The Real-Time Graphics Pipeline

Benjamin Brown

Monday, 26th September 2022

The Key Idea

Aim is to ***render*** 3d objects on 2d screen

Two methods of rendering:

- ***object-order rendering***: for each object, which pixels are influenced by it?
- ***image-order rendering***: for each pixel, which object is influenced by it?

Object- vs. Image-Order Rendering

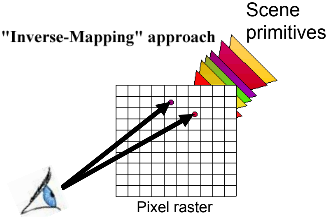


Figure 1: Image-order rendering.

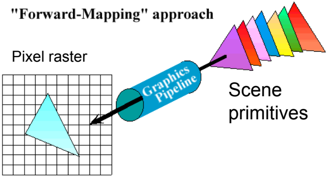


Figure 2: Object-order rendering.

Real-Time Rendering

Real-time refers to rendering a scene in less than 1/30th of a second.

- Fast enough to allow for the user's ***real-time interaction***.
- Temporal delay of 15 ms of temporal delay can interfere with the interactivity.

Speed is essential!

- 1080p image at 90 Hz, image-order rendering needs 186,624,000 iterations times per second!

The Graphics Pipeline

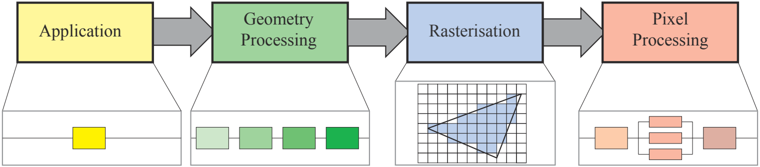
Object-order rendering uses the ***graphics pipeline***:

- Given a virtual camera, 3d objects, light sources, etc., render a 2d image;
- Object locations and shapes determined by their geometry, environment, camera placement, etc.;
- Object appearance affected by material, light sources, shading, etc.

The states execute in parallel; each stage depends on the previous.

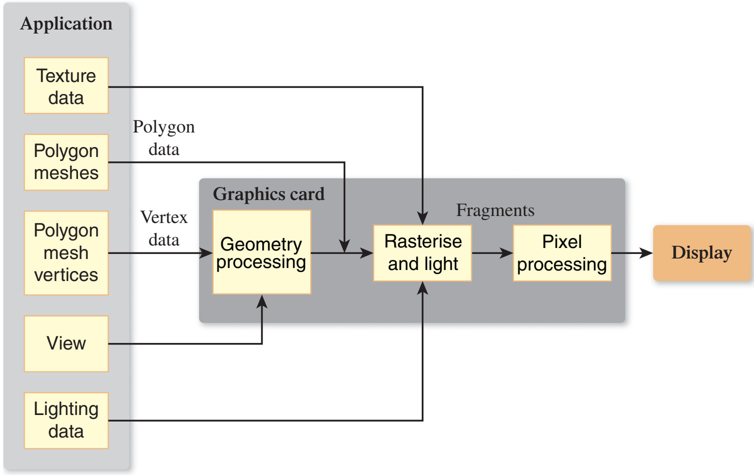
Stages of the Graphics Pipeline

Roughly, these stages are:



Geometry processing, rasterisation, and pixel processing happen within the ***GPU***.

Stages of the Graphics Pipeline



Application Stage

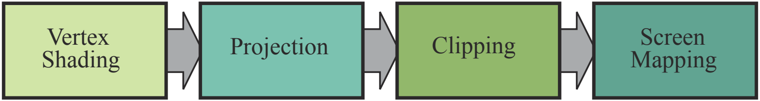
The ***application*** on the ***CPU*** supplies the data about the scene to the ***GPU***:

- ***Primitive*** data:
    - Vertex data (position, normal vectors, colour, etc.),
- Initialises GPU memory ***buffers***:
    - Vertex and index buffers.

