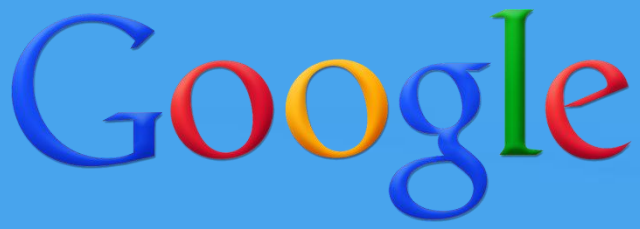
WebGL Seminar: O3D

Alexander Lokhman
Tampere University of Technology

What is O3D?

- O3D is an open source **JavaScript API** for creating rich, interactive 3D applications in the browser
- Created by **Google** and mantained by **community**
- Provided under **BSD License**
- Last release: **0.1.42** from 5.10.2009



O3D Extends

- System
  O3D provides a **browser plug-in** that adds graphics capabilities inside standard web browsers on different platforms

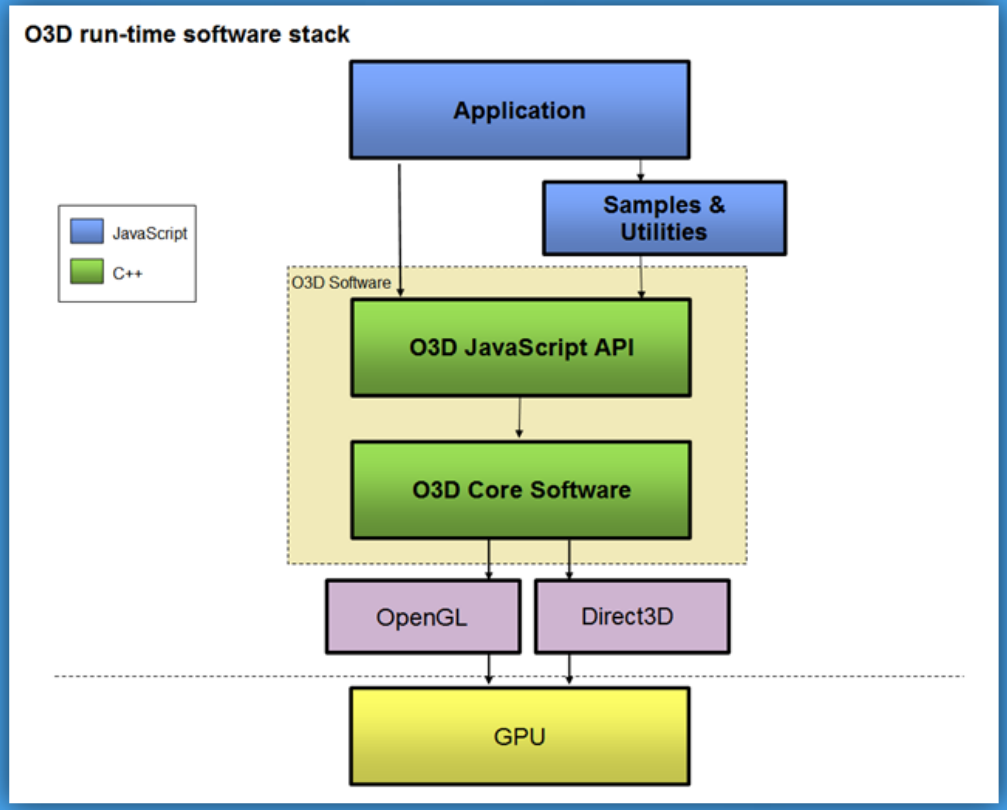
- Content
  Web content in the form of HTML, image files, and video files. Provided information on how to create a file **converter and loader** for any 3D content

- Code
  Application JavaScript code with **API for 3D graphics**. It uses standard JavaScript event processing and callback methods

Features

- **Retained** mode API
  - Calls describe objects and hierarchies
  - High-level information about objects is retained between frames (no need to redraw objects every frame)
- Extensive **access to** the **graphics hardware**:
  - User-supplied shader programs (HLSL/Cg)
  - Multi-pass rendering and render-to-texture
  - Render states
- **Image loaders** for common formats
- **Animation** and **Skinning**
- Flexible asset import path based on **JSON**

