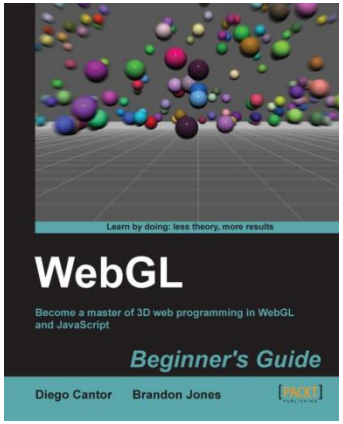
Intro to WebGL

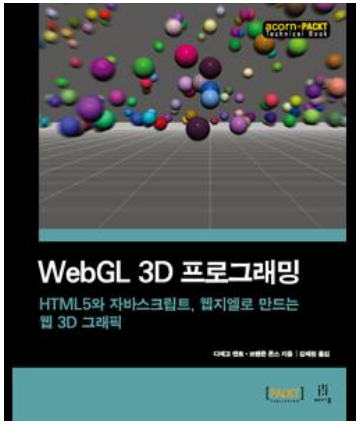
Diego Cantor

March 25th 2014

About me

- Software Engineer
- PhD Candidate in Biomedical Engineering
- Wrote the WebGL Beginner's Guide
- I spend my free time working on voxelent
- (WebGL framework)





Email: diego.cantor@gmail.com
Twitter: @diegocantor
Voxelent: http://www.voxelent.com

Goal

- Have a basic knowledge of what WebGL is and how it works
- Identify the data structures needed to perform parallel rendering in the GPU
- Learn about what is a vertex shader and a fragment shader
- Show you some cool demos and run a tutorial

WebGL

- WebGL is a Software Specification
  http://www.khronos.org
  - Drafted around May 2010
  - Version 1.0 March 3rd 2011
- Real-time rendering capabilities to internet browsers -> access to GPU
- Implemented in JavaScript
  - Flexible / Mutable, Is it slow?

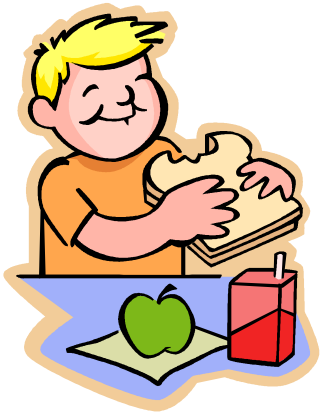
What about JavaScript performance

- JavaScript is slow
  - Interpreted language
- WebGL provides JavaScript bindings to GLSL
- GPU code runs fast!

WebGL is low level

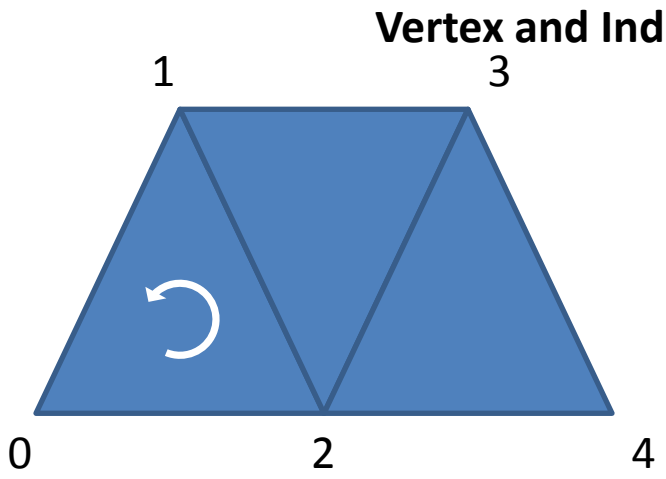
- Lots of functions to communicate with the GPU
- It works as a machine state
- Creating domain-specific applications is a lengthy process if used directly

Making bread is low level



Making a sandwich is high level

Defining Minimal Geometry



Index	Vertex Coordinates
0	(0,0)
1	(10,10)
2	(20,0)
3	(30,10)
4	(40,0)

coordinates

Vertex array = [0,0,10,10,20,0,30,10,40,0]