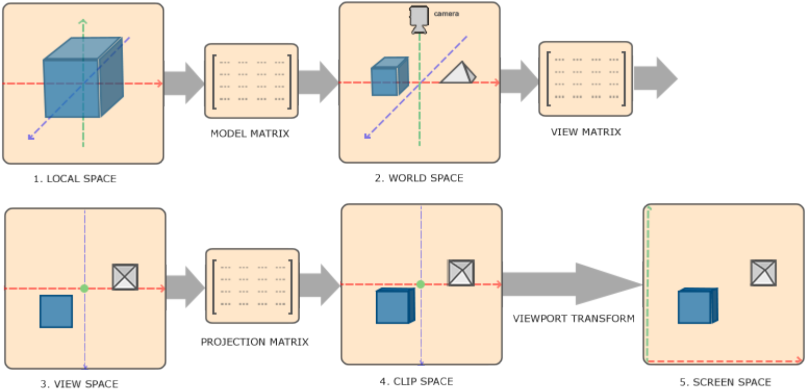
Geometry Processing Stage

Performs per-triangle and per-vertex operations:

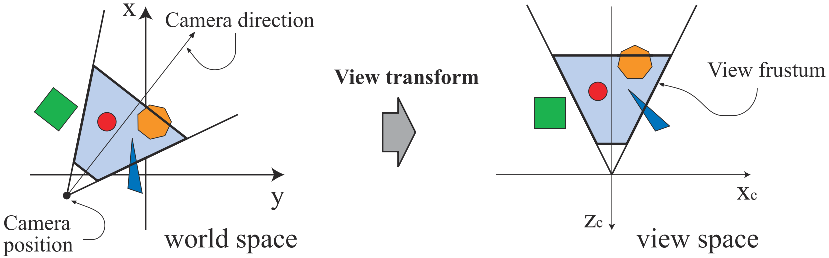
- Vertex processing (transforming and shading);
- Projecting;
- Clipping;
- Screen mapping.



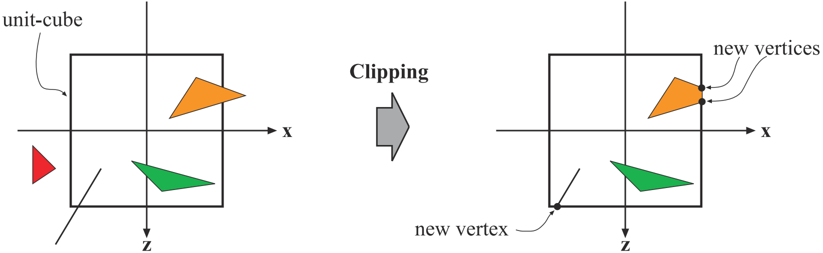
The Geometry Stage – Transformations



The Geometry Stage – Local-to-View Space

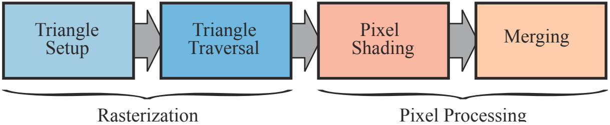


The Geometry Stage – Vertex Shading



The Rasterisation Stage

Rasterisation can be split into two sub-pipelines:



Triangle Setup

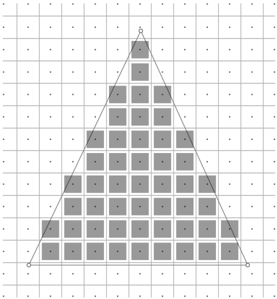
Triangle Traversal

Pixel Shading

Merging

Rasterization

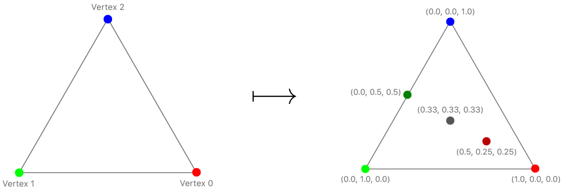
Pixel Processing

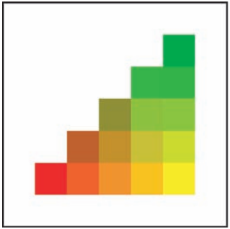


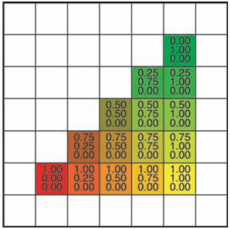
Data for each pixel is called a ***fragment***.

