The Graphics Pipeline

CS2150

Anthony Jones

Introduction

- What is this lecture about?
  - The graphics pipeline as a whole
  - With examples from the video games industry
- Definition
  - The sequence of steps that are applied to a graphics primitive before it may be visually represented.
  - The graphics pipeline typically accepts some representation of a three-dimensional scene as an input and results in a 2D raster image as output.*

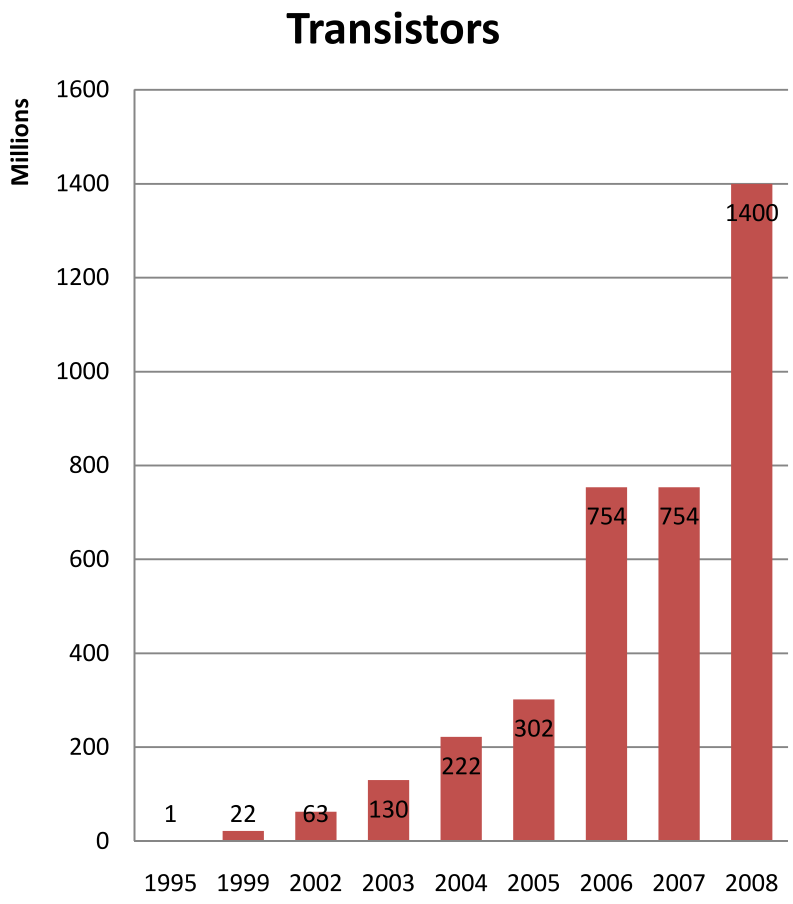
Introduction

- Games are significant drivers for current advances in computer graphics technology

- The global games market is worth £18bn and growing at 9% per annum*
- The video gaming industry is estimated to be worth £500m to the UK economy**
- Games made in the UK between 2006 and 2008 alone are on track to generate global revenues of £4bn*

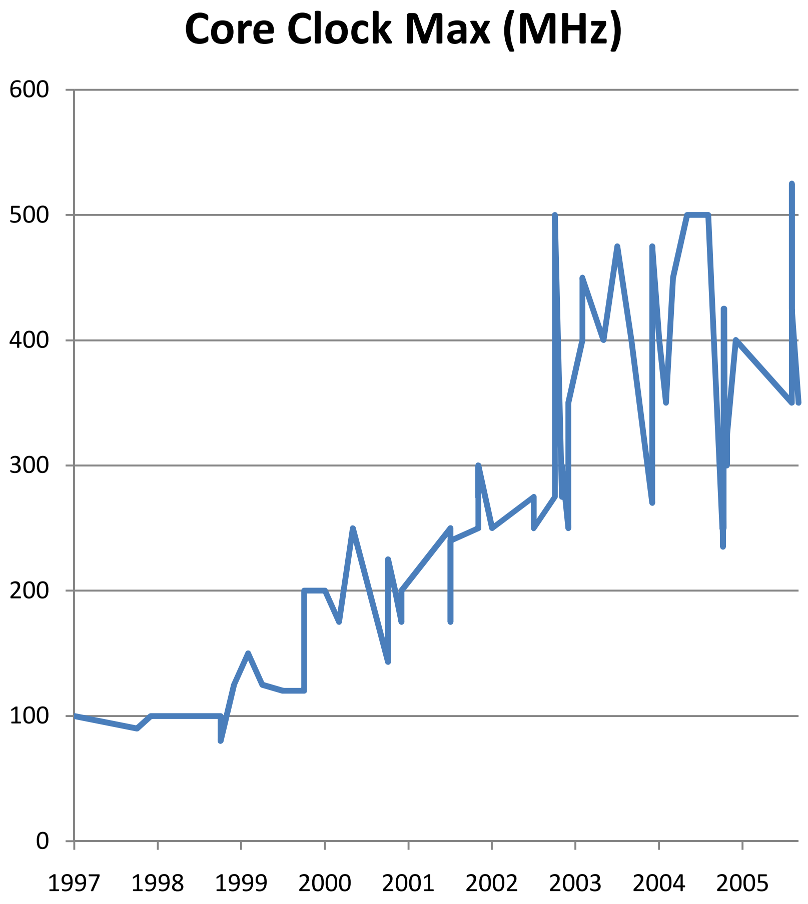
Introduction

- Games are significant drivers for current advances in computer graphics technology
- Increasing power of dedicated rendering hardware



Introduction

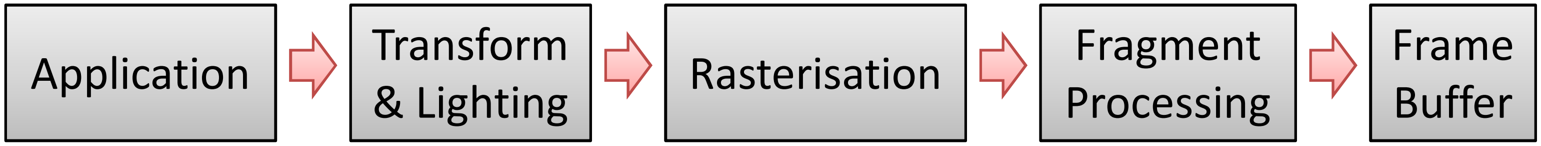
- Games are significant drivers for current advances in computer graphics technology
- Increasing power of dedicated rendering hardware



Video

- Examples of pre-dedicated hardware games
- Proprietary software renderers
  - Descent
  - Quake