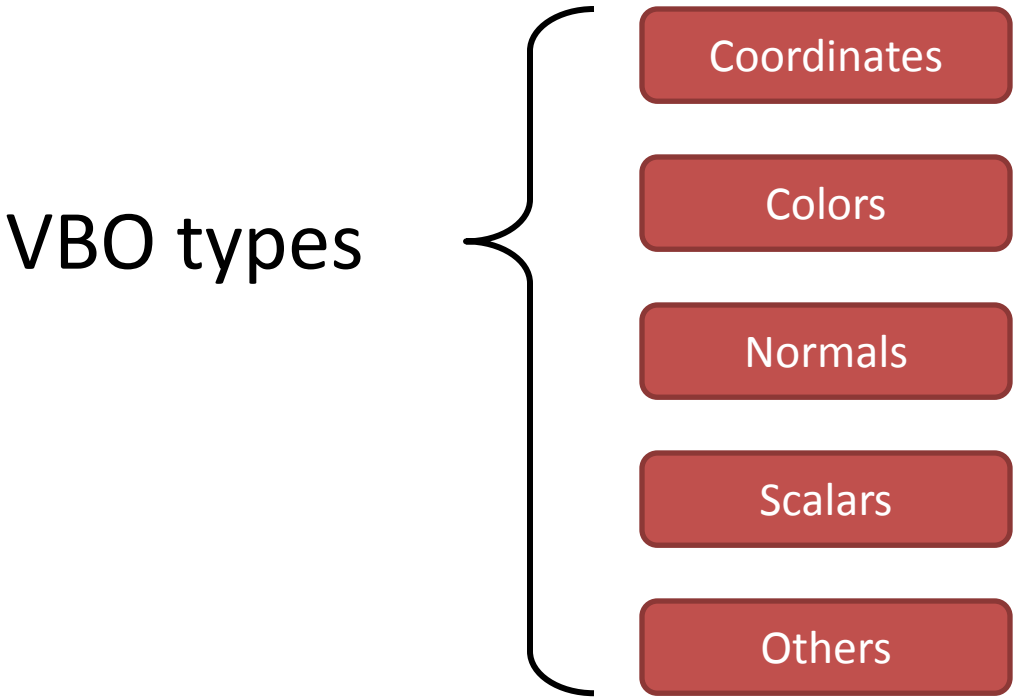
Index array = [0,2,1,1,2,3,2,4,3]

triangles

Vertex Buffer Object (Coordinates)