Architecture



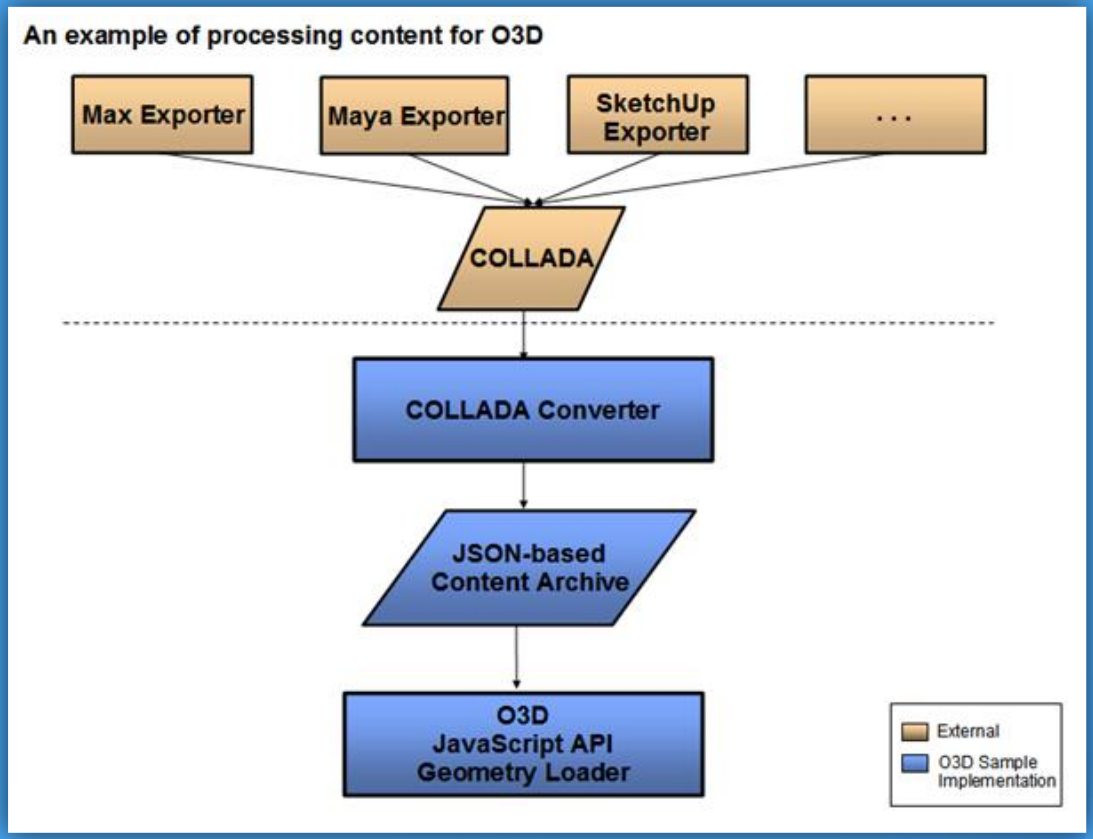
Major components:

- O3D JavaScript **application**
- JavaScript **Utilities** provided as sample code
- **JavaScript API**, which contains classes and functions used in application

Importing Content

- O3D can draw **primitives** for creating shapes directly within application

- It also provides a sample **COLLADA Converter**, which can be used to import files in COLLADA format that is supported by SketchUp, 3ds Max, and Maya



Loading Assets

- O3D can load **zipped archive files** (.tar.gz) containing:
  - Text (JSON)
  - Image files (for textures)
  - Binary buffers (for mesh data, animation, etc)
- Application can access file content data and make regular API calls to create **O3D objects** from it
- Provides **a lot more flexibility** over a fixed format

Development Summary

- O3D is a **low-level 3D graphics API**. Programming with it requires some level of understanding of 3D graphics principles
- Provided **JS utility libraries** make programming easier
- Over time higher-level libraries **will be developed**

Creating Simple Application

- HTML Basics

```
<script src="o3djs/base.js"></script>
<script>
  o3djs.require('o3djs.util');
  o3djs.require('o3djs.math');
  ...
  window.onload = function() { init(); }
</script>
...
<div id="o3d" style="width: 640px; height: 480px">
  <!-- DISPLAY BLOCK -->
</div>
```

Creating Simple Application (cont'd)

- Initialization and creating a simple object

```
o3djs.require('o3djs.simple');
// Create the client area:
function init() {
  o3djs.util.makeClients(initStep2);
}
// Initialize application:
function initStep2(clientElements) {
  var o3dElement = clientElements[0];
  var g_simple = o3djs.simple.create(o3dElement);
  var g_cube = g_simple.createCube(50);
  ...
}
```

[DEMO]

Creating Simple Application (cont'd)

- Experimenting with object

```
// Translate the cube:
g_cube.transform.translate(0, 0, -50);
// Make the cube red:
g_cube.setDiffuseColor(1, 0, 0, 1);
// Use texture:
g_cube.loadTexture("http://someplace.org/somefile.jpg");
// Move the camera:
g_simple.setCameraPosition(200, 100, -50);
// Move the camera's target:
g_simple.setCameraTarget(0, 10, 0);
// Change the field of view:
g_simple.setFieldOfView(30 * Math.PI / 180);
```

Creating Simple Application (cont'd)

- Adding keyboard events

```
var g_keyDown = []; // which keys are down by key code

function initStep2(...) {
  ...
  o3djs.event.addEventListener(o3dElement, 'keydown',
   onKeyDown);
  o3djs.event.addEventListener(o3dElement, 'keyup',
   onKeyUp);
}

function onKeyDown(e) { g_keyDown[e.keyCode] = true;  }
function onKeyUp(e)   { g_keyDown[e.keyCode] = false; }
```