Fixed Function Graphics Pipeline



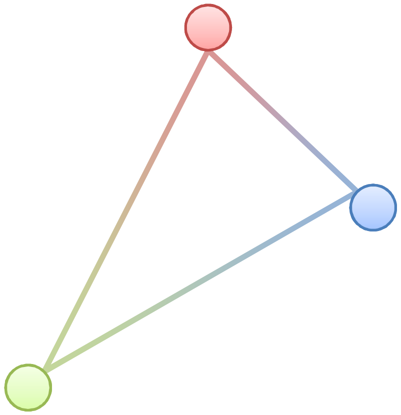
Application

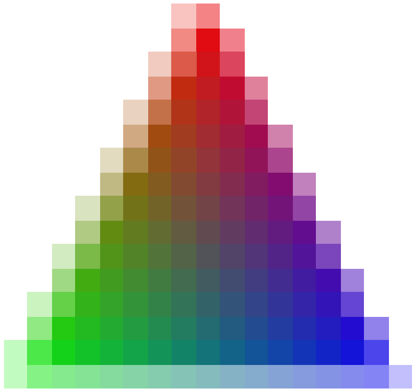
Transform & Lighting

Rasterisation

Fragment Processing

Frame Buffer





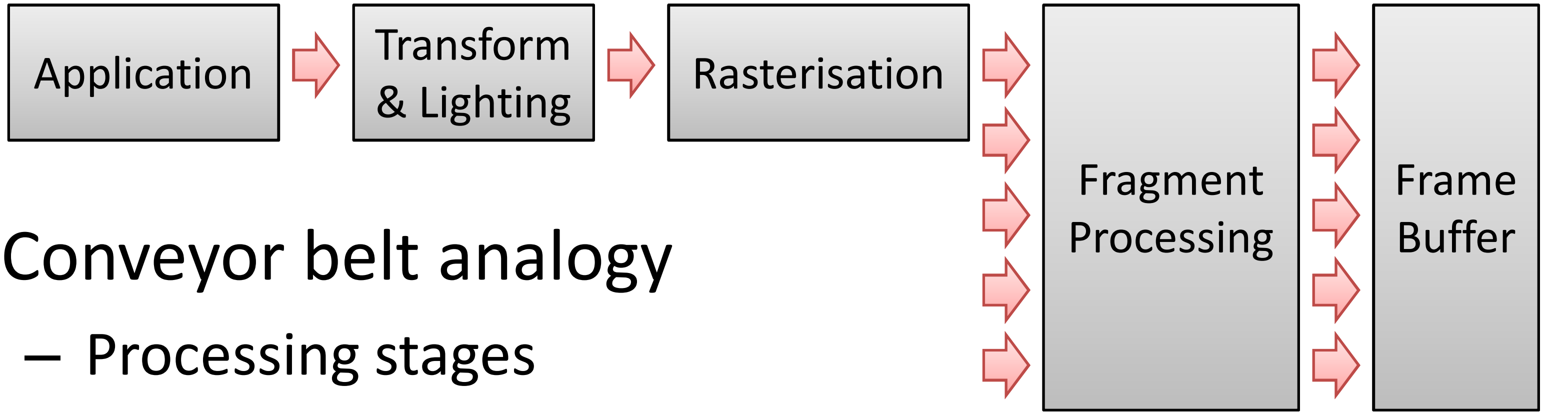
In:

- 3D scene information
- Geometry and attributes

Out:

- 2D raster image
- Pixel position and colour

Fixed Function Graphics Pipeline



Application

Transform & Lighting

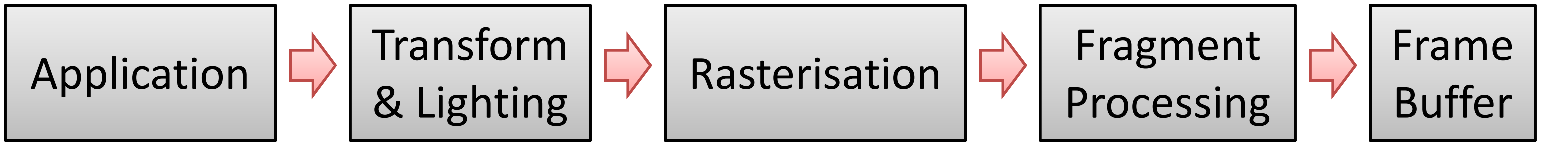
Rasterisation

Fragment Processing

Frame Buffer

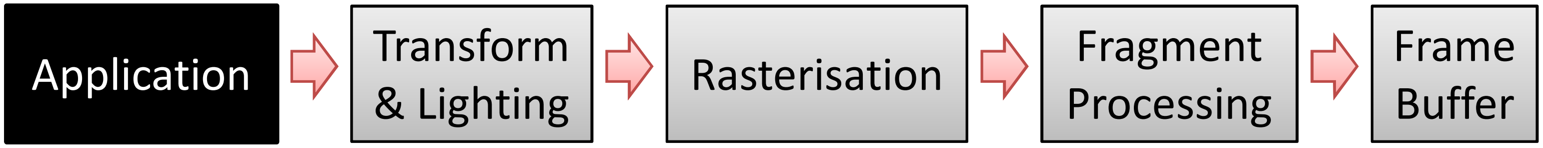
- Conveyor belt analogy
  - Processing stages
  - All stages benefit from parallel processing

Fixed Function Graphics Pipeline



Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Disclaimer
  - The position of certain operations in the pipeline (e.g. clipping and culling):
    - May not be consistent with what you may have read or heard elsewhere
    - Can take place at multiple stages in the pipeline
    - May depend on graphics hardware vendor (e.g. nVidia vs. ATI), graphics card family and API (e.g. DirectX vs. OpenGL)
    - Will change over time as graphics pipelines evolve