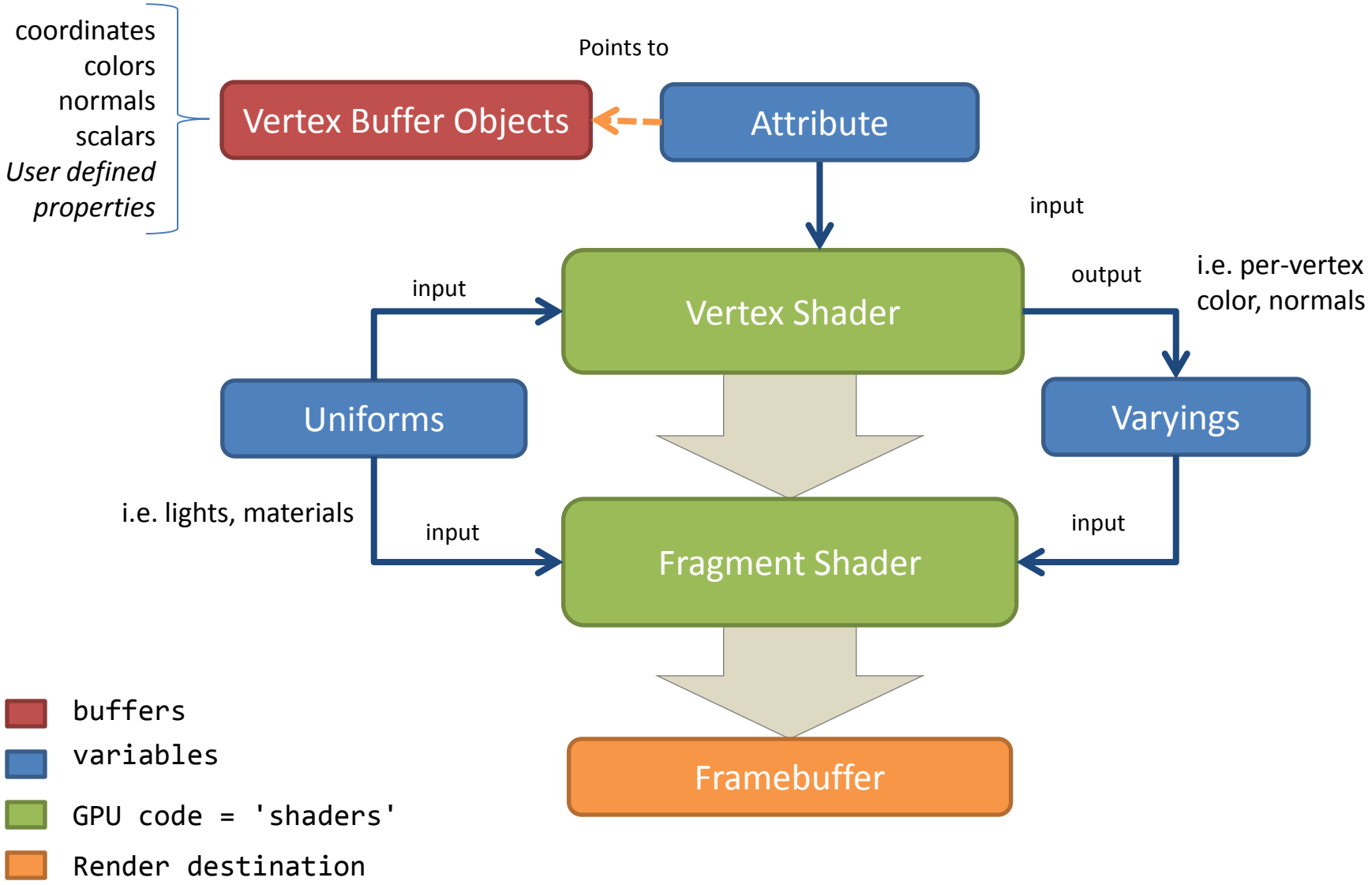
Index Buffer Object

Additional Vertex Elements



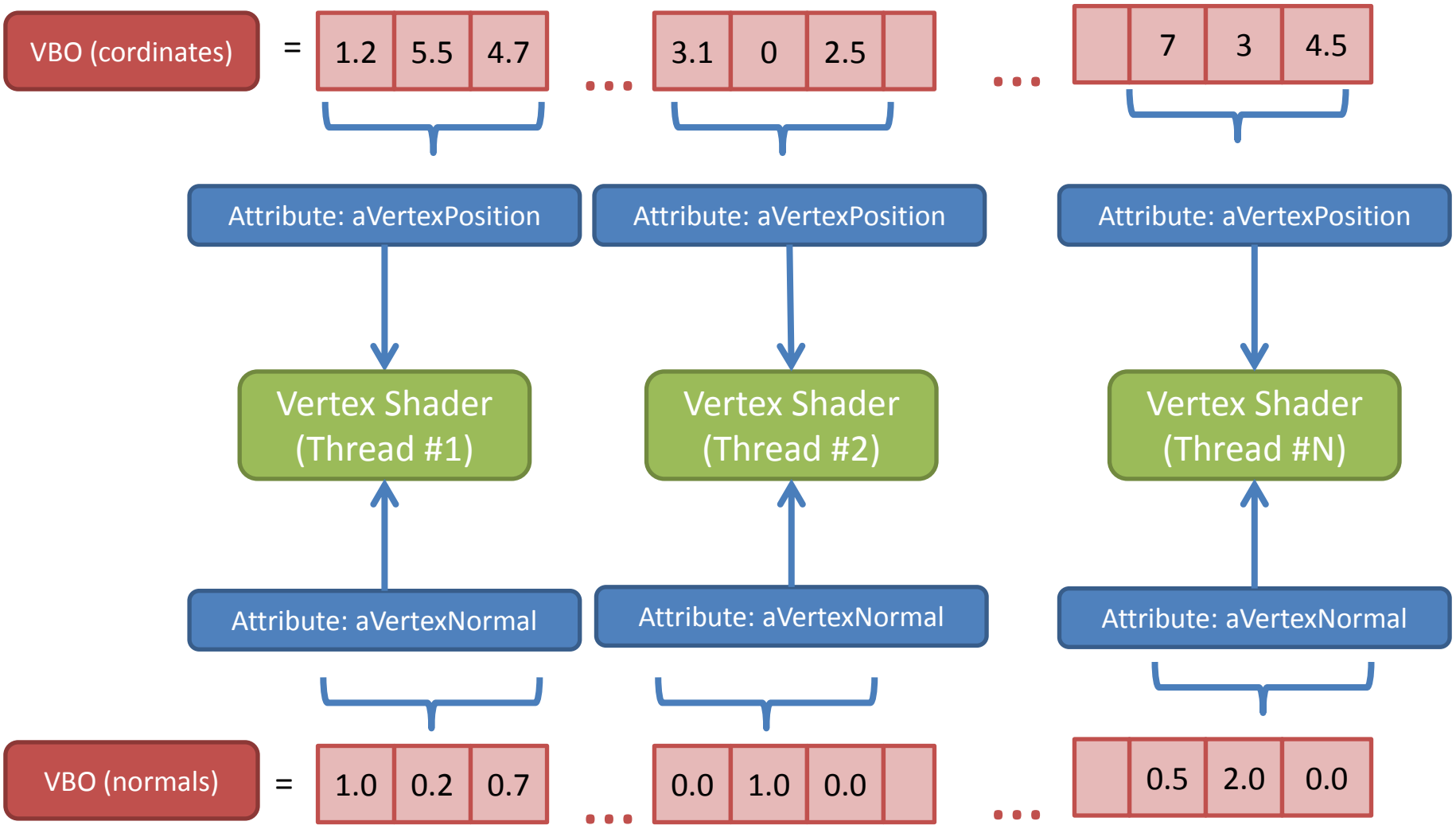
WebGL pipeline

from JavaScript to GPU



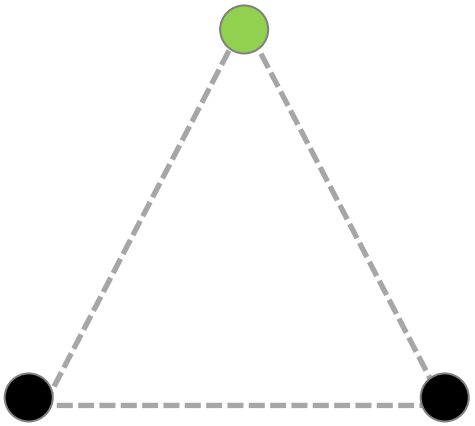
Parallel procesing in the Vertex Shader

The number of threads depends on the local GPU capabilities



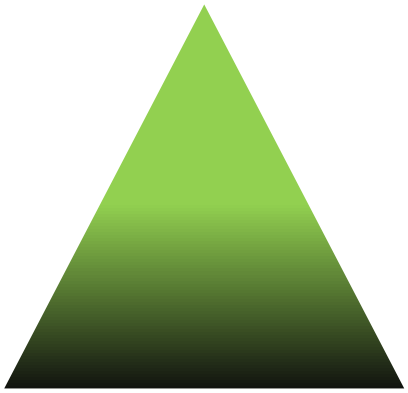
From vertices to fragments

In the Vertex Shader:



