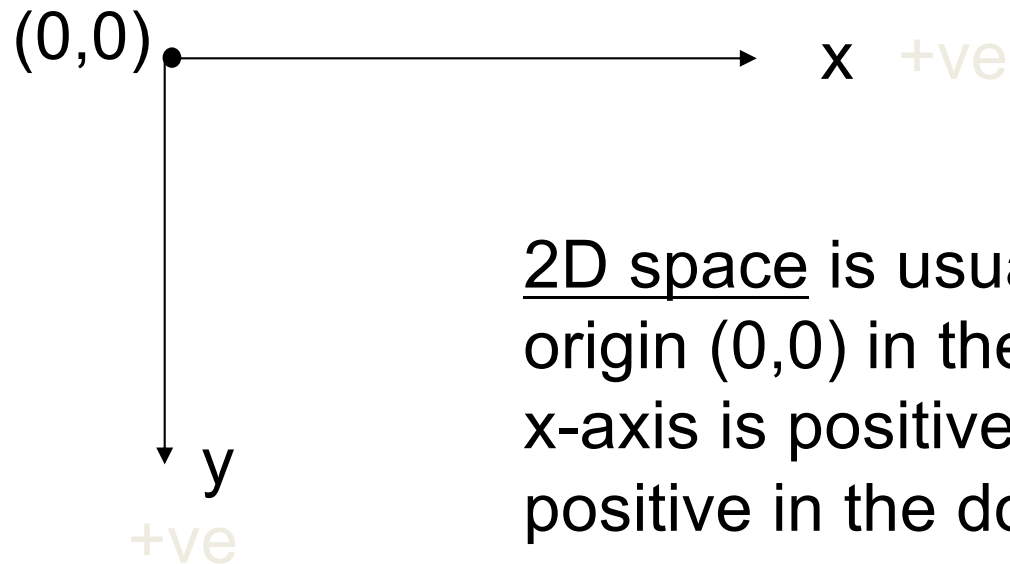


3D object  
made of  
triangles

More complex objects can be modelled from a number of polygons. Often triangles are used as modern graphics hardware is optimised to work quickly with these polygons.

A common challenge for the modeller is to represent objects with the minimal number of polygons.

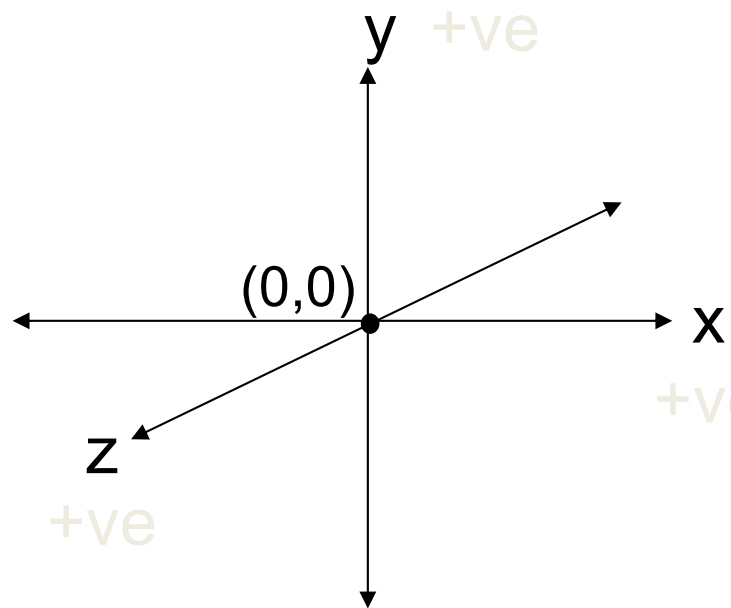
# Modeling Transformation



2D space is usually described with the origin  $(0,0)$  in the top left hand corner. The x-axis is positive to the left. The y-axis is positive in the downward direction.

*The way 2D space is defined in graphics is a slightly different then most other applications. The y-axis runs in a positive direction down the screen. This also matches the way pixels on a screen are usually defined.*

# Modeling Transformation



3D space is usually described with the origin (0,0) in the centre. The x-axis is positive to the right. The y-axis is positive in the upward direction. The z axis is positive in the direction away from the screen.

