WebGL and the Three-D Internet

Hugh Fisher
CECS ANU
2013
Overview

The Vision: origins of cyberspace

The rendering problem

VRML: the grand attempt

XAML: the Microsoft web gets 3D

WebGL: what it can do

The Future?
The Vision

Prototypes in 1984

VR
Ivan Sutherland, USAF

Internet
ARPAnet running TCP/IP

MUDs and MOOs
Shared worlds with hundreds of users

All very expensive and/or exclusive
1984: “Neuromancer” by William Gibson

Keystone of the cyberpunk science fiction genre

Defined cyberspace

Cyberspace ... A graphic representation of data abstracted from the banks of every computer in the human system ... Lines of light ranged in the nonspace of the mind, clusters and constellations of data ...
The 1990s VR Boom

VR no longer experimental
SGI purple workstations
VPL DataGlove, EyePhone

“Garage VR”
386 PCs with 3DFX Voodoo
Miniature LCD TV displays

World Wide Web
Internet beyond universities
The web grew exponentially
“Unthinkable complexity” now reality

The web was interactive
And hypertext was new
No physical equivalent, unlike email

The web user interface was primitive
Forms and round trip per page
“IBM 3270 with pictures”

What should it become?
The Other Plane

The road not taken

1981 “True Names” by Vernor Vinge

A world wide computer networked virtual reality
Magical metaphors: wizards and spells

Why didn’t this succeed?
Fantasy outsells science fiction!
Client or Server Side?

Scene must be rendered
Model is geometry, textures, lights, materials, ...
Displayed as 2D raster image
As seen from viewpoint, digital camera

In realtime
Unlike CGI, don’t control viewpoint

Where to render?
Model stored on server(s)
Displayed on client device
Client Side

Google Earth, World of Warcraft
Server sends 3D scene to client
Current room at least
Can use LOD techniques progressive download
Client renders with own GPU
3D accelerator before 2000

No round trip for interaction
Faster response to keyboard / mouse
Unless multi user
Disadvantages

Must write client side application
MacOS vs MS Windows vs Android ...

Must allow for different capabilities
RAM? Disk space? Screen resolution?

Must allow for different GPUs
NVIDIA vs ATI vs PowerVR ...
Shader Model 3? 4? 5?
Server Side

Server renders scene
Sends 2D images to client
Client tells server what resolution required

Client much simpler
Both hardware and software
Anything that can receive streamed video
Disadvantages

Round trip delay on all input
Even for single user case

Needs constant downstream bandwidth
Client side can handle slow initial download

Can’t buffer stream to avoid jitter
User will not accept being 10 seconds behind

Can’t use asymmetrical compression
MPEG etc not designed real time encoding

Extra copy back from GPU
Client side is CPU -> GPU -> display
Server side CPU -> GPU -> CPU -> network
Even Worse...

Server side 3D does not scale
Horrible context swap overhead
All textures and scene data stored in GPU RAM
Must reload for different scene
No virtual memory

Very deep GPU pipelines
Expect to run to completion on frame
Not designed for many short tasks

Need (almost) one GPU per user
For peak load, not just average
Electricity, heat dissipation
Server side doesn’t scale
Client side is hard
(Cross platform applications always hard)

Web browser is "universal" client
Handles text and 2D (raster) graphics
Create HTML, no need for new applications
Example: E-book readers

Extend web browsers to 3D?
Create 3D scenes, no new applications?
VRML

Virtual Reality Modelling Language

Tony Parisi and Mark Pesce
Developed as open standard

VRML
First version static scenes

VRML 2 / 97
Animation and event handling
Javascript
Based on SGI Inventor

High level OpenGL library

Hierarchical scene graph

Nested transformations

DEF nodes for multiple parents

Geometry

High level: spheres, cylinders, ...

Low level: triangles, indexed face sets, ...