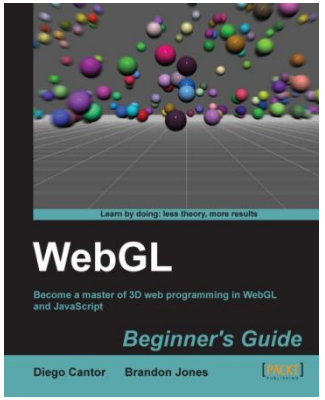
Vertex Coloring

In the Fragment Shader:



Pixel Coloring

Example



http://bit.ly/webglsquare

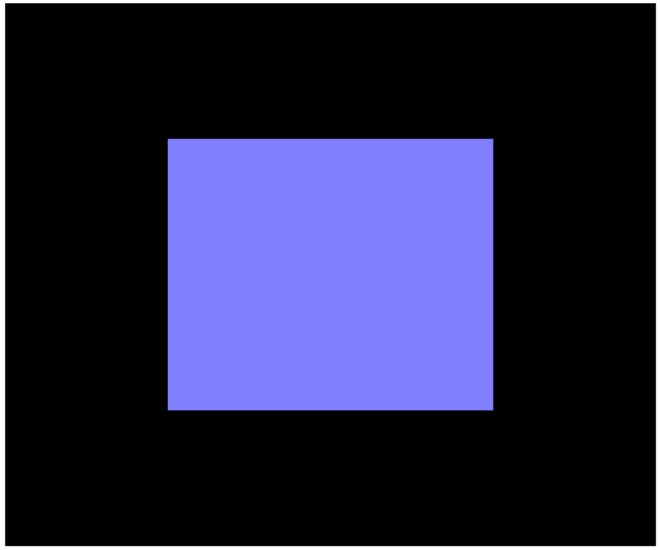
WebGL Beginner's Guide - Chapter 2

PACKT PUBLISHING

Rendering a Square

WebGL uses buffers to store and process vertex and index data. The mechanism is the same whether we are rendering a simple object like a square or a racing car as we will see later on.