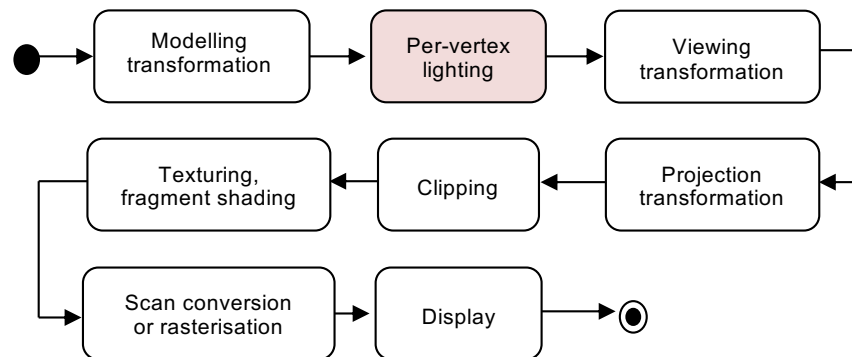
+ve (sometimes the y and z axis are exchanged)

*Note that objects are quite often defined in their own 3D space with the origin centred at the middle of the object (object space). This makes it easy to build objects and then rotate and scale them. They can then be placed (transformed) into a larger world which is described by it's own set of 3D coordinates or so called world space.*

# Per-vertex Lighting

The geometry is described by a number of polygons made up of points (vertices).

The amount of light affecting the model at each vertex is calculated. This is also called "vertex shading".



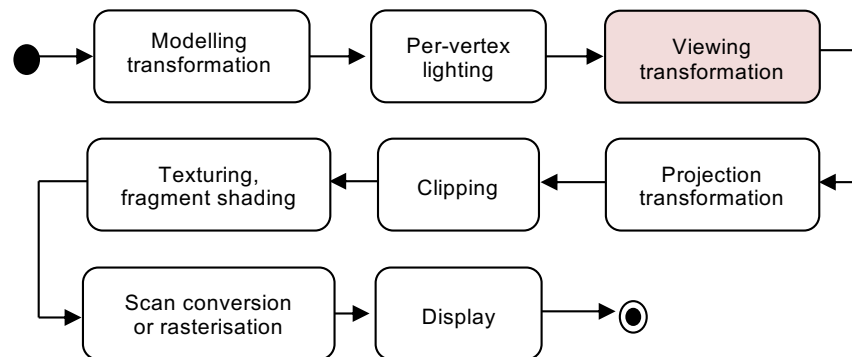
# Per-vertex Lighting

This step may adjust the position, colour and texture at each vertex by considering the types of lights and their placement in the world as well as the surface properties of objects such as their reflectance.

Often these lighting and shading calculations are simplified in the name of speed.

# Viewing Transformation

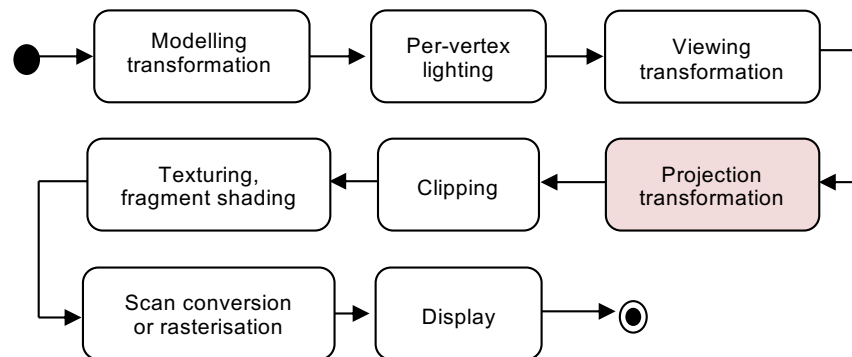
The viewing transformation changes the way the world space is seen by using a specific viewpoint. This is usually described as the view from a virtual camera. A change of viewpoint will require the visible objects in the scene to be recalculated, taking into consideration position and orientation of the "camera".



2D

# Projection Transformation

For a 2D model there is typically not much to do as the scene is already represented on a 2D plane.