Application

Transform & Lighting

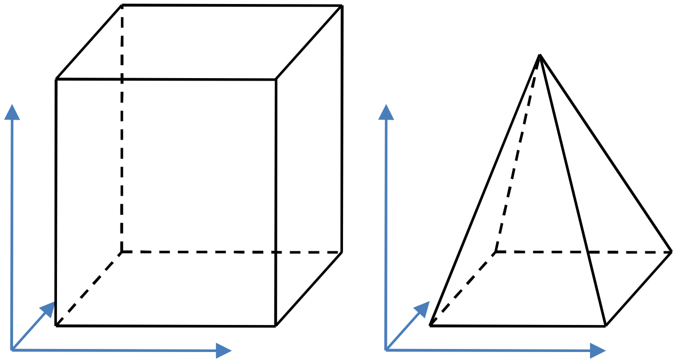
Rasterisation

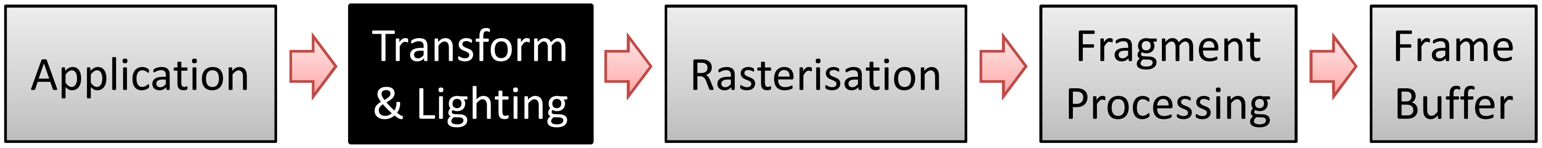
Fragment Processing

Frame Buffer

- Geometry submission
  - Typically vertices, normals and object-space texture coordinates
- Attribute submission
  - Material and colour settings, lights, texture bindings
- In OpenGL:
  - glBegin, glEnd, glVertex and glNormal
  - OpenGL state machine manipulations

- Object/World space

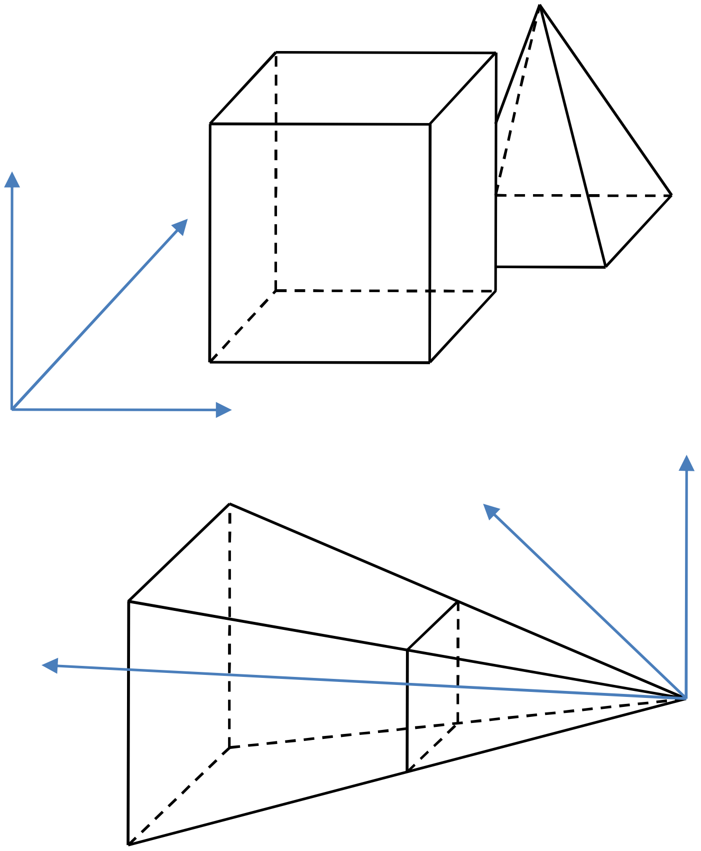


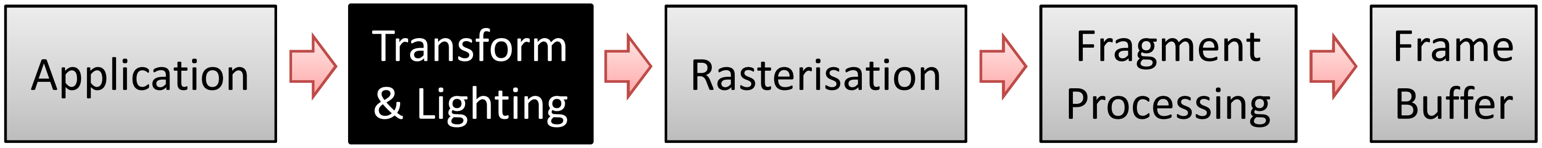


Application → **Transform & Lighting** → Rasterisation → Fragment Processing → Frame Buffer

- ModelView* transform → **View space**
  - – Normalize normals
  - – Vertex lighting
  - – Generate eye-space texture coordinates
- Trivial rejection and back face culling

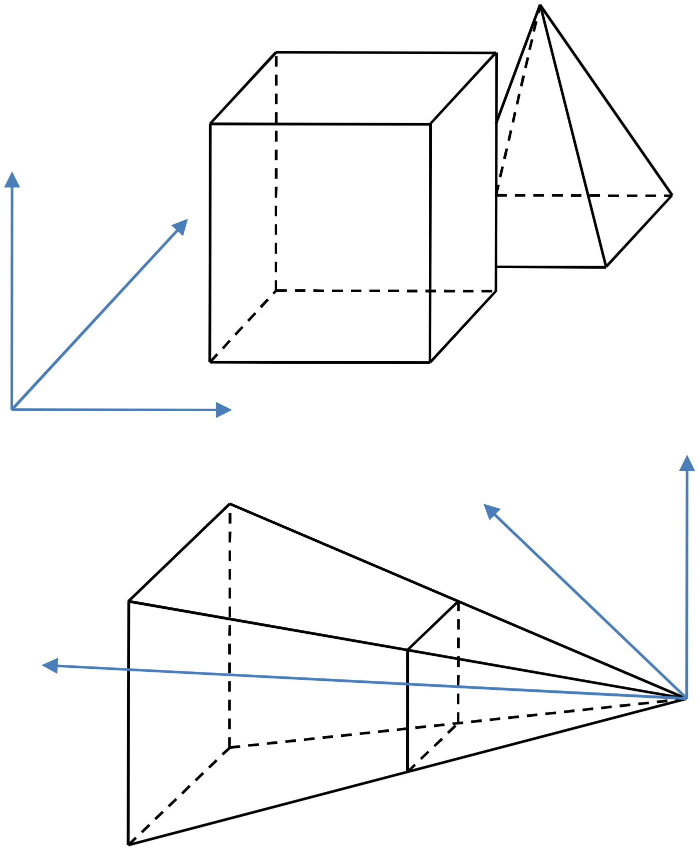
- Object/World space

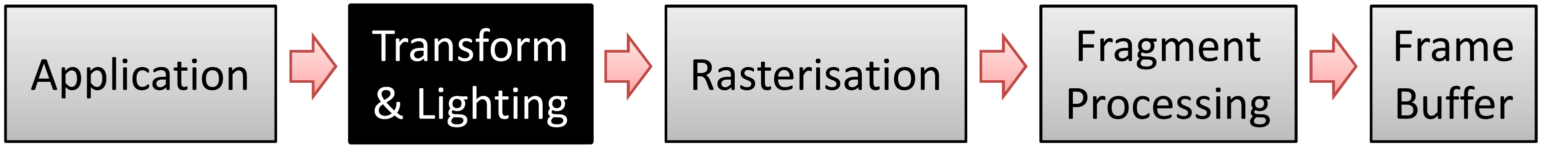




- ModelView* transform → **View space**
  - Normalize normals
  - Vertex lighting
  - Generate eye-space texture coordinates
- Trivial rejection and back face culling