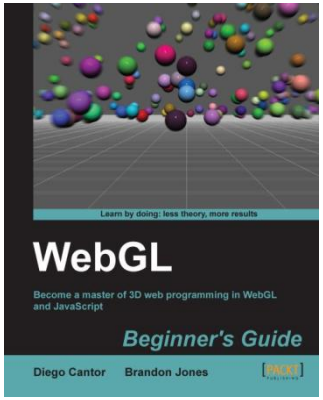
```
10.         var indices = []; //JavaScript array to store the indices of the square
11.         var vertices = []; //JavaScript array to store the vertices of the square
12.
13.
14.         /*
15.         * The program contains a series of instructions that tell the Graphic Processing Unit (GPU)
16.         * what to do with every vertex and fragment that we pass it. (more about this on chapter 3)
17.         * The vertex shader and the fragment shader together are called the program.
18.         */
19.         function initProgram() {
20.                 var fgShader = utils.getShader(gl, "shader-fs");
21.                 var vxShader = utils.getShader(gl, "shader-vs");
22.
23.                 prg = gl.createProgram();
24.                 gl.attachShader(prg, vxShader);
25.                 gl.attachShader(prg, fgShader);
26.                 gl.linkProgram(prg);
27.
28.                 if (!gl.getProgramParameter(prg, gl.LINK_STATUS)) {
29.                         alert("Could not initialise shaders");
30.                 }
31.
32.                 gl.useProgram(prg);
33.
34.                 //The following lines allow us obtaining a reference to the uniforms and attributes de
35.                 //This is a necessary step as the shaders are NOT written in JavaScript but in a
36.                 //specialized language called GLSL. More about this on chapter 3.
37.                 prg.vertexPosition = gl.getAttribLocation(prg, "aVertexPosition");
38.
```



WebGL JS | Vertex Shader | Fragment Shader | HTML

Full View

A more interesting example



http://bit.ly/webglcar

WebGL Beginner's Guide - Chapter 9 - Virtual Car Showroom

PACKT PUBLISHING

Customize your Car

Car

Select the car to load: Ford Mustang

Car Color:

Car Shininess: 0

Lights

	Ambient	Diffuse	Specular
far-left:	0.0	0.4	0.8
far-right:	0.0	0.4	0.8
near-left:	0.0	0.4	0.8
near-right:	0.0	0.4	0.8

Translate Lights

Camera

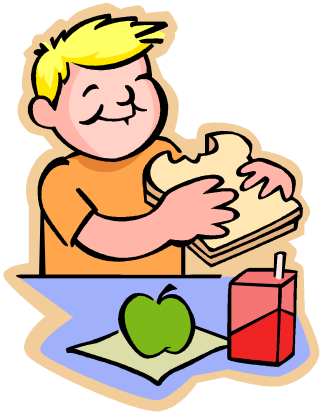
Home | Above | Front | Back | Left | Right

Zoom: Alt + Drag Floor Visible

WebGL is low level

- Lots of functions to communicate with the GPU
- It works as a machine state
- Creating domain-specific applications is a lengthy process if used directly

