Working with shaders

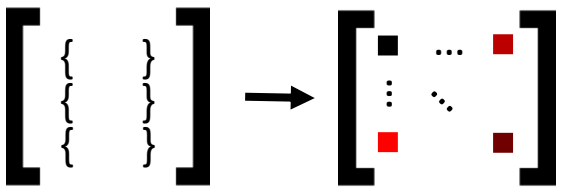
Patrick SARDINHA

What’s a shader?

Shaders are small programs that run on the GPU

They are run for each specific section of the graphics pipeline

They are isolated & are not allowed to communicate with each other



Their purpose is to transform inputs to outputs

It works with geometric primitives, lights, textures, …

Different languages

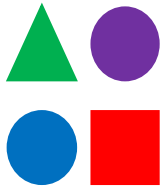
OpenGL Shading Language

DirectX High-Level Shader Language

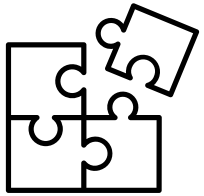
Cg Shader Language

Goal of the project

Construct a DSL for shaders

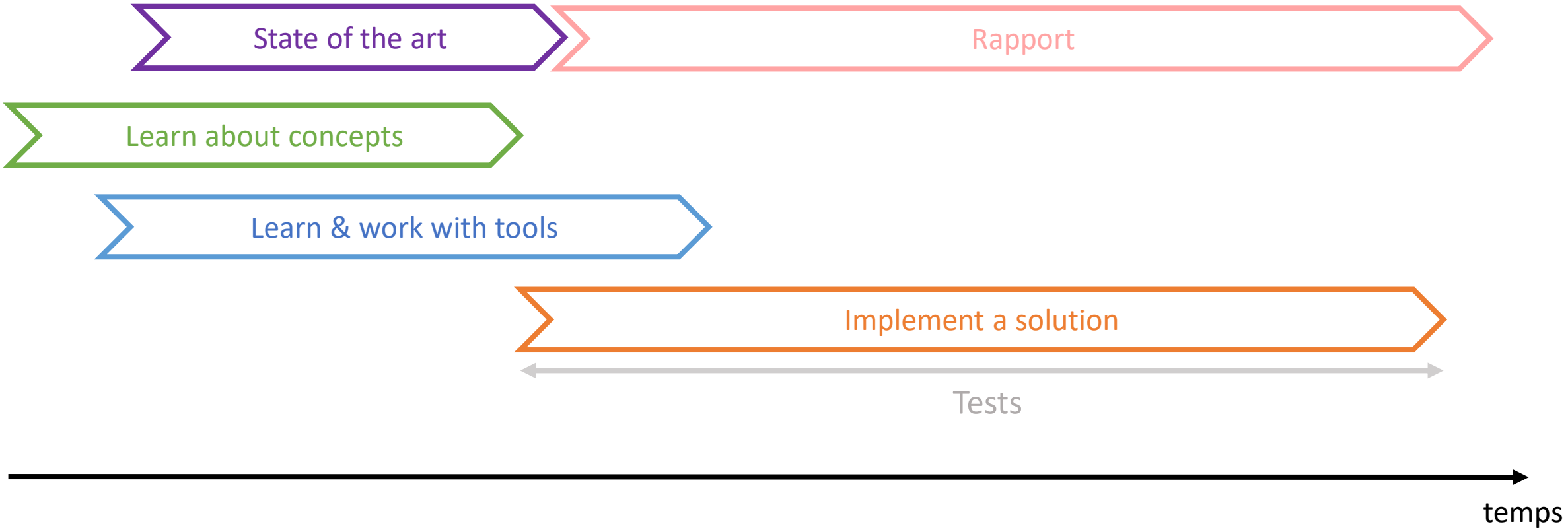


Abstract types

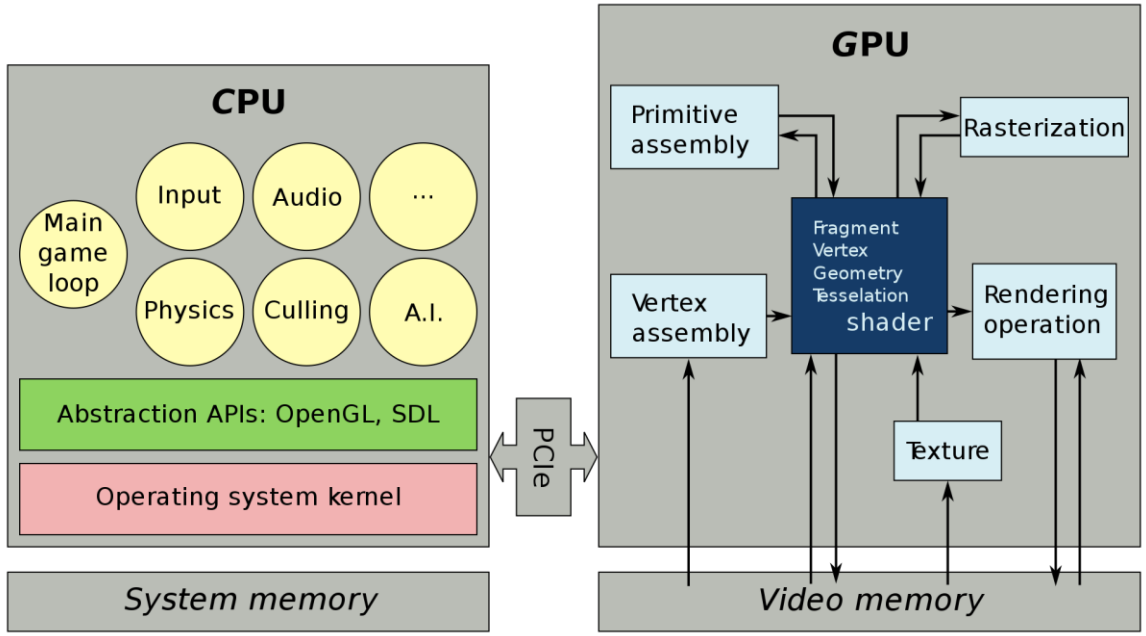


Factorize several shaders as one

Road map


State of the art
Rapport
Learn about concepts
Learn & work with tools
Implement a solution
Tests
temps

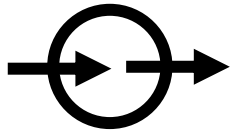
Shaders in the Graphics Processing Unit



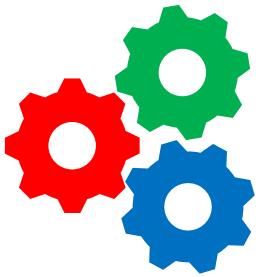
Work between CPU & GPU

Shaders are good to be executed in parallel and are executed by the GPU

Graphics pipeline

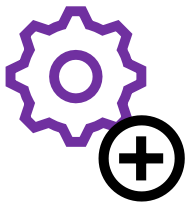


Input & Output Data



3 different shaders processing units

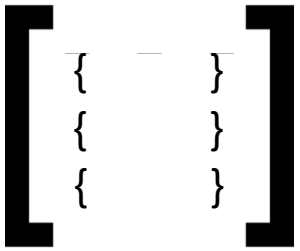
- Vertex Shader
- Geometry Shader
- Fragment/Pixel Shader



Some others processes

Tessellation, Rasterization, Color blending

Input Data