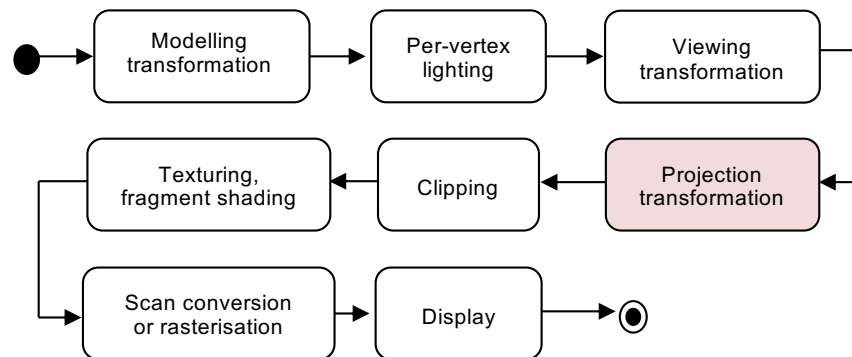


3D

# Projection Transformation

In a 3D model the view from the camera is still represented in a 3D scene.

In this step the view is projected into a 2D image plane to prepare it for display - using either a parallel or perspective transformation.



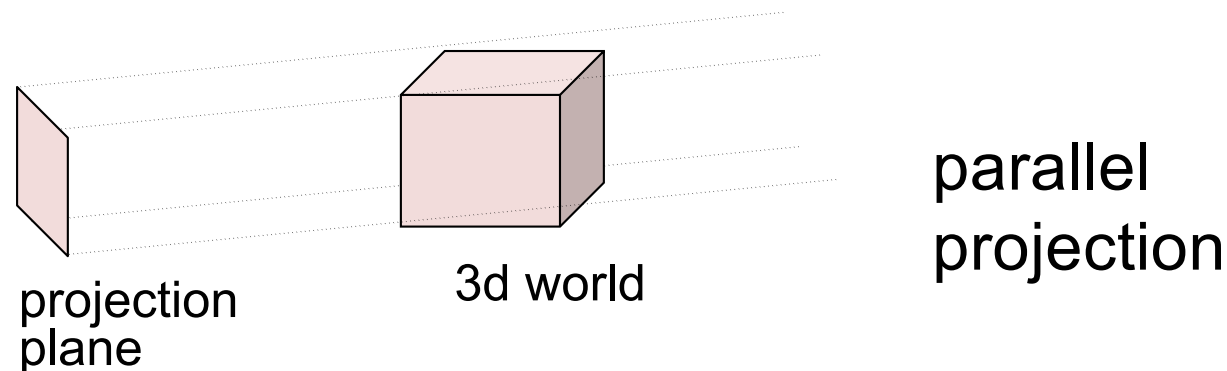
parallel projections  
perspective projections



3D

# Projection Transformation

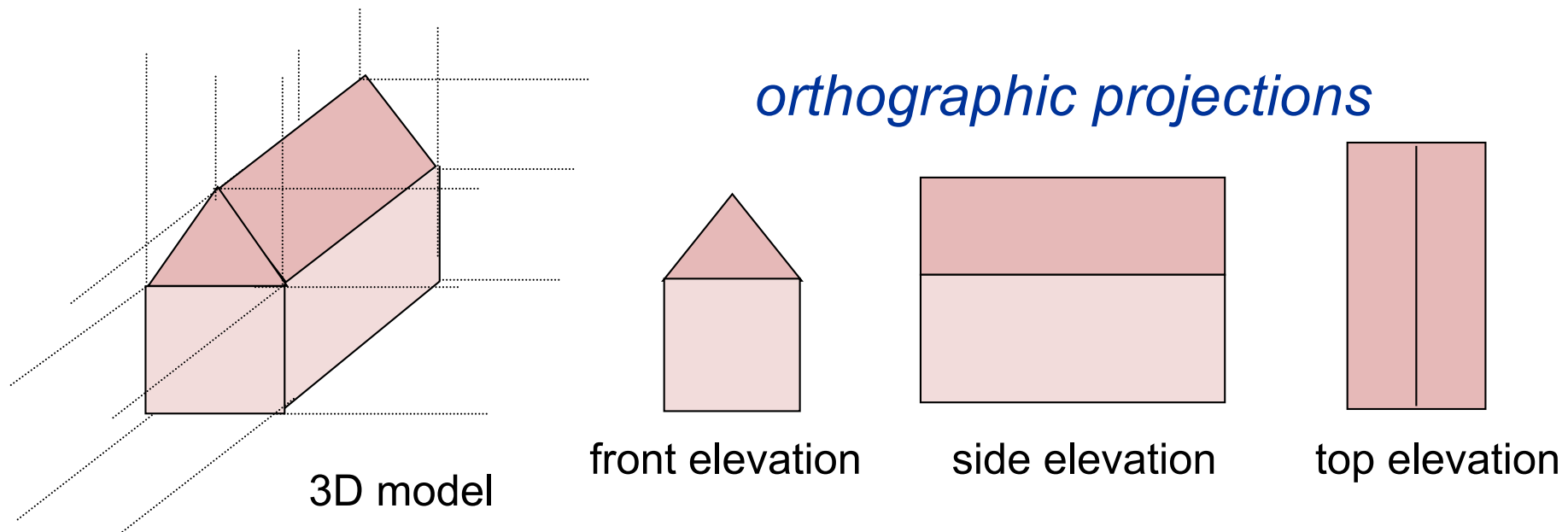
The most common *parallel projections* are those aligned with the x-axis (side-elevation), y-axis (front-elevation) and z-axis (top elevation) - these three projections are also called orthographic projections.



