Shaders

Teaching Modern Computer Graphics

Anton Gerdelan
gerdela@scss.tcd.ie

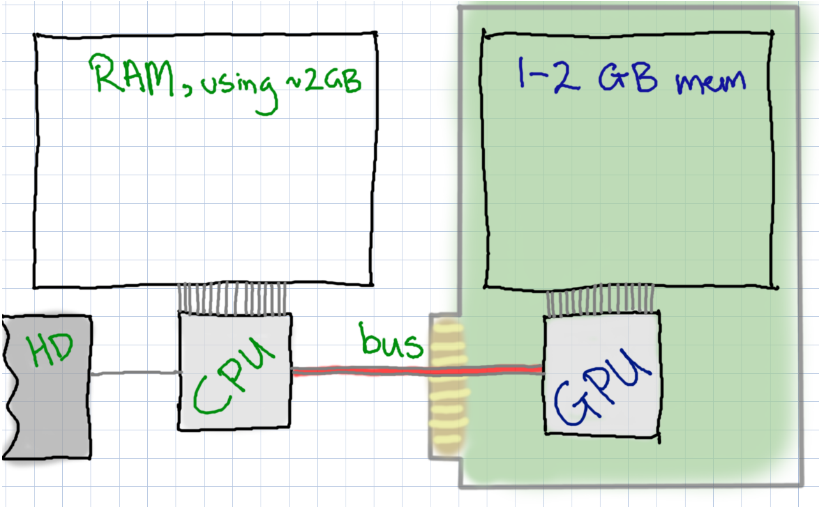
Knowledge and Data Engineering Group
Trinity College Dublin, Ireland

March 5, 2014

Overview

- Sharing some teaching and obsess...learning experiences
- Overview of the “new” computer graphics pipeline
- Shader programming
- WebGL and OpenGL ES
- Resources of note
- Some bold criticism / common problems

Modern Hardware



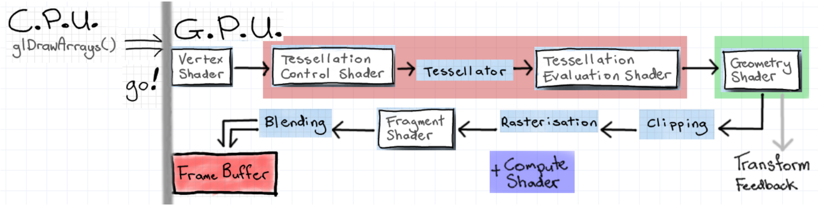
Data Parallelism

Copy all drawing data **before** drawing loop starts

- per-vertex “attribute” data points, normals, texture coordinates,...
- textures
- linked shader programmes
- buffers of commonly-used data (camera transformation matrices)

Update as seldom as possible e.g. camera matrices

Graphics Pipeline



- Many new re-programmable stages
- Some optional
- Vertex Shader (transform) and Fragment Shader (colour) are basics

APIs

In general:

- Direct3D - horrible OTT ‘90s-style object-oriented interface
- OpenGL - horrible crusty ‘80s-style state-machine C interface
- Must be good at memory allocation, pointers, addressing
- These are all huge problems for students learning
- Much better documentation for D3D
- Much better platform support for GL, docs are getting better-ish
- Functionality is almost 100% the same now (hardware-driven)
- GLSL and HLSL shaders are almost 100% the same
- Shaders are a lot of fun

APIs

Latest versions:

- WebGL 1.0 / OpenGL ES 3.0 / OpenGL 2.1 / Direct3D 9 - vertex shaders + fragment shaders
- OpenGL 3.2 / Direct3D 10 - geometry shaders
- OpenGL 4 / Direct3D 11 - tessellation shaders
- OpenGL 4.3 / Direct3D 11 - compute shaders (GPGPU)

What's Gone?