Take as input a Vertex (Vertices) [] which is a data structure that describes geometric primitives with certain attributes like:

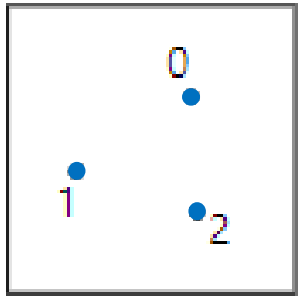
Position (2D, 3D coordinates)

Color (RGB, …)

Reflectance

Texture coordinates

Vertex Shader

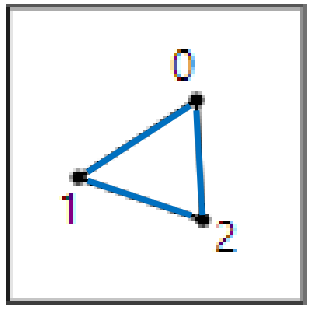


Compute the projection of the vertices of primitives from 3D space into 2D screen space

Input data: some properties of the vertices (position, color or texture coordinates)

Output data: the corresponding properties on the screen

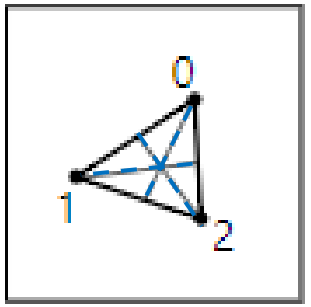
Primitives Assembly



This process takes all the vertex given by the step before and assemble them in order to create a geometric shape

Sources : https://learnopengl.com

Tessellation

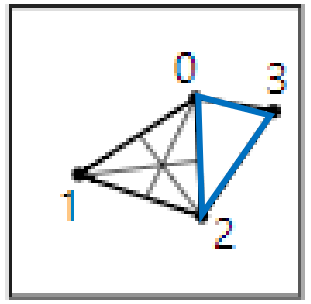


In 3D, the surfaces are built with triangular tiles

Tessellation allows to double triangles on a given surface and therefore increase the level of details

Sources : https://www.modding.fr/cest-quoi-la-tessellation/

Geometry Shader

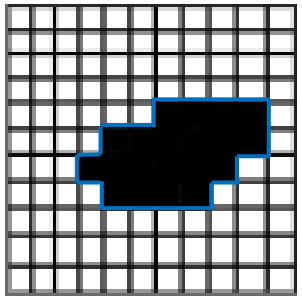


Allows to modify the geometry of each polygon and allows to create new polygons

Input data: data of a geometric primitive

Output data: data of one or more geometric primitive (with the Primitives Assembly step)

Rasterization



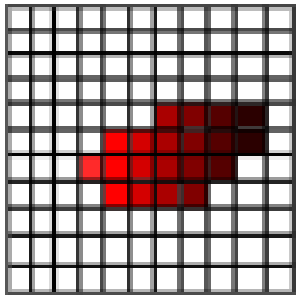
Method of converting a vector image into a raster image to be displayed on a screen

