Google Developer Day2009

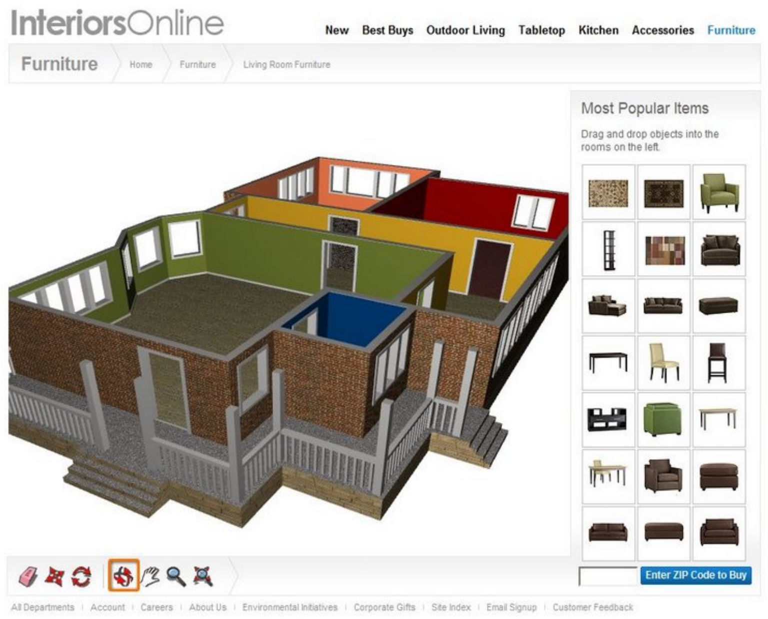
O3D: Adding Interactive 3D Content to your site.

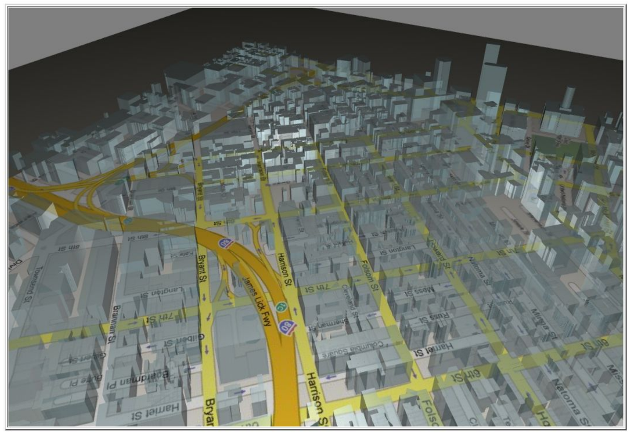
Gregg Tavares

Google Developer Day2009

Introducing O3D

- O3D is a JavaScript API for doing hardware accelerated 3D graphics in a web page.





Design Goals

- Expressive and fast
  - Capable of rendering a complex modern game
- Flexible
  - Make no assumptions about target apps
- Portable
  - Write once, run anywhere

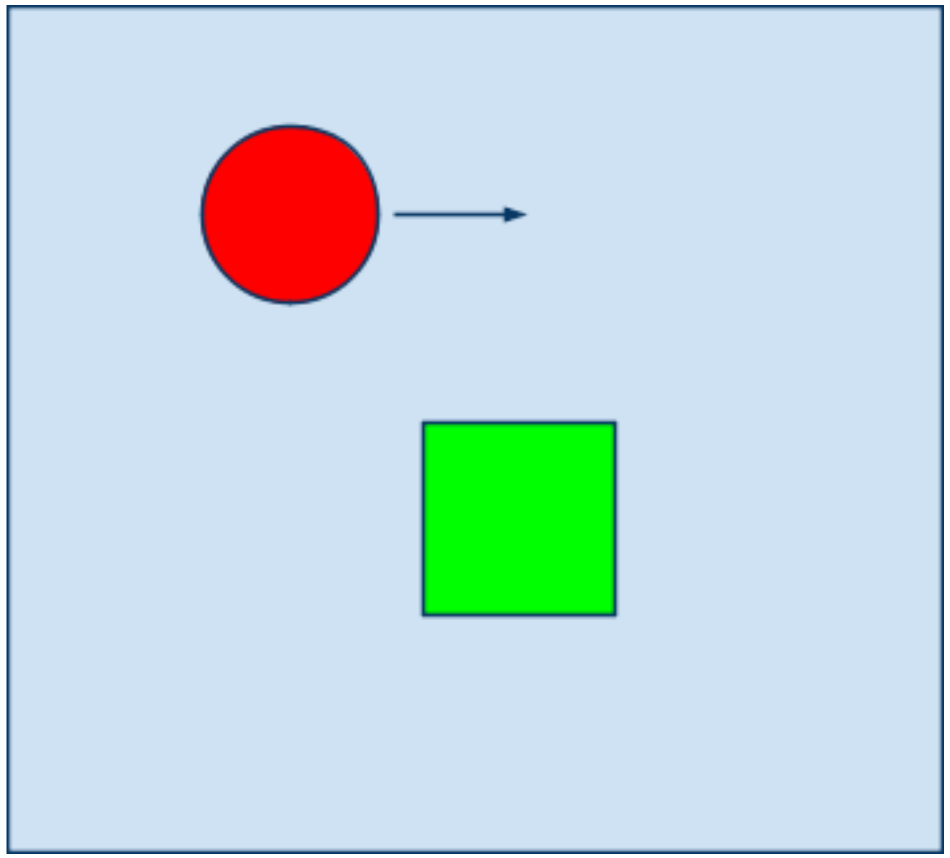
O3D Feature Highlights

- 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
- Embedded JS engine using Google Chrome's V8
- Flexible asset import path based on JSON

Immediate vs Retained Mode APIs