- `glBegin()` and `glEnd()` - gone
- `glMatrix` - gone
- `glLight` (point/spot/directional) - gone
- **glut** - plenty of modern alternatives
- You must know how the memory/processor architecture works (roughly)
- You must know how to do transformation matrices and dot products
- You must write the lighting and texturing algorithms by hand

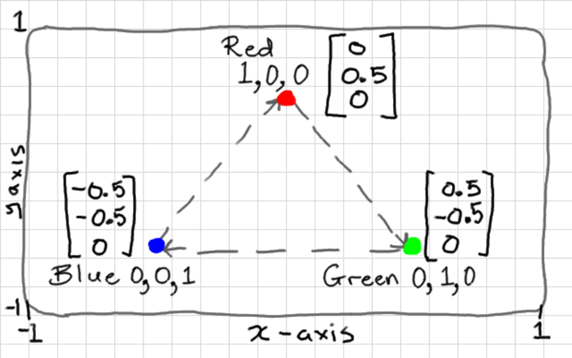
‘Client’ side (what happens on CPU)

1. use supporting API to start an OS window/canvas
2. use API start a graphics ‘context’
3. attach context to window/surface
4. load/download meshes, textures
5. use API to create buffer and copy meshes/textures onto graphics mem.
6. load shaders from strings. compile and link shaders. copy to GPU
7. switch shaders/geometry data to draw with
8. occasionally update a matrix
9. say `draw()` using current shaders and buffered data

In other words **not a lot** - just tie everything together

“Hello Triangle”

Define a triangle of XYZ points in a vertex buffer



- can add an RGB colour for each point too
- just put into a arrays of 18 `float`s, total
- “bind” or enable this buffer before drawing

“Hello Triangle”

Vertex Shader in GLSL

```
#version 440

in vec3 vpos, vcolour;
out vec3 colour;

void main () {
  colour = vcolour;
  gl_Position = vec4 (vpos, 1.0);
}
```

- set each vertex position in homogeneous clip space
- output any variables to next stage in pipeline

"Hello Triangle"

Fragment Shader in GLSL

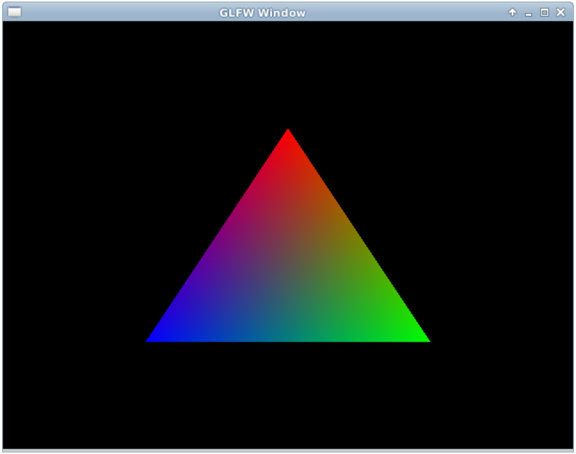
```
#version 440

in vec3 colour;
out vec4 frag_colour;

void main () {
  frag_colour = vec4 (colour, 1.0);
}
```

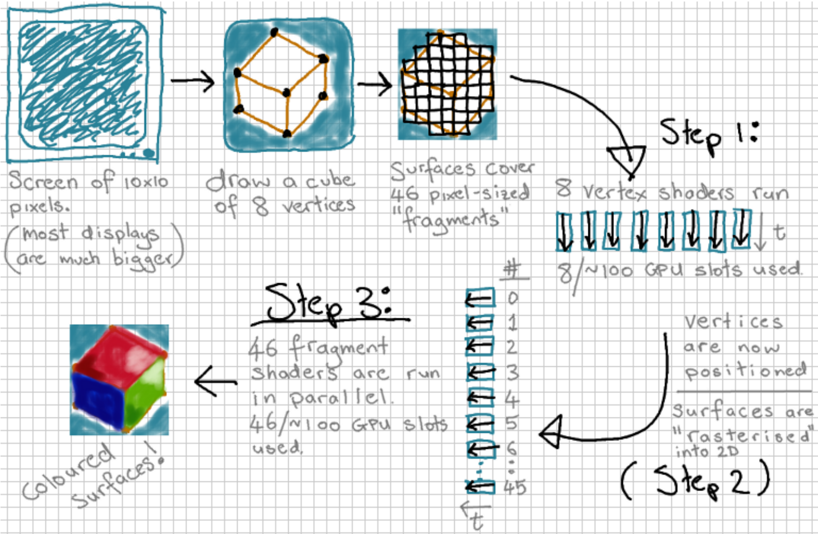
- set colour of each pixel-sized fragment on surface of geometry
- any input variables are **interpolated** to fragment position

“Hello Triangle”



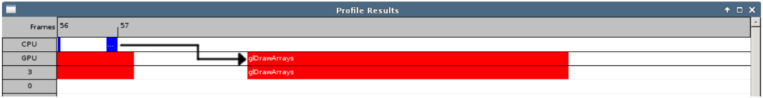
- Interpolation
- How many vertex shaders executed and how many fragment shaders executed?

