Week 9: Introduction to VFX
Part 1: The Graphics Pipeline

COMP270: Mathematics for 3D Worlds and Simulations

Objectives

- **Recall** the key stages of the graphics pipeline
- **Explain** the differences between a CPU and a GPU

Hardware: CPUs vs. GPUs

- CPU = central processing unit; GPU = graphics processing unit
- GPUs are highly parallelised
  - Intel i7 6900K: 8 cores
  - NVIDIA GTX 1080: 2560 shader processors
- GPUs are highly specialised
  - Optimised for floating-point calculations rather than logic
  - Optimised for performing the same calculation on several thousand vertices or pixels at once

General Purpose GPU (GPGPU)

- Early GPUs used a fixed pipeline – could only be used for rendering 3D graphics
- Modern GPUs use a programmable pipeline – can be programmed for other tasks
  - Physics simulation (e.g. PhysX)
  - Scientific computing (e.g. CUDA)
  - Deep learning (TensorFlow, PyTorch)

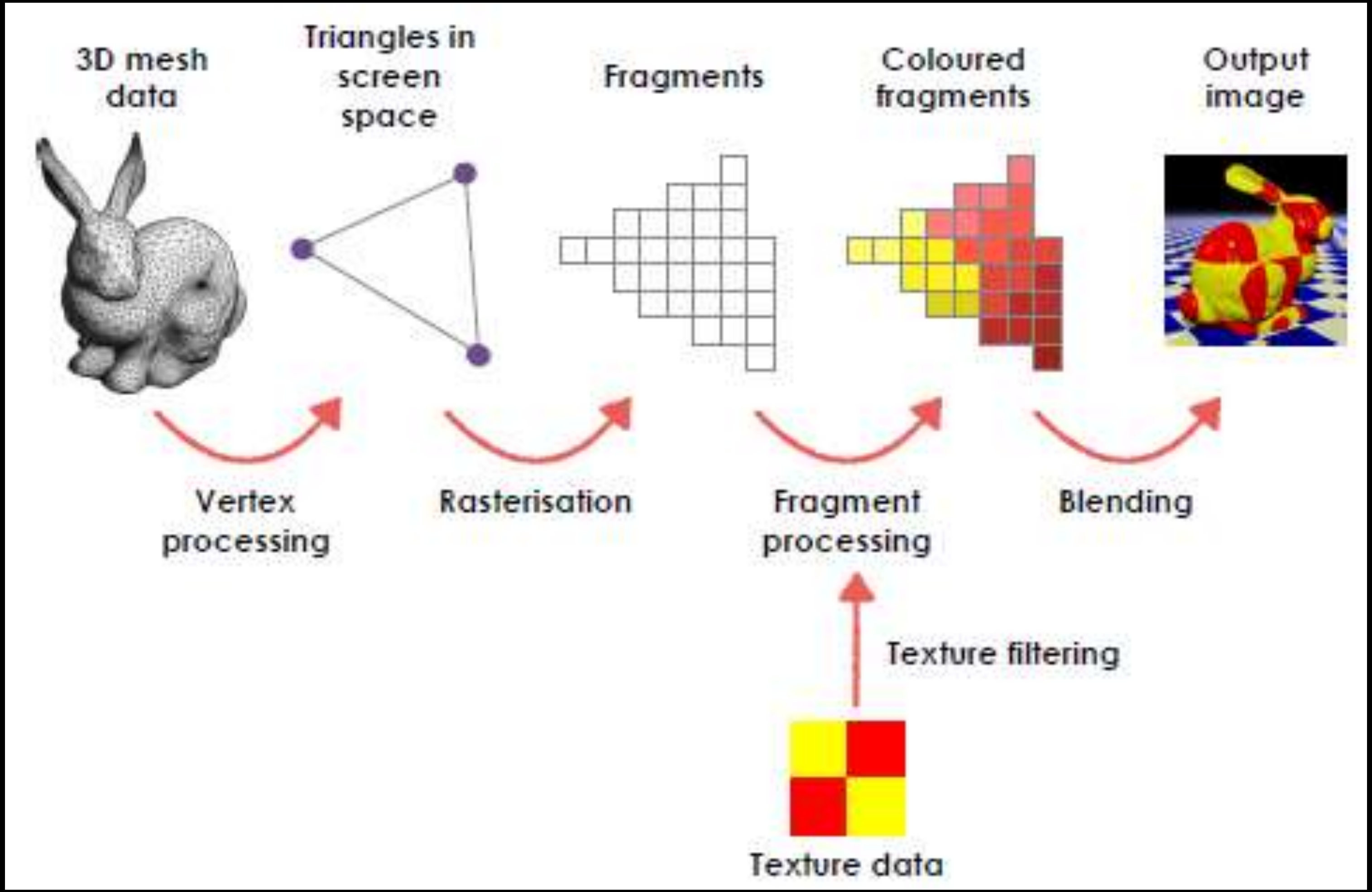
Graphics APIs

- Graphics APIs abstract away the differences between different manufacturers' GPUs
- There are several APIs in use today:
  - OpenGL: Open standard, very mature (since 1992), very widely supported
  - Vulkan: Open standard, less mature, lots of control on rendering, lots of work to get a basic sample working
  - Direct3D: Microsoft only
  - Metal: Apple only
  - Sony and Nintendo consoles have their own APIs (Microsoft use Direct3D)

Game Engines and Graphics APIs

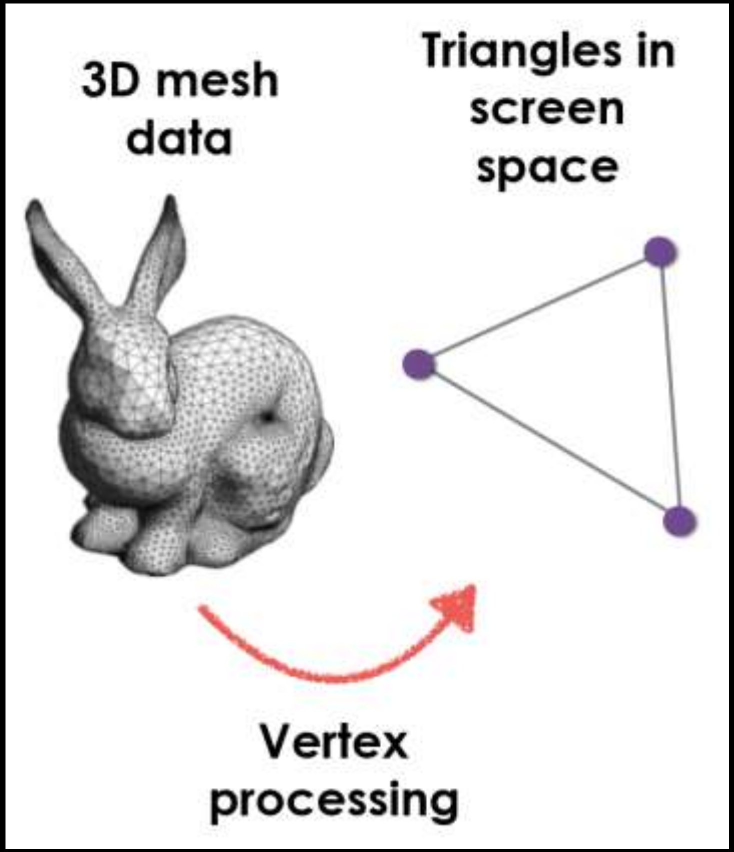
- Game Engines tend to support multiple Graphics API
- They have an abstract rendering layer which has concrete implementations of D3D, OpenGL, Vulkan, Metal, Console APIs etc.
- This allows the Engine to support multiple platforms
- In addition, this makes it easier to upgrade the engine to support new versions of APIs or newly released APIs