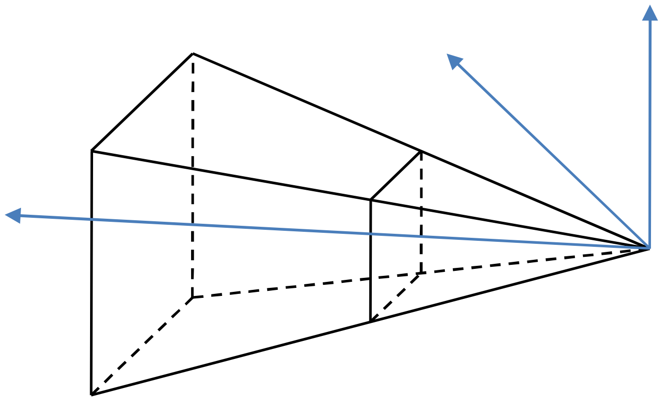
- View space

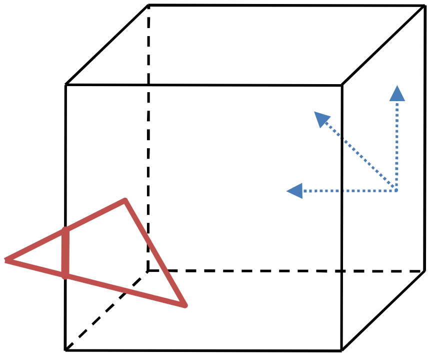


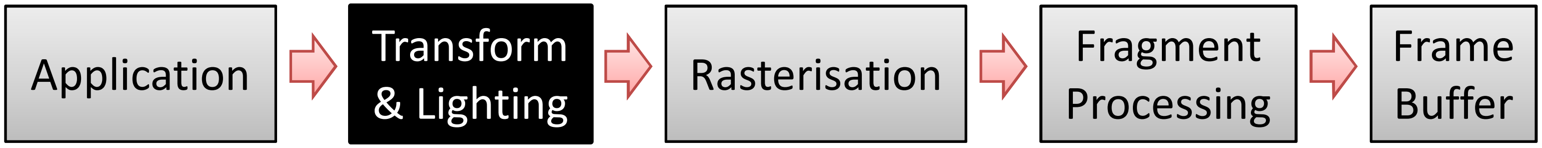


- Projection* transform and perspective (w) divide → **Clip space**
  - Normalised Device Coordinate cube
    - x and y from -1 to 1
    - z from 0 to 1
  - Gives depth cues such as perspective *foreshortening* and *motion parallax*
- Clipping
  - Clipped geometry is *retesselated*

- View space



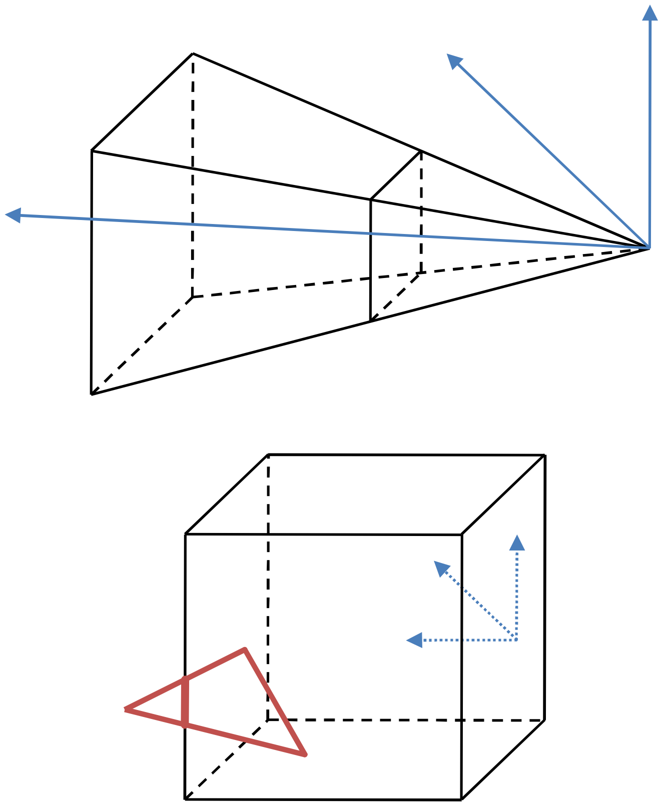


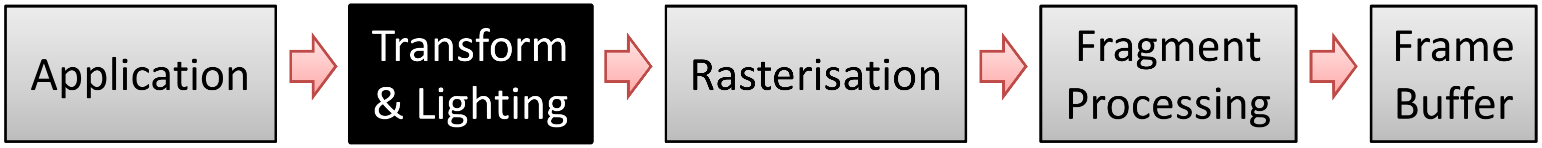


Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Projection* transform and perspective (w) divide → **Clip space**
  - Normalised Device Coordinate cube
    - x and y from -1 to 1
    - z from 0 to 1
  - Gives depth cues such as perspective *foreshortening* and *motion parallax*
- Clipping
  - Clipped geometry is *retesselated*