3D

# Projection Transformation

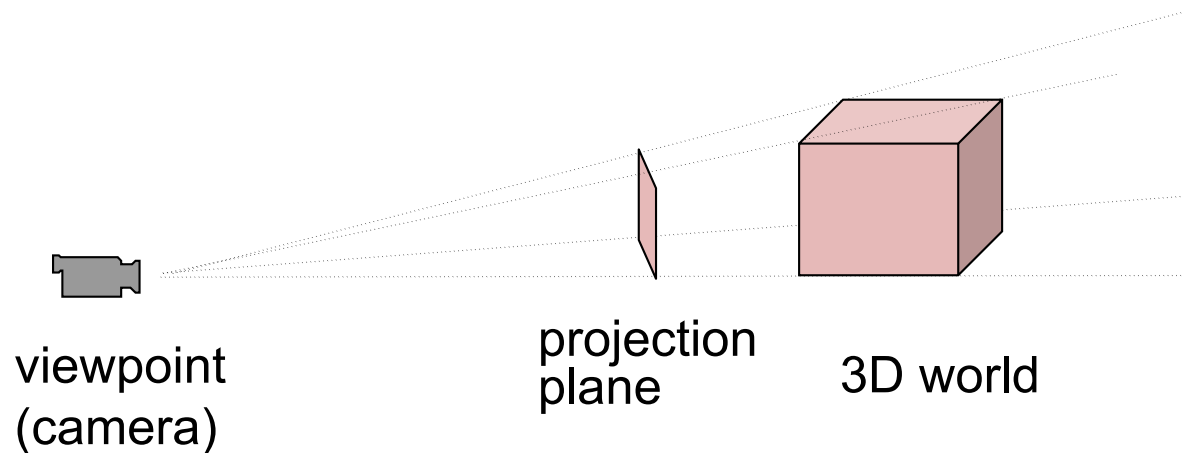
Parallel projections are often used when building 3D models because they ignore one dimension and so remove any distortion based on distance. Making it easier to judge sizes and shape in the model being built.



3D

# Projection Transformation

However, parallel projections ignore one dimension and hence remove depth information from the scene - this is important for judging the distance of objects - so they are not normally used to render a 3D game world itself - this is usually performed with a *perspective projection*.



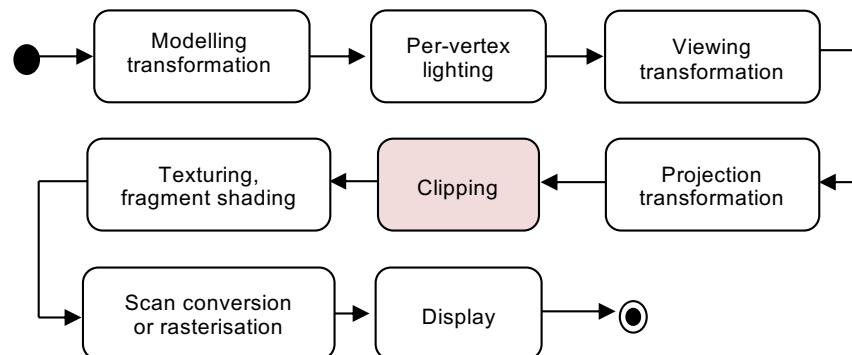
*perspective  
projection*

## 3D

# Clipping

In 2D graphics the clipping involves simply cutting out parts of the 2D model that will not be visible in the scene. (That is too far left or right or too high or low).

The view is clipped to a rectangle that bounds the screen.