GPU Parallelism



GPU Parallelism

- GPU driver knows how to optimise use of cores.
- Some drawing may start before other drawing has finished
- **not in sync with code running on CPU**
- Great tool called APITrace for visually profiling CPU and GPU calls `http://apitrace.github.io/`



- Can also do GPU performance queries in code
- Shaders are hard to debug. Mixed-arms approach

GPU Hardware

Selected High-End Desktop Adaptors

Year	Model	Shader Cores	Memory (GB)
2009	GeForce GTX 295	$2\times$ 240	1.6
2009	Radeon HD 5970	1600	2
2011	GeForce GTX 590	$2\times$ 512	3
2011	Radeon HD 6990	1536	4
2012	GeForce GTX 690	$2\times$ 1536	4
2013	Radeon HD 8990	2304	6
2014	GeForce GTX Titan Black	2880	6

Mobile adaptors follow a similar trend but $10 \sim 60\%$ size ranges.

Teaching Shaders

- most courses are still teaching **fixed-function** graphics
- ’70s/’80s algorithms the same but practical side will never be used again
- **teach shaders**
- requires students build some knowledge of:
  - hardware (shader cores/bus/interpolation)
  - transformation pipeline and basic linear algebra
  - graphics algorithms to **write by-hand** (Phong, Catmull, etc.)
  - building and **linking libraries**
- requires relatively modern computers (but there are fall-backs)

Choosing an API

- BTH - use any graphics API; Direct3D, OpenGL, MS XNA
- TCD - upgraded from pre-shaders GL to OpenGL 4
- Apple - 2.1, 3.2 core forward / 4.1 core forward
- Laptops - 2.1. Almost all have support for shaders / software emulation.
- fall back to 2.1, 3.2 - wider support, similar shaders, no tessellation
- **Android ADK/NDK is a technical nightmare world**
- iOS is okay but there is licence/fee/certificate nonsense
- or...

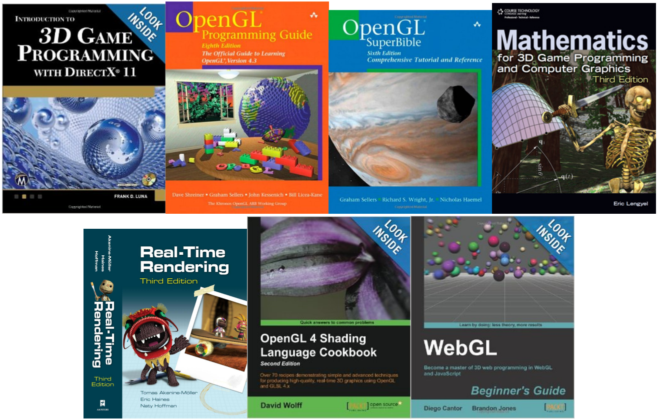
WebGL

- OpenGL in a browser. Based on OpenGL ES 2.0
- Shaders are the same as ES, and GLSL 1.20 (OpenGL 2.1)
- **Very quick to develop**
- Runs on just about everything
- JavaScript 'glue' instead of C/C++
- No mess with libraries / IDEs
- Eric Haines' (Autodesk) course https://www.udacity.com/course/cs291
- Three.js library (super-easy option) threejs.org
- Experimenting with shaders shadertoy.com
- My 48-hour challenge Dolphin Rescue

Problems

- Training staff - took me 2 years to get to a level of expertise with GL
- Lab support - programming basics, linking libraries, interface quirks
- *"It worked on my machine at home"*
- Supporting libraries for loading textures, opening windows, etc.
- Projects take a lot of student hours (think small or do groups)
- Will show embarrassing gaps in CS fundamentals (emergency C++ intro)
- Loading animated meshes

Resources of Note



- Students do not like the go-to mega-tomes. Exorbitant, unclear.
- I put all my simplified GL4 teaching material on-line http://antongerdelan.net/opengl
- Maths cheat-sheet pdf and simple code