- Clip space




Application

Transform & Lighting

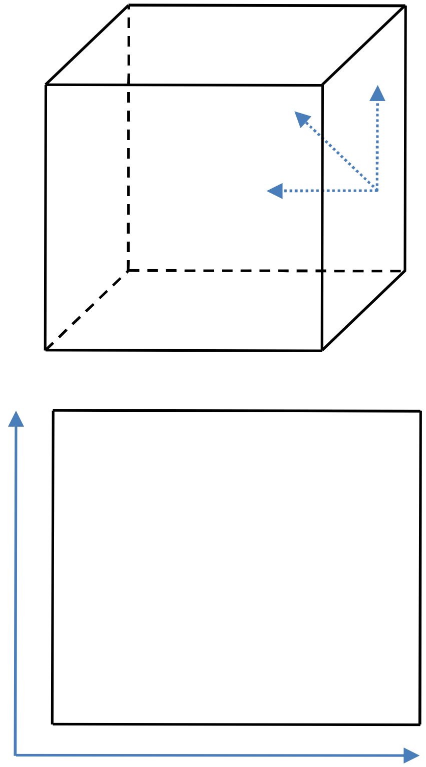
Rasterisation

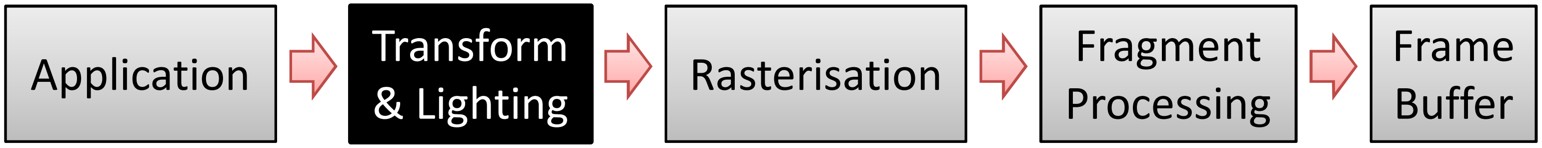
Fragment Processing

Frame Buffer

- Scale and translate → **Screen space**
  - The current viewport maps clip space to the frame buffer
  - 3D vertices are finally transformed to a 2D coordinate system
  - Although z and w are retained for fragment processing

- Clip space

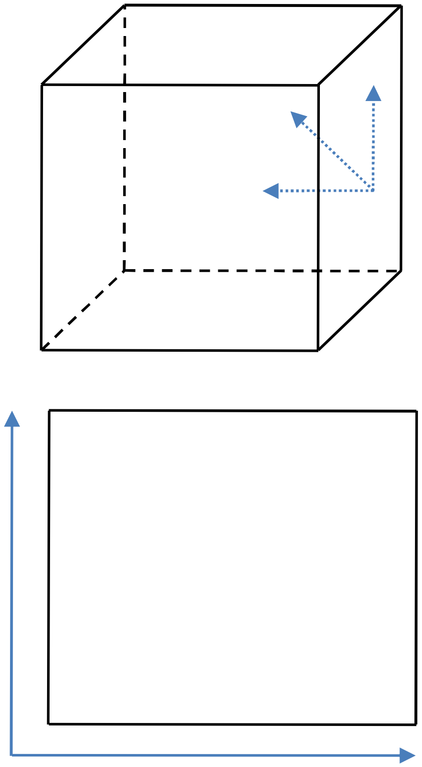


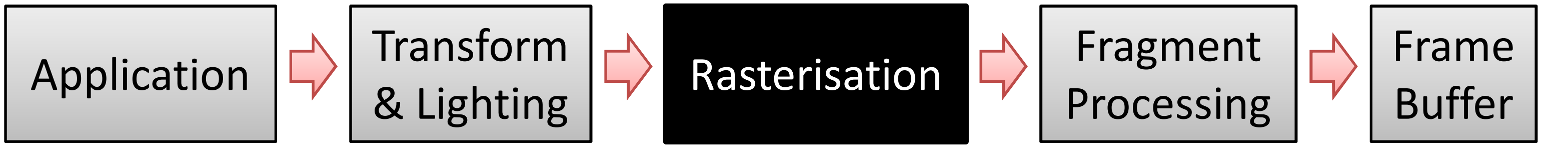


Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Scale and translate → **Screen space**
  - The current viewport maps clip space to the frame buffer
  - 3D vertices are finally transformed to a 2D coordinate system
  - Although z and w are retained for fragment processing

- Screen space

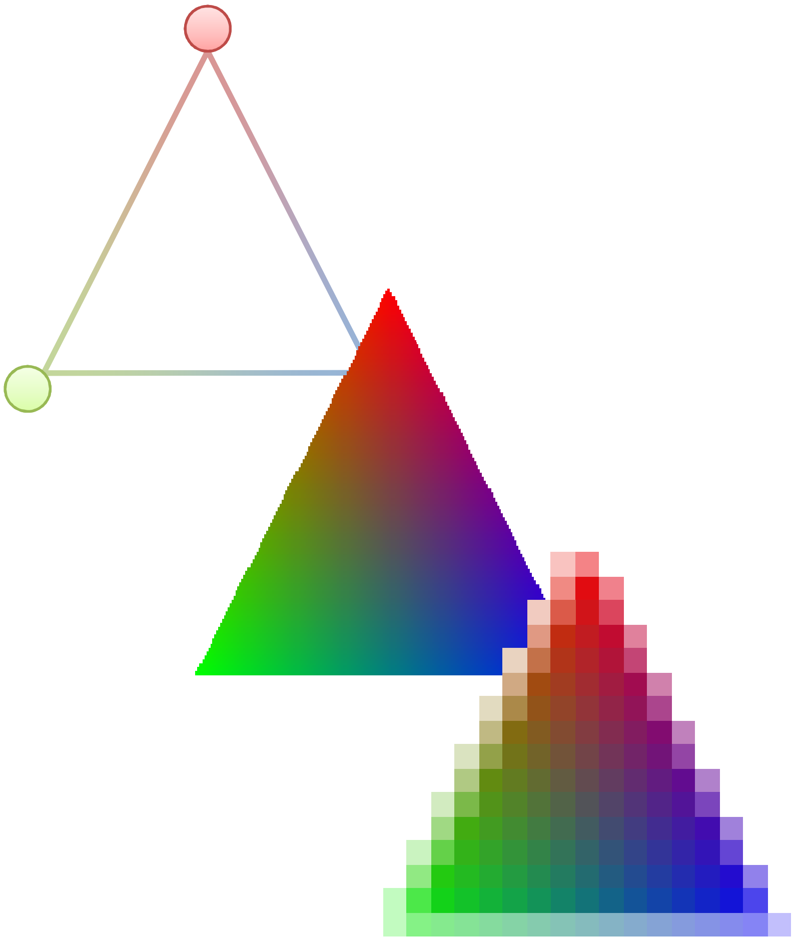


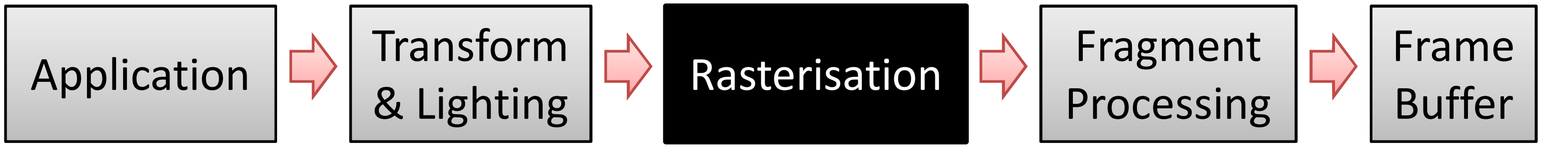


Application → Transform & Lighting → **Rasterisation** → Fragment Processing → Frame Buffer