Creating Simple Application (cont'd)

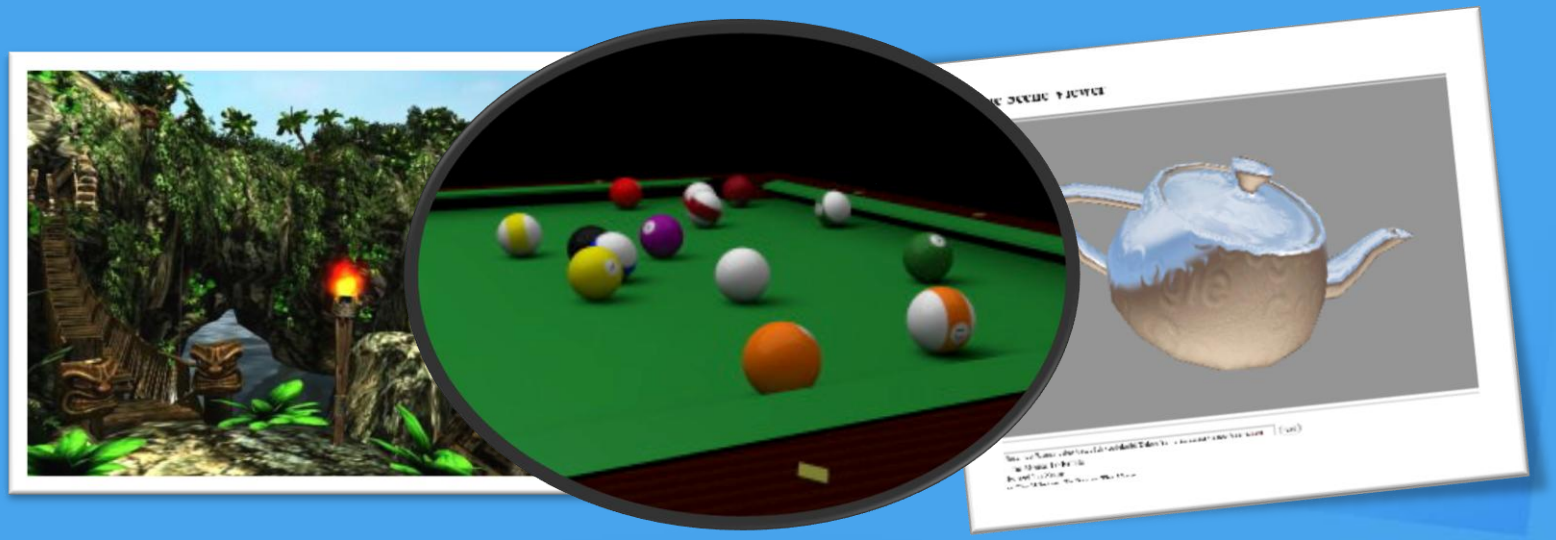
- Loading a character

```
function initStep2(...) {
  ...
  var g_client = o3dElement.client;
  var g_pack = g_client.createPack();
  var g_globalParams =
   o3djs.material.createAndBindStandardParams(g_pack);
  var transform = g_pack.createObject('Transform');
  var playerPack = g_client.createPack();
  o3djs.scene.loadScene(g_client, playerPack, transform,
   "assets/character.o3dtgz", initStep3);
}
function initStep3(playerPack, playerParent, exception) {
  // Callback function of loading scene method
}
```

[DEMO]

Demos

- Basics and typical applications:
  http://code.google.com/apis/o3d/docs/samplesdirectory.html
- WebGL implementation of O3D:
  http://code.google.com/p/o3d/wiki/Samples



Summary

- The main advantage O3D has over alternative desktop or console based 3D rendering engines is that O3D may **load**, **render**, and **transform models and** their respective **textures dynamically**, using AJAX and/or COMET in real-time.
- Traditional **compilation** of source code, application resources, and object libraries **is no longer necessary**, since all of these aspects are loaded in real-time.

http://en.wikipedia.org/wiki/O3D

Changing Direction

- O3D API was originally shared at an early stage as a browser plug-in but has **evolved into** a **JavaScript library** on WebGL.
- ”Previously, Google believed that JavaScript wouldn’t be fast enough for a low level API like OpenGL, instead preferring an O3D plug-in.”

  - GNT Media. 3D Web: O3D oriented towards WebGL (10.5.2010)

Links

- WebGL implementation of O3D
  http://code.google.com/p/o3d/
- O3D Plug-in API
  http://code.google.com/apis/o3d/
- The O3D API Blog
  http://o3d.blogspot.com/
- Source code repository
  http://src.chromium.org/viewvc/chrome/trunk/src/o3d/

Questions?

Thank you for attention!