OpenGL 1 Gouraud shading

Point, directional, spot lights

Ambient, diffuse, specular material

Texture maps
Built in navigation
Walk, fly, or third person
Programmable viewpoints

Events and routes
User input, timers
Change transformations, visibility, properties
VRML 2: JavaScript

3D picking
Clickable nodes
Menus and buttons
Generate events
Integrated into web
Branches could be loaded from URLs
Single scene from multiple servers

Nodes could be anchors
Click on node, jump to new location
Including existing web sites

Superset of existing web
What Went Wrong?

Technology not ready?
Late 1990s PC were good at 3D

Internet bubble burst?
Web kept growing

Not multiuser?
Special viewers like Blaxxun
Maybe in VRML 3?
A better solution to unthinkable complexity
VRML is not dead

Good interchange format
UTF-8, regular syntax
Matches OpenGL 1.x, DirectX 7 to 9

Good for special purpose programs
CSIRO haptic surgery programs
New viewers and plugins
Recommend Cortona 3D
**XAML**

**Windows Presentation Foundation**
Major redesign and rewrite of MS Windows APIs
Based on .NET managed runtime
2D / 3D for modern GPUs

**XAML**
Dynamic web programming, Microsoft style
WPF for web browsers
Could run on MacOS, Linux

**Intended for viewing 3D models**
Not VR or games
Extended XML syntax
Need arrays of floating point

Hierarchical scene graph
Nested transformations
Reference nodes with multiple parents

Geometry
No high level spheres, cylinders, etc
Low level: triangles, indexed face sets
Auto calculation of surface normals, tex coords

No GPU shaders
Point, directional, spot lights
Ambient, diffuse, specular material
Texture maps
Events and routes
User input, timers
Change transformations, visibility, properties

Integrated 2D text, graphics
Use 2D content as texture maps

Missing
Built in navigation
3D picking
Anchor nodes
.NET web applications
Like Java, browser downloads and runs

XAML models loaded into app
More complex event handling
3D picking
Jump to new locations

What went wrong?
End of plugins
New era of mobile web
iPhone: no plugins
Obsoleted Flash, Silverlight as collateral damage

MS internal politics
.NET, WPF out of favour
MS very enthusiastic about HTML 5

No future for XAML?
Backwards compatibility only in Windows 8
No new versions
Not recommended for developers
WebGL

OpenGL API for JavaScript
SVG is API for 2D vector graphics
WebGL for 3D
New typed arrays for JS
VRML, XAML describe scenes
Retained mode
Data: can be stored, copied, exchanged
WebGL is code
Immediate mode
Programs, not documents
OpenGL 2 ES “Embedded Systems”

iPhone
Android
Consoles

3D assembly language

Points, lines, triangles
Texture maps
GPU shaders

That’s it!
Scene graph
No nested transformations
Can’t attach scenes from other servers

High level geometry
No spheres, Beziers, quads

Built-in lights, materials
Have to write shaders

Advanced GPU capabilities
Geometry or tessellation shaders
Instanced rendering

3D picking
Extensions?
Traditional way to extend OpenGL
Runtime check for new capabilities
Yes ...
WebGL includes API for extensions
... No
Only official WebGL extensions
Not OpenGL V3/4
High entry barrier
OpenGL 2 API
and GLSL

Toolkits
“You can build exactly the high level functions you want”
Worked with jQuery for AJAX
Hasn’t happened with desktop OpenGL 3/4
three.js, spider.js, ...
Does it have to be this hard?

Lines of code for shaded cube in web page
Single light source, diffuse material, no texture mapping

1997 VRML  14

2007 XAML  76

2012 WebGL 2,277
Availability (Dec 2012)

MacOS Safari
Most desktop Firefox
Most desktop Chrome
Some Android

No iThing
Capability is there, Apple haven’t enabled

No Internet Explorer
Plugins available
WebGL Growth

"60% of all statistics are made up on the spot"
WebGL will succeed?

Networks are better
Much more bandwidth available
Less need for compression, LOD schemes

GPUs are better
iPhone more powerful than SGI workstations

The web is much, much, larger
Soon a billion WebGL-capable smartphones
1% market will exceed best selling console games
Internet 2021?

Google Glass

Microsoft Kinect
learningwebgl.com

cs.anu.edu.au/~hugh.fisher/webgl/

Questions?