- Maps continuous primitives to the frame buffer’s discrete grid
- Interpolates colour, texture coordinates and depth values across fragments
- Converts primitives into **fragments** (not pixels!)
  - Scan conversion

- Screen space

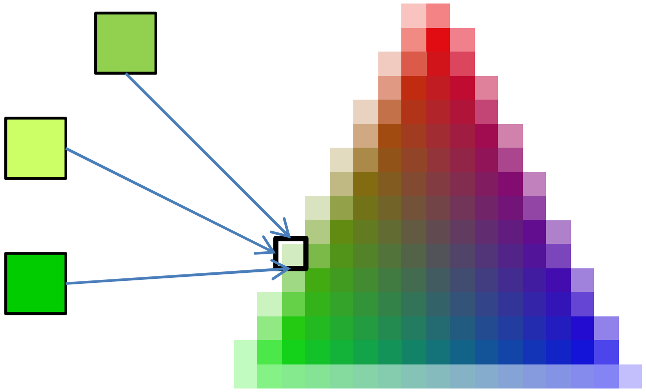


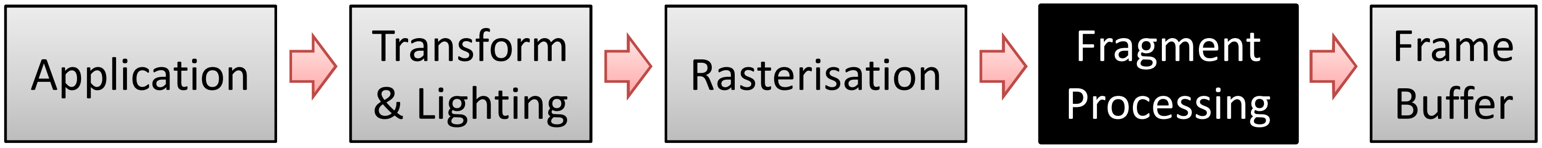


Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Fragments
  - Transient representations
  - Frame buffer position (x/y coordinates), colour and depth
- Fragments are **not pixels!**
  - Pixels belong to the frame buffer
  - Fragments could be considered *potential* pixels

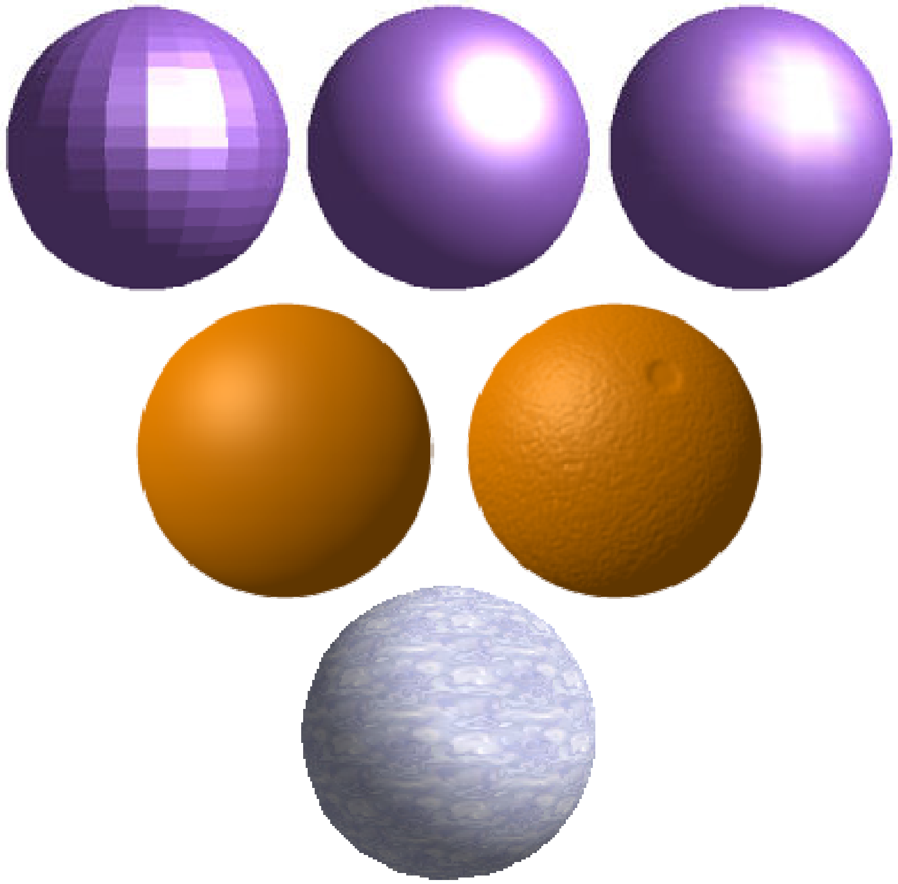
- Screen space

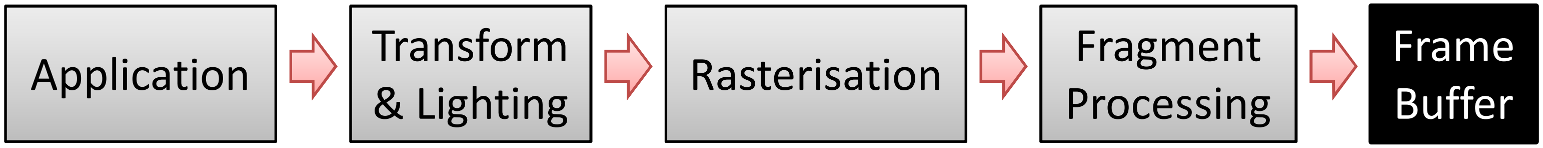



Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Fragment operations*
  - Fragments may be rejected by tests such as scissor, stencil, alpha and depth (**z-buffer**)
- Fragment shading based on:
  - Vertex colour attributes
  - Shading system in use (e.g. flat, Gouraud, Phong)
  - Texture lookup
- Special effects
  - Including fog, environment mapping, bump mapping, and shadows