Making bread is low level



Making a sandwich is high level

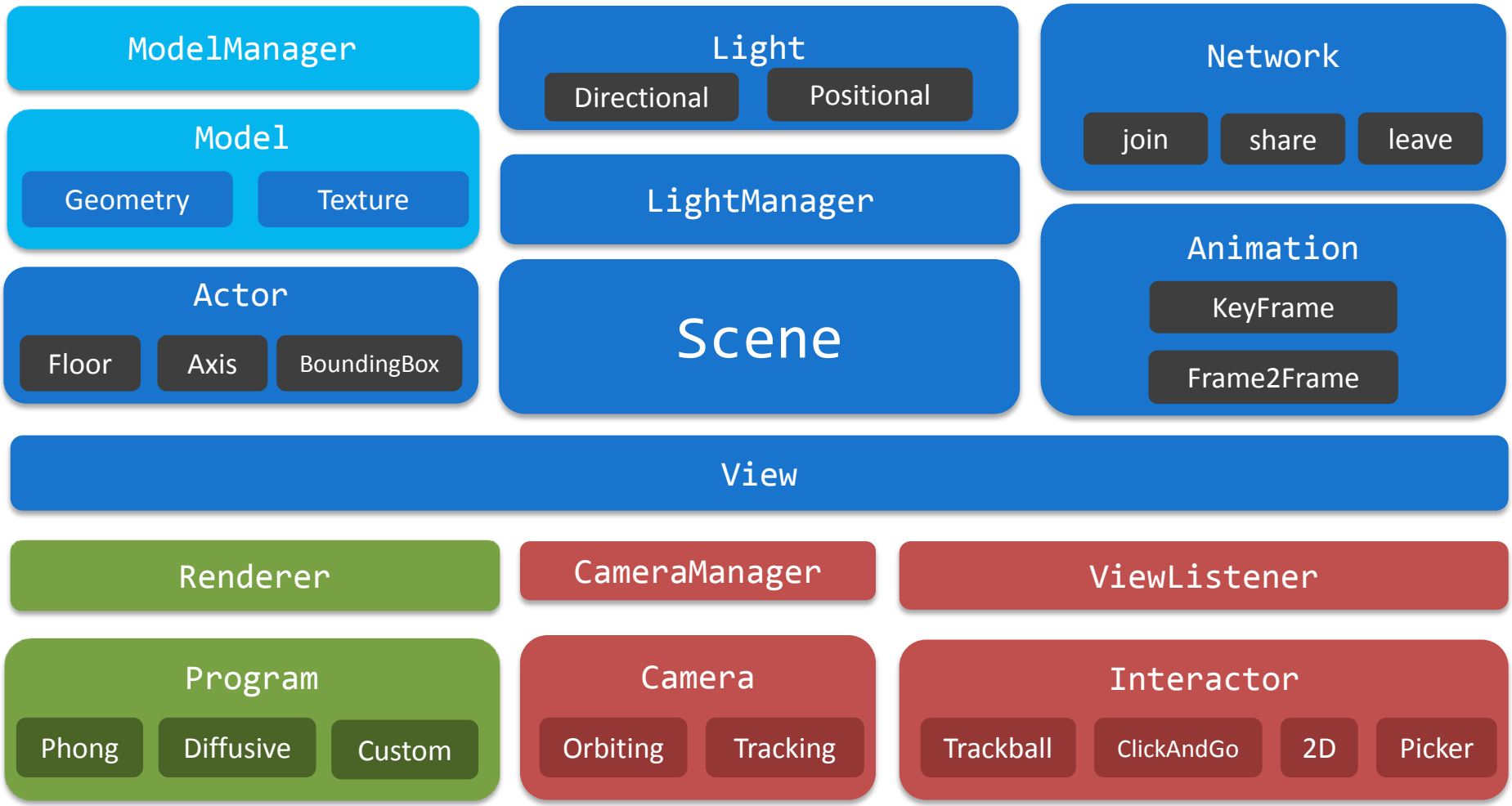
WebGL Libraries

- `Three.js`: minimalistic, simple platform.
- `Scene.js`: robust, tested, real apps.
- `GLSE`: game oriented, working on animation.

Voxelent

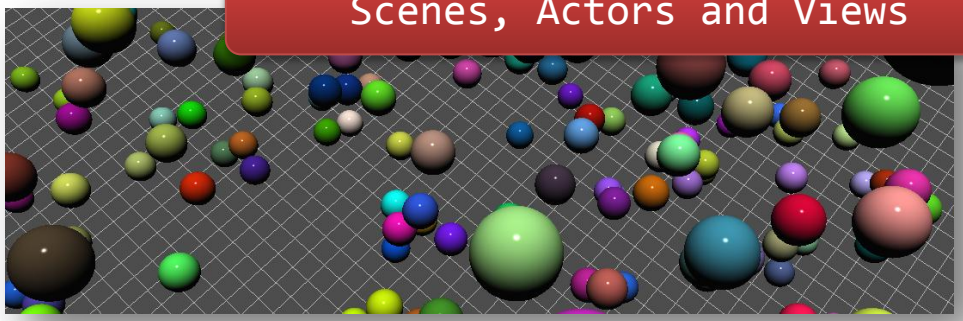
Robust, simple (easy to learn, easy to use) API with focus on software architecture