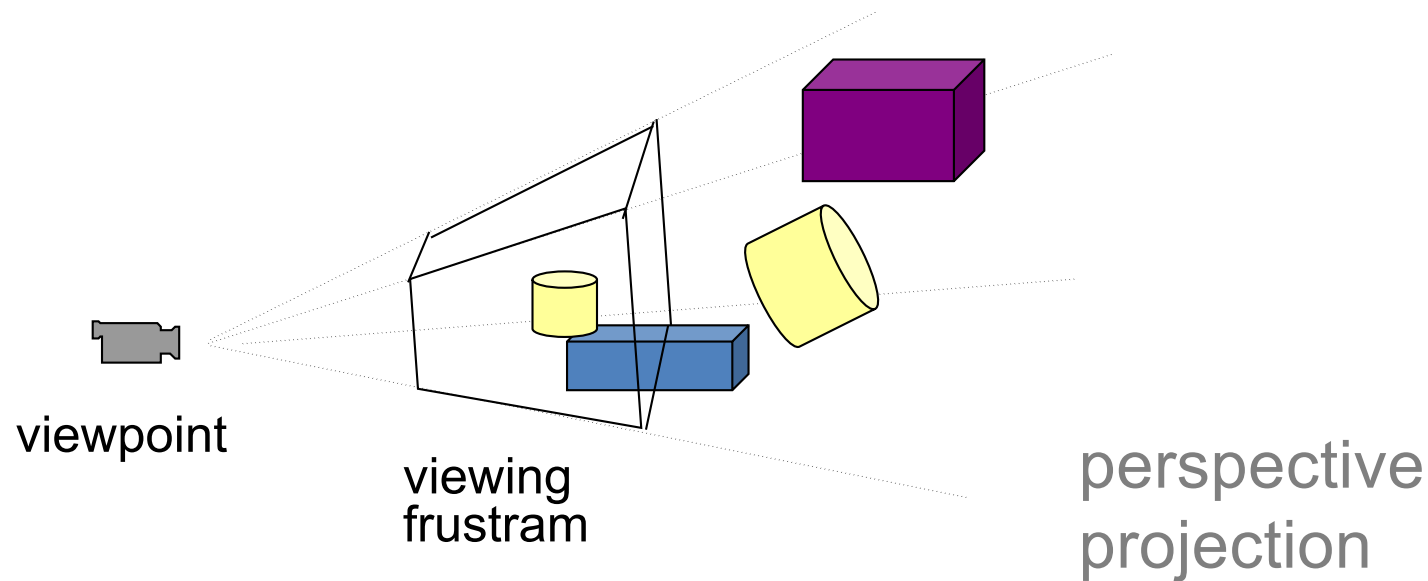


## 3D

# Clipping

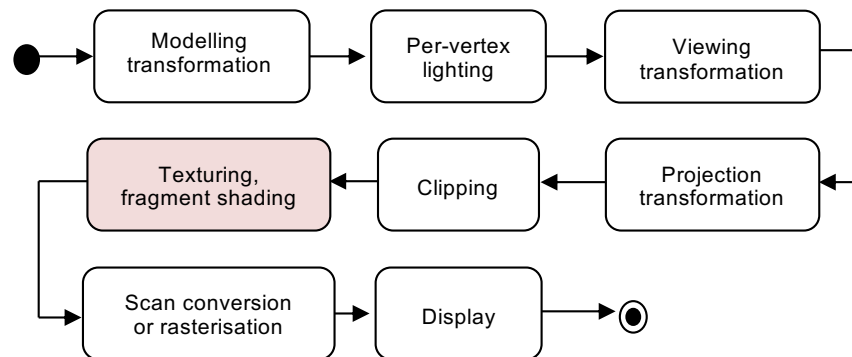
In 3D the clipping takes place in a volume (the **viewing frustum**) - objects to the left and right, top and bottom of the scene need to be clipped.

As do those objects that are near the camera or too far in the distance to impact on the scene.



# Texturing - Fragment Shading

In this stage the final shade at each pixel is calculated based on data such as the colour values at the vertices or from a texture in memory. Extra shading algorithms may also be applied.

