# CPSC 314
# Computer Graphics

Dinesh K. Pai

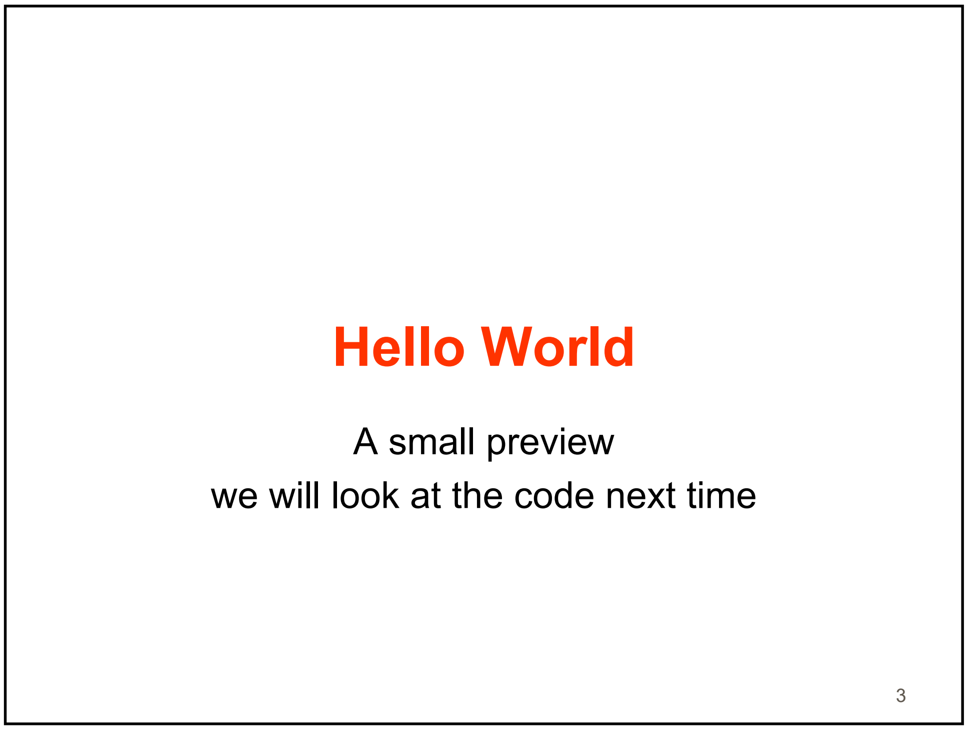
A first look at the Graphics Pipeline
and WebGL

Many slides courtesy of Min Hyuk Kim, KAIST and Steven Gortler, Harvard

## Announcements

- Today:
  - Introduction to the OpenGL Graphics Pipeline
  - Intro to programming with GLSL, WebGL, Three.js (Assignment 1)
- Assignment 1 out on Friday
  - Details will be available on Canvas

2

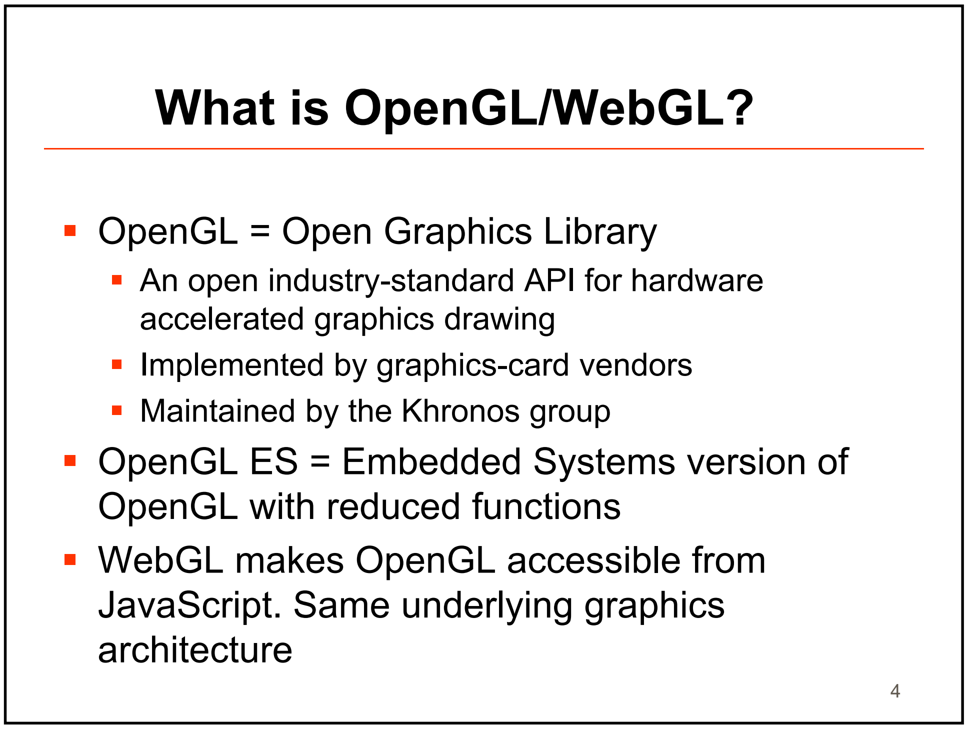


## Hello World

A small preview

we will look at the code next time

3



## What is OpenGL/WebGL?

- OpenGL = Open Graphics Library
  - An open industry-standard API for hardware accelerated graphics drawing
  - Implemented by graphics-card vendors
  - Maintained by the Khronos group
- OpenGL ES = Embedded Systems version of OpenGL with reduced functions
- WebGL makes OpenGL accessible from JavaScript. Same underlying graphics architecture