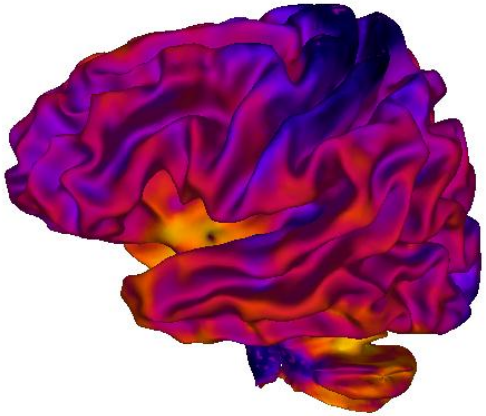
Voxelent


ModelManager
Model
Geometry
Texture
Actor
Floor
Axis
BoundingBox
Light
Directional
Positional
LightManager
Scene
Network
join
share
leave
Animation
KeyFrame
Frame2Frame
View
Renderer
Program
Phong
Diffusive
Custom
CameraManager
Camera
Orbiting
Tracking
ViewListener
Interactor
Trackball
ClickAndGo
2D
Picker

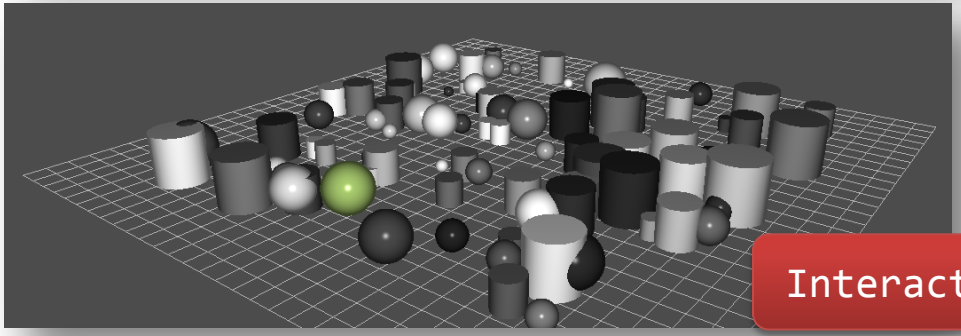
Demos

Scenes, Actors and Views

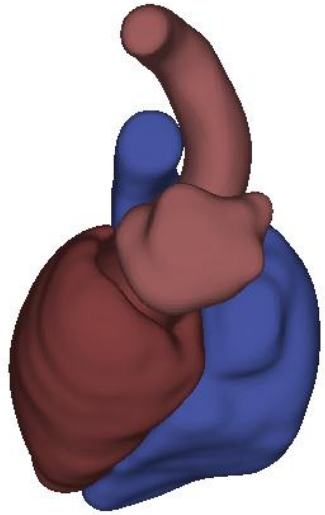


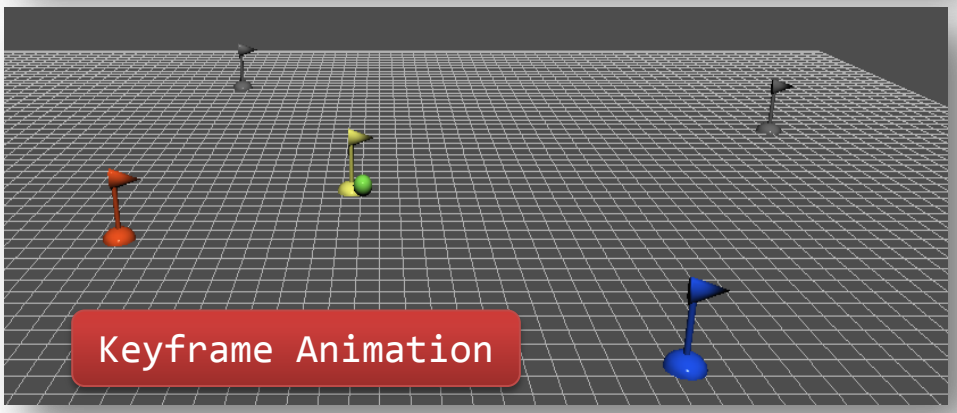


Models



Interaction





Keyframe Animation

Frame-to-Frame Animation

To Do

1. Run tutorial-1
2. Learn how to operate the camera
3. Load more objects
4. Run tutorial on camera landmarks

Thank you!

- The book source code and demos available:

http://Bit.ly/webgl-code

- voxelent tutorials at:

http://voxelent.com/tutorials/

I can be reached by email at:

diego.cantor@gmail.com