The 3D Graphics Pipeline

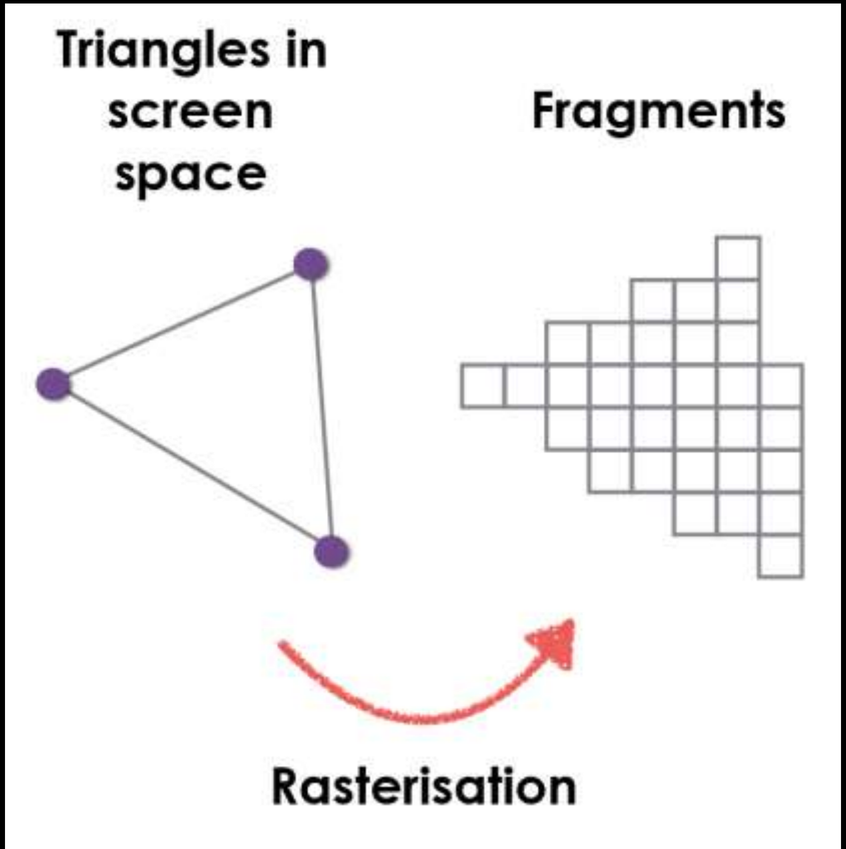


Vertex Processing

- Geometry is provided to the GPU as a mesh of triangles
- Each triangle has three vertices specified in 3D space (x, y, z)
- **Vertex processor** transforms (rotates, moves, scales) vertices and projects them into 2D screen space (x, y)
- May also apply particle simulations, skeletal animations or deformations etc.
- Can associate data with each vertex