The Rasterisation Stage – Pixel Shading

- Example – fragment data from interpolation.







					0.00 1.00 0.00	
				0.25 0.75 0.00	0.25 1.00 0.00	
			0.50 0.50 0.00	0.50 0.75 0.00	0.50 1.00 0.00	
		0.75 0.25 0.00	0.75 0.50 0.00	0.75 0.75 0.00	0.75 1.00 0.00	
	1.00 0.00 0.00	1.00 0.25 0.00	1.00 0.50 0.00	1.00 0.75 0.00	1.00 1.00 0.00	

The Rasterisation Stage – Pixel Shading

- Example – light shading.

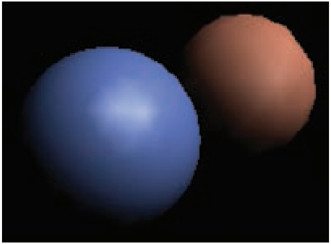


Figure 3: Per-vertex shading.

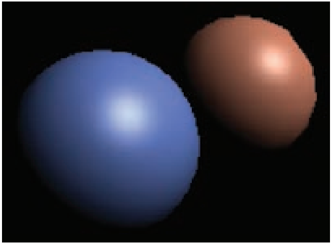
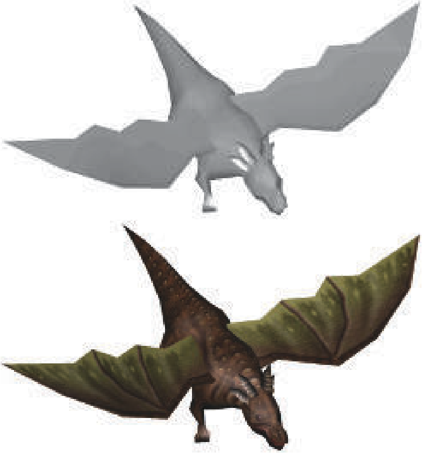
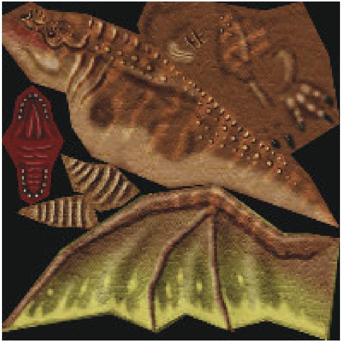


Figure 4: Per-fragment shading.

The Rasterisation Stage – Pixel Shading

- Example – texture mapping.





The Rasterisation Stage – Merging

- Storing pixels in a colour buffer;
- With a depth- or z-buffer.

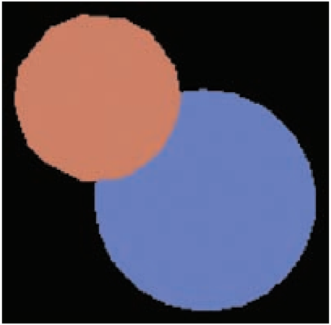


Figure 5: No depth sorting.

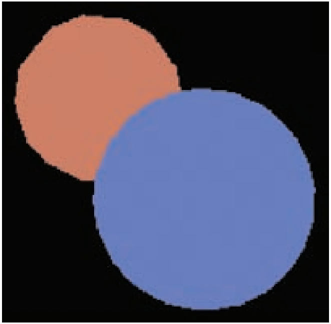


Figure 6: With depth sorting.

Final Remarks

Modern day 3d graphics APIs include:

- Vulkan/OpenGL (Khronos Group);
- Direct3D (Microsoft);
- Metal (Apple).

The pipeline performs parallel and regular computations $\Longrightarrow$ GPUs are specialised for this!