- Screen space

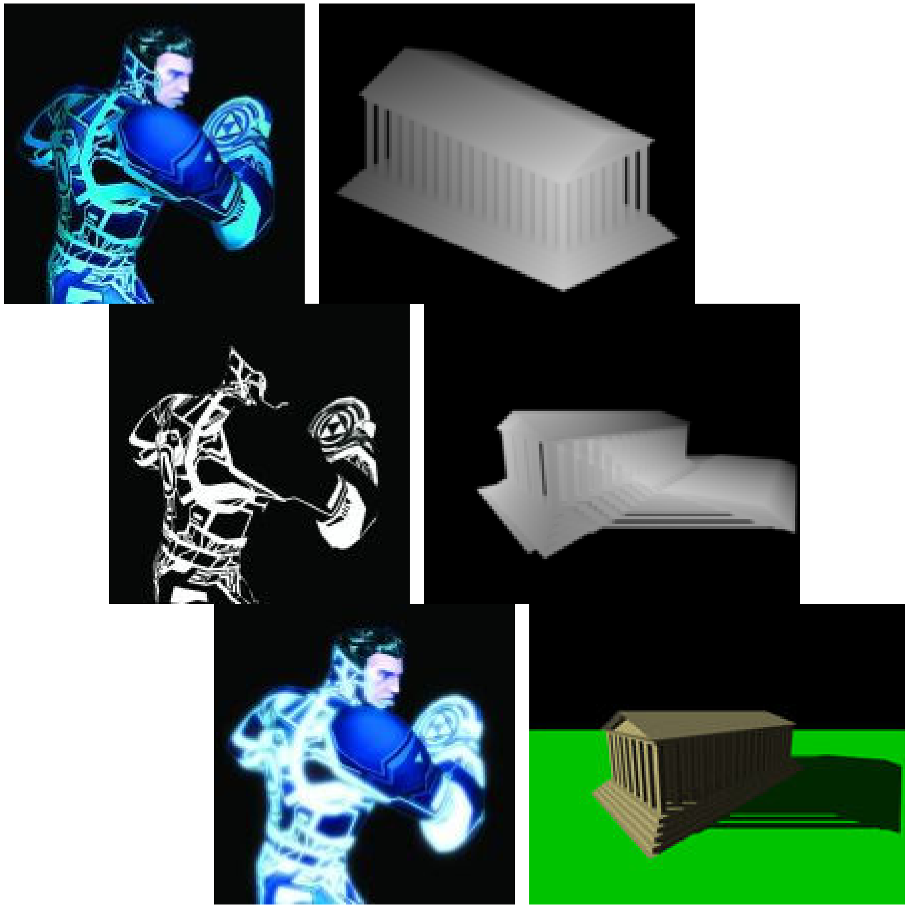




Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Pixel memory on the graphics card
  - May be output to the screen, or retained for further use (buffered)
- Texture targets
  - Available to the application for special effects, scene feedback (occlusion culling, shadow mapping), etc.

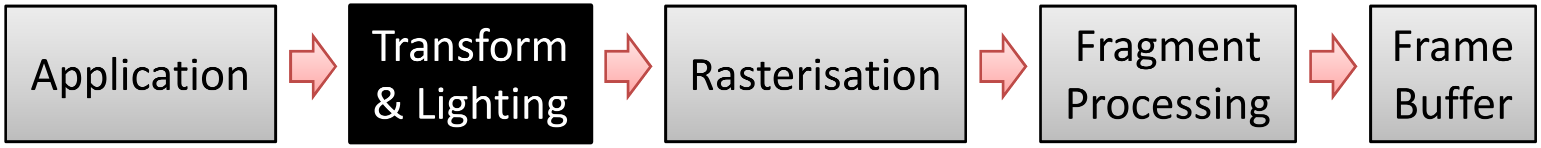
- Screen space



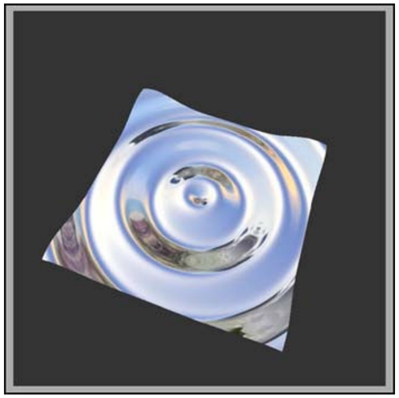
Video

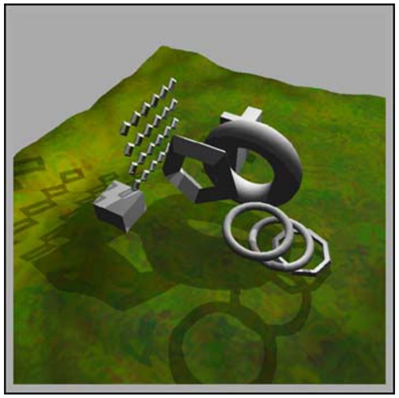
- Examples of fixed functionality games
  - Quake 2
  - Half life