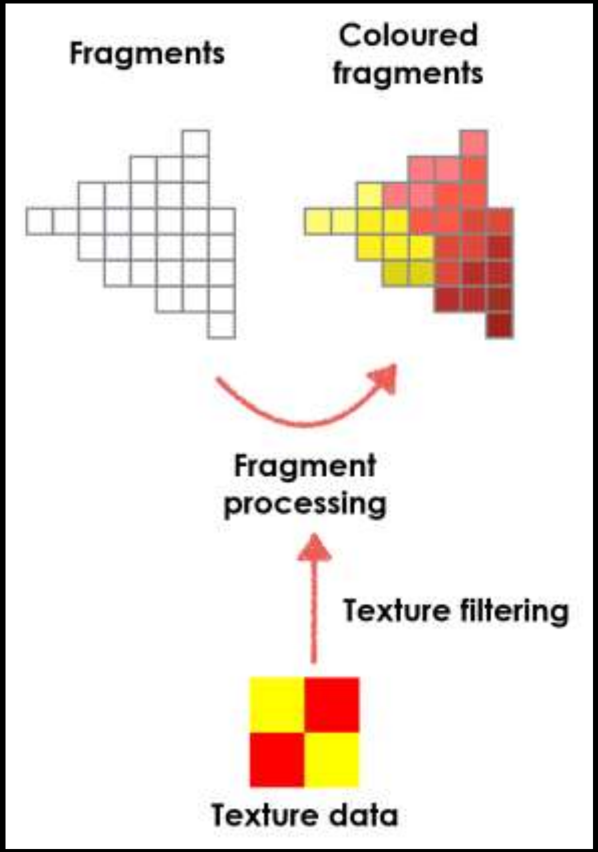
Rasterisation



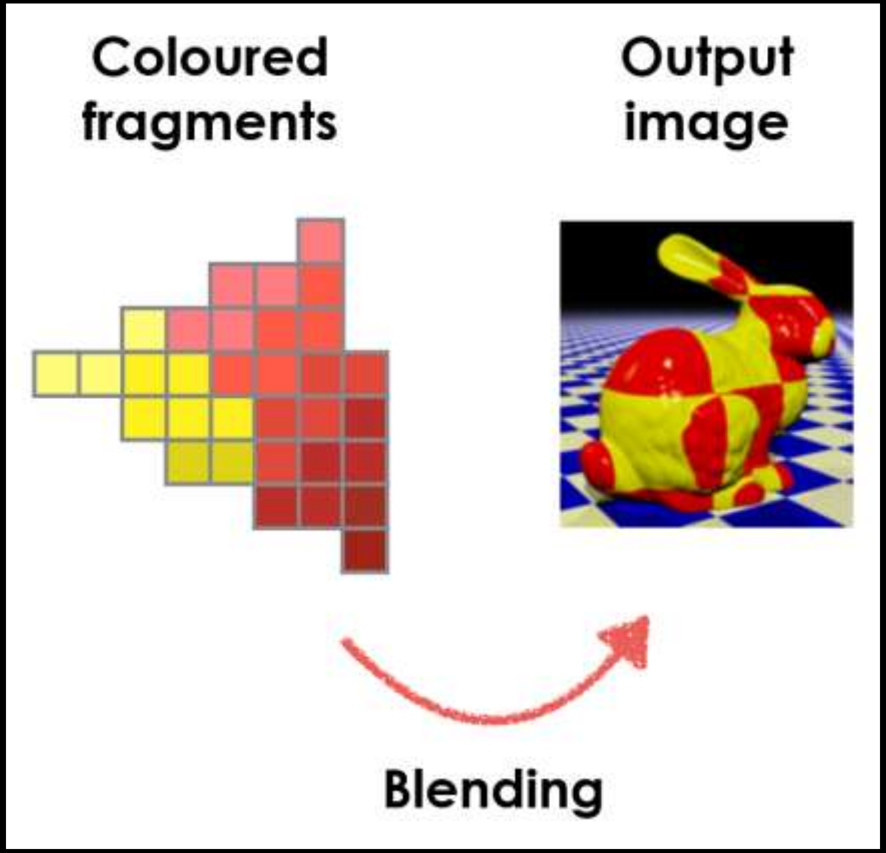
- Determine which fragments are covered by the triangle
  - In practical terms, “fragment” = “pixel”
  - Alternative to ray-tracing/ray-casting

Fragment Processing

- Determine the colour of each fragment covered by the triangle
- Textures are 2D images that can be wrapped onto a 3D object
- Colour is calculated based on texture, lighting and other properties of the surface being rendered (e.g. shininess, roughness)



Blending



- Combine these fragments with the existing content of the image buffer (or frame buffer)
- Depth testing:
  - if the new fragment is "in front" of the old one, replace it
  - if it is "behind", discard it
- Alpha blending: combine the old and new colours for a semi-transparent appearance