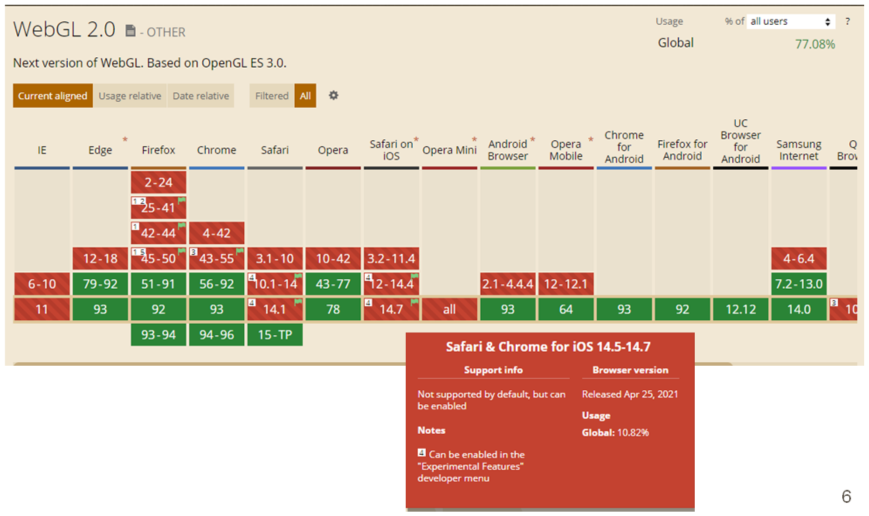
4

## Which WebGL?

- WebGL 1.0 is based on OpenGL ES 2.0
  Now supported in almost all browsers
- WebGL 2.0 is based on OpenGL ES 3.0
  Finalized in 2017. Supported in Chrome and Firefox, not yet in Safari, etc.
- We'll plan to use some features of WebGL 2.0 this term, so please use one of the compatible browsers. See
  https://caniuse.com/#feat=webgl2

5

## WebGL 2.0 on Sep 10, 2021



WebGL 2.0 - OTHER

Usage % of all users ?
Global 77.08%

Next version of WebGL. Based on OpenGL ES 3.0.

Current aligned | Usage relative | Date relative | Filtered | All

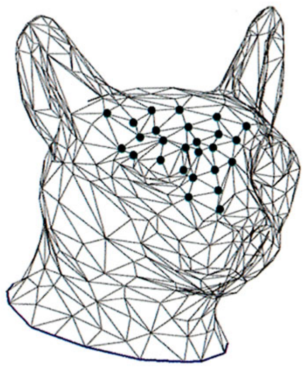
| IE | Edge | Firefox | Chrome | Safari | Opera | Safari on iOS | Opera Mini | Android Browser | Opera Mobile | Chrome for Android | Firefox for Android | UC Browser for Android | Samsung Internet | Q Brow |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 🟥 2-24 | | | | | | | | | | | | |
| | | 🟥 [1][2] 25-41 | | | | | | | | | | | | |
| | | 🟥 [1] 42-44 | 🟥 4-42 | | | | | | | | | | | |
| | 🟥 12-18 | 🟥 [1][5] 45-50 | 🟥 [3] 43-55 | 🟥 3.1-10 | 🟥 10-42 | 🟥 3.2-11.4 | | | | | | | 🟥 4-6.4 | |
| 🟥 6-10 | 🟩 79-92 | 🟩 51-91 | 🟩 56-92 | 🟥 [4] 10.1-14 | 🟩 43-77 | 🟥 [4] 12-14.4 | | 🟥 2.1-4.4.4 | 🟥 12-12.1 | | | | 🟩 7.2-13.0 | |
| 🟥 11 | 🟩 93 | 🟩 92 | 🟩 93 | 🟥 [4] 14.1 | 🟩 78 | 🟥 [4] 14.7 | 🟥 all | 🟩 93 | 🟩 64 | 🟩 93 | 🟩 92 | 🟩 12.12 | 🟩 14.0 | 🟥 [3] 10 |
| | | 🟩 93-94 | 🟩 94-96 | 🟩 15-TP | | | | | | | | | | |

**Safari & Chrome for iOS 14.5-14.7**

| Support info | Browser version |
|---|---|
| Not supported by default, but can be enabled | Released Apr 25, 2021 |
| **Notes** | **Usage** |
| [4] Can be enabled in the "Experimental Features" developer menu | **Global:** 10.82% |