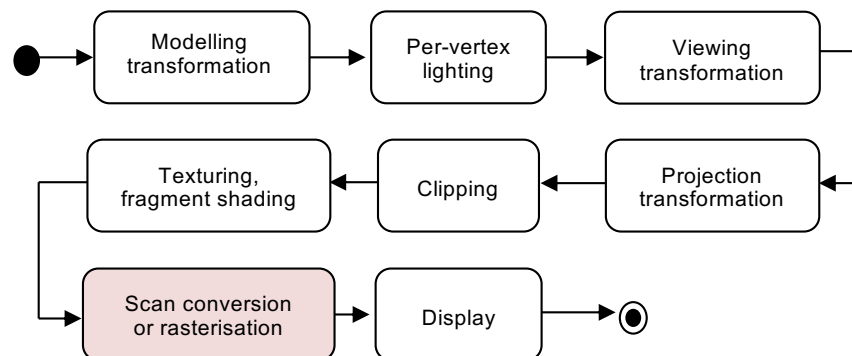


A fragment (pre-pixel) is a term used to describe all the data that is required to calculate the final colour (shade) of a pixel. It includes things like colour and texture, depth, raster position, ...

# Rasterisation

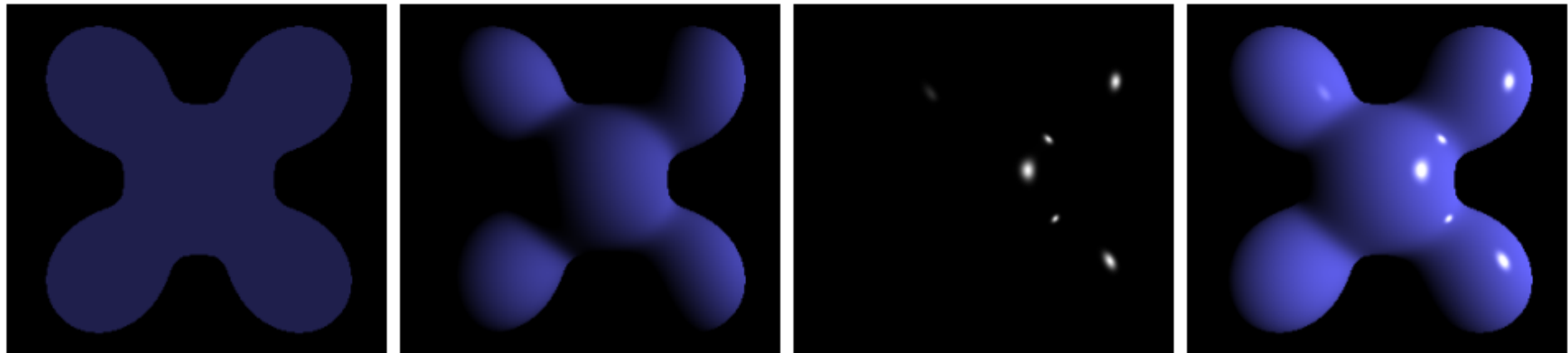
At this stage the 2D image (screen) has been calculated, and fragments have been calculated with a colour value.

During scan conversion the 2D image is converted to raster format and final pixel values are calculated in the framebuffer.



# Rasterisation

## Examples



**Ambient**

**+**

**Diffuse**

**+**

**Specular**

**=**

**Phong Reflection**

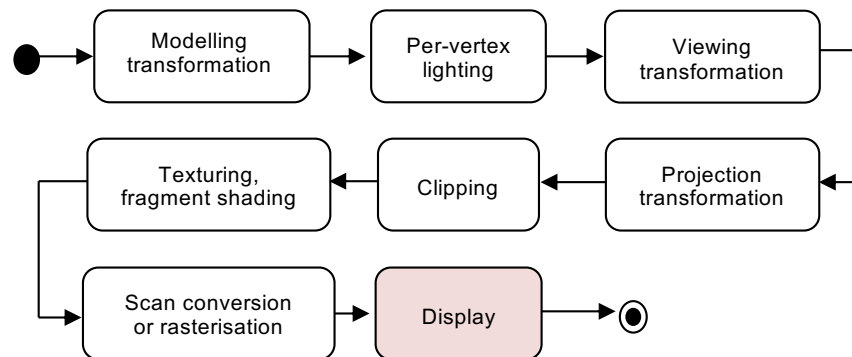
<http://www.adriancourreges.com/blog/2015/03/10/deus-ex-human-revolution-graphics-study/>



# Display

The final pixel values are displayed.

Usually this is simply a matter of moving the pixel values from a framebuffer to the screenbuffer.



# Display