Vector image
(composed of geometric objects)

Raster image or Bitmap
(composed of pixels)

Sources : https://fr.wikipedia.org/wiki/Rastérisation/

Fragment/Pixel Shader

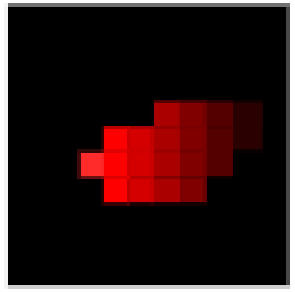


Calculates the color of each pixel individually

Input data: data of each pixel of the image (position, texture coordinates, color)

Output data: the pixel color

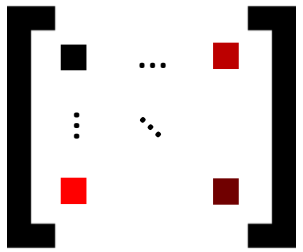
Color Blending



The technique of gently blending two or more colors to create a gradual transition

Sources : https://www.thesprucecrafts.com

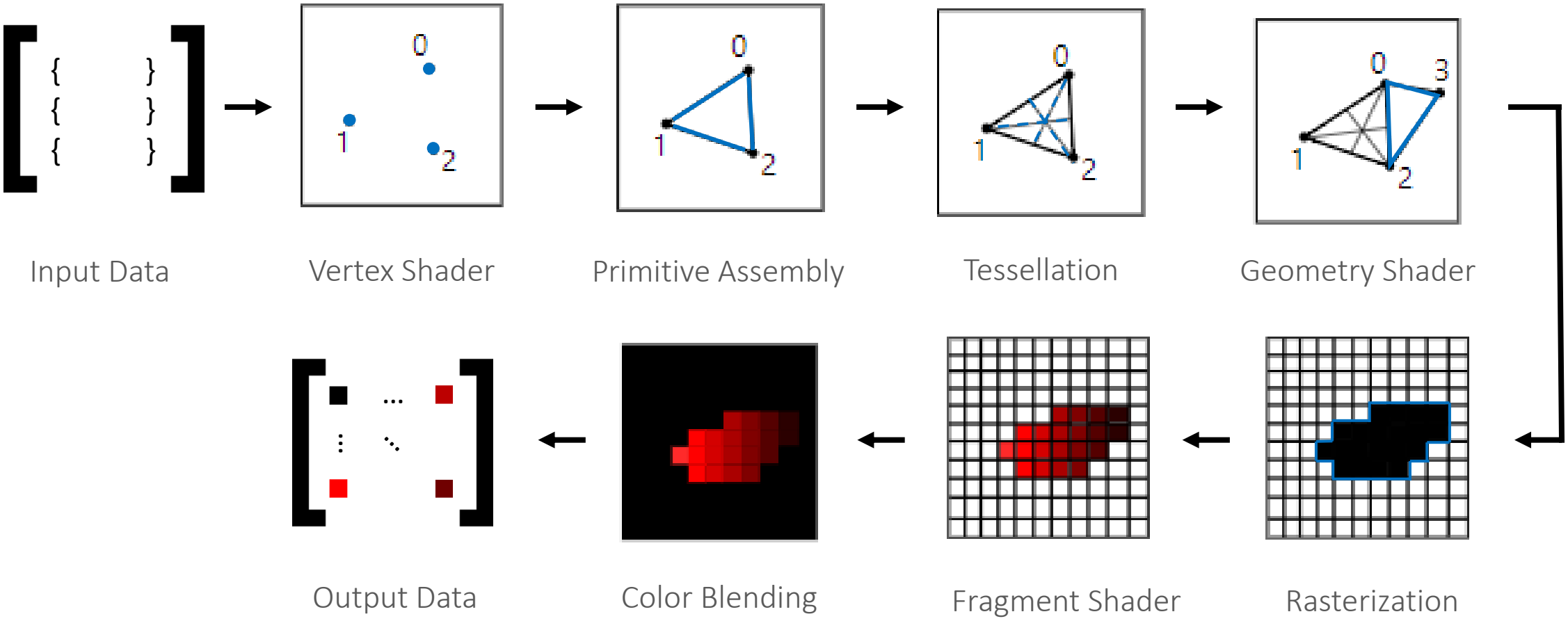
Output Data



Return a Framebuffer

The information in this buffer are the values of the color components (RGB) for each pixel

Overall view



Work incoming

Learn more about shaders & OpenGL

Begin to work with Rendery

Learn about DSL & SIMD language

References

https://fr.wikipedia.org/wiki/Shader

https://github.com/RenderyEngine/Rendery

https://fr.wikipedia.org/wiki/OpenGL

https://www.khronos.org/opengl/wiki/Core_Language_(GLSL)

https://fr.wikipedia.org/wiki/DirectX

https://en.wikipedia.org/wiki/Cg_(programming_language)

https://developer.apple.com/metal

https://learnopengl.com

https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview

Working with shaders

Patrick SARDINHA