6

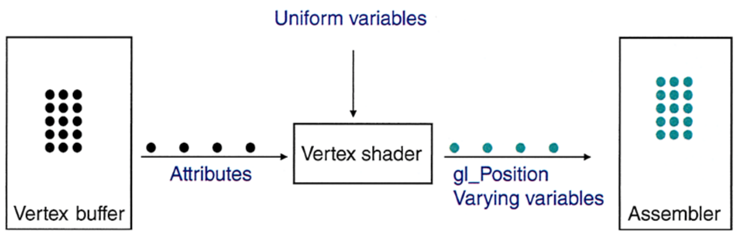
## OpenGL Pipeline



- Reference:
  Textbook Chapter 1
- Shapes are “discretized” into primitives: triangles, line segments, …
- We’ll focus on triangles most of the time
- Triangles defined by positions of their vertices

7

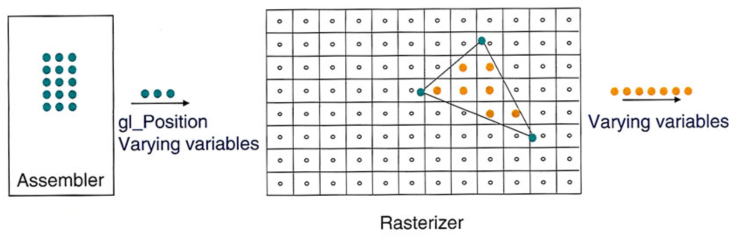
## OpenGL Pipeline: Vertex Shader



- Vertices are stored in a vertex buffer.
- When a draw call is issued, each of the vertices passes through the vertex shader
- On input to the vertex shader, each vertex (black) has associated attributes.
- On output, each vertex (cyan) has a value for gl_Position and for its “varying” variables (in WebGL 2, called “out/in”).

8

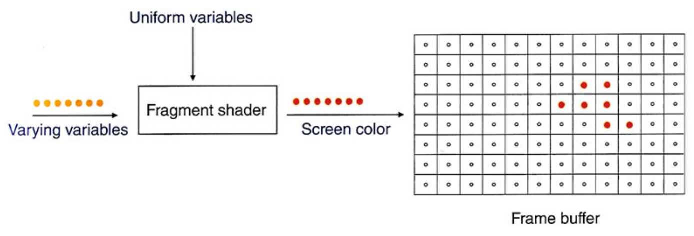
## OpenGL Pipeline: Rasterization



- The data in gl_Position are used to place the three vertices of the triangle on a virtual screen.
- The rasterizer figures out which pixels (orange) are inside the triangle and interpolates the varying variables from the vertices to each of these pixels.

9

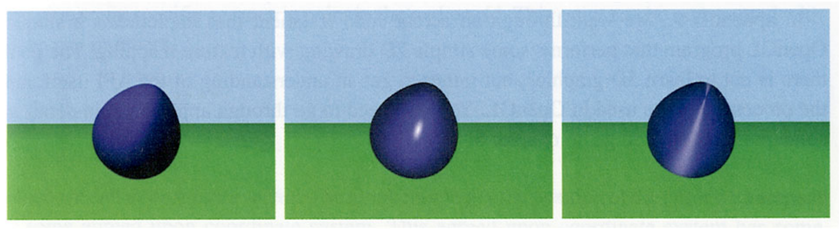
## OpenGL Pipeline: Fragment Shader



- Each pixel (orange) is passed through the fragment shader, which computes the final color of the pixel (pink).
- The pixel is then placed in the framebuffer for display.

10

## OpenGL Pipeline: Fragment Shader



- By changing the fragment shader, we can simulate light reflecting off of different kinds of materials.

## A brief look at Three.js

- A high level library that can use WebGL for rendering
  - Can also use the basic HTML5 canvas for simple things
- Setup is much easier compared to WebGL
- Implements “scene” and “mesh” abstractions
- Mesh ≅ geometry + material properties
  - Warning: this usage of “mesh” is non-standard
- Scene contains a hierarchy of mesh objects
- Render a scene using a Camera

## Demo

https://threejs.org/editor/

## Summary

- What is OpenGL/WebGL?
  - A software interface that allows a programmer to communicate with the graphics hardware
  - A programming interface for rendering 2D and 3D graphics
  - A cross-language multi-platform API for computer graphics
- What is Three.js
  - A high level JavaScript library that provides easy setup and access to WebGL

## Important Point!

- In this course we will use WebGL and Three.js to understand the principles of 3D computer graphics
- This is NOT a course about programming with WebGL and all the intricacies of Three.js
- Our primary focus will be on writing small shaders in GLSL to implement the key concepts of a computer graphics application

15

## Preparation for next class

- Explore Threejs.org website's documentation, esp. https://threejs.org/docs/index.html#manual/en/introduction/Creating-a-scene
- Read Chapter 2 of textbook

16