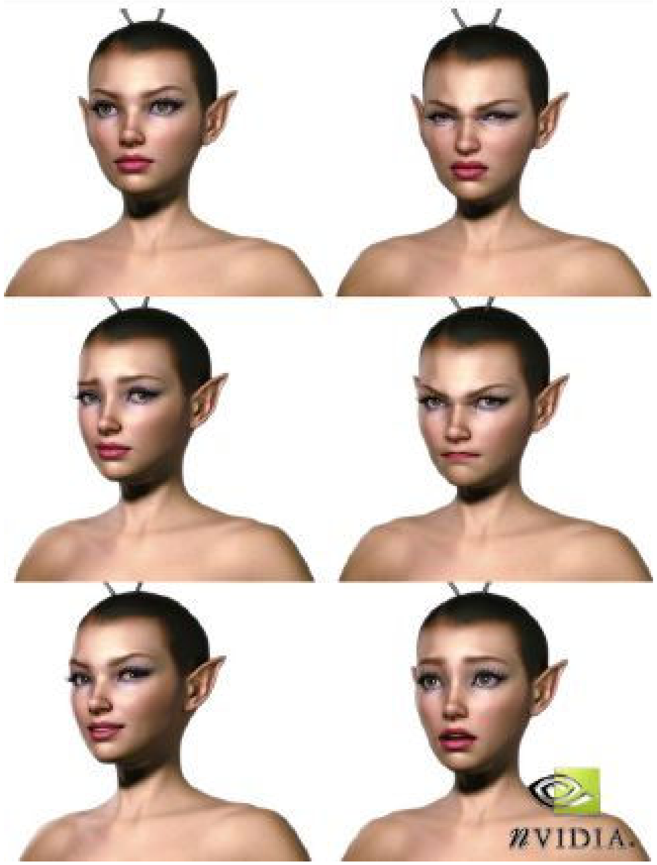
Programmable Graphics Pipeline


Application
Transform & Lighting
Rasterisation
Fragment Processing
Frame Buffer

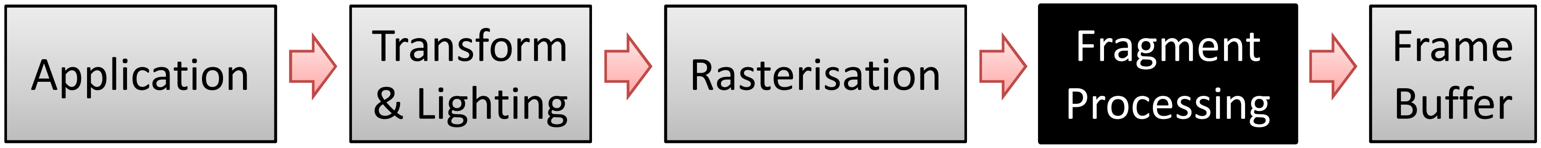
- Vertex programs replace fixed transform and lighting
- In: 1 vertex
  - 3D position
  - Normal
  - Texture coords
  - Colour
- Out: 1 vertex
  - 3D position
  - Normal
  - Texture coords
  - Colour
  - Screen space position







Programmable Graphics Pipeline



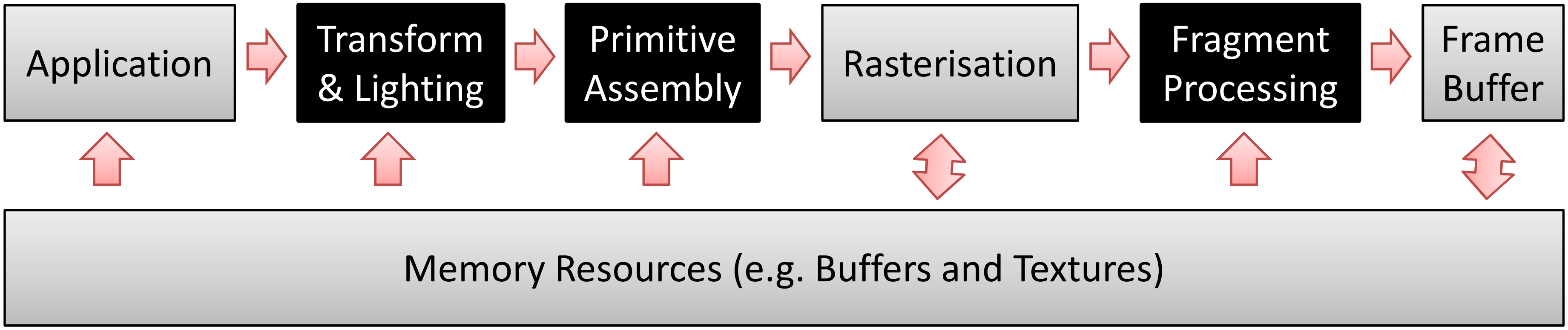
Application → Transform & Lighting → Rasterisation → Fragment Processing → Frame Buffer

- Fragment programs replace fixed fragment processing and selection
- In: 1 fragment
  - 3D position
  - Normal
  - Texture coords
  - Colour
  - Screen space position
  - Depth
- Out: 0 or 1 fragment
  - Colour
  - Depth

Video

- Examples of games based on today’s programmable technology
  - Quake 3 Arena
  - Doom 3
  - Gears of War
  - Ghost Recon Advanced Warfighter
  - FarCry 2

Modern Graphics Pipeline



Application

Transform & Lighting

Primitive Assembly

Rasterisation

Fragment Processing

Frame Buffer

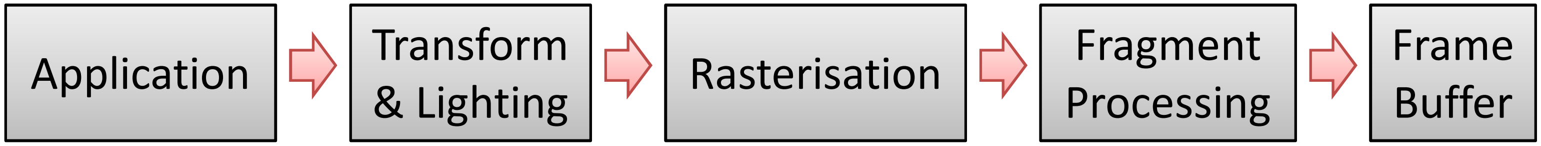
Memory Resources (e.g. Buffers and Textures)

- Geometry programs allow primitive assembly and manipulation prior to rasterisation
- New memory resources with more flexible read/write access

Video

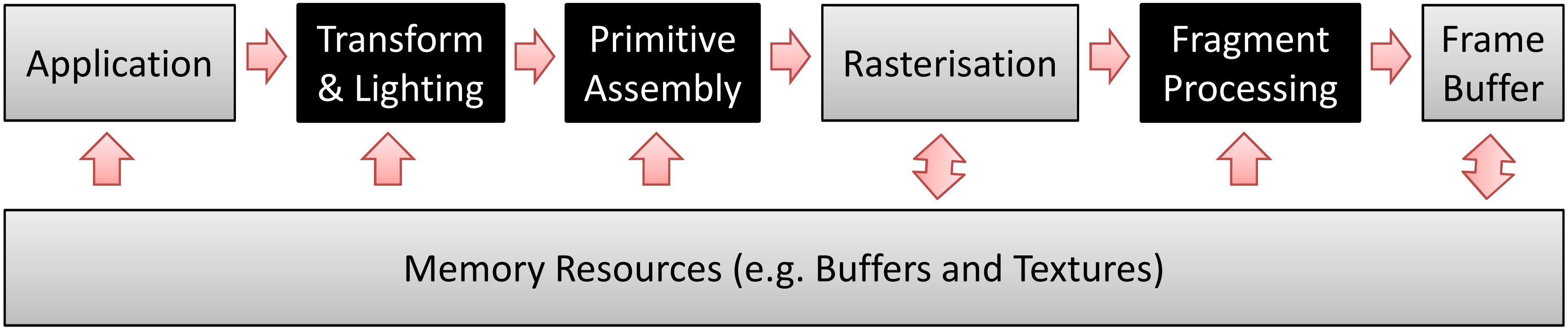
- Examples of what will be available soon
  - nVidia’s Human head demo
  - WarDevil
  - Project Offset
  - Gran Turismo 5
  - Smoke and Water demos

Summary



- The overall graphics pipeline is still here...

Summary



Application

Transform & Lighting

Primitive Assembly

Rasterisation

Fragment Processing

Frame Buffer

Memory Resources (e.g. Buffers and Textures)

- Although a few improvements have been made:
  - More programmable and flexible
- Video games continue to push the envelope, with:
  - More content and detail
- Video games are starting to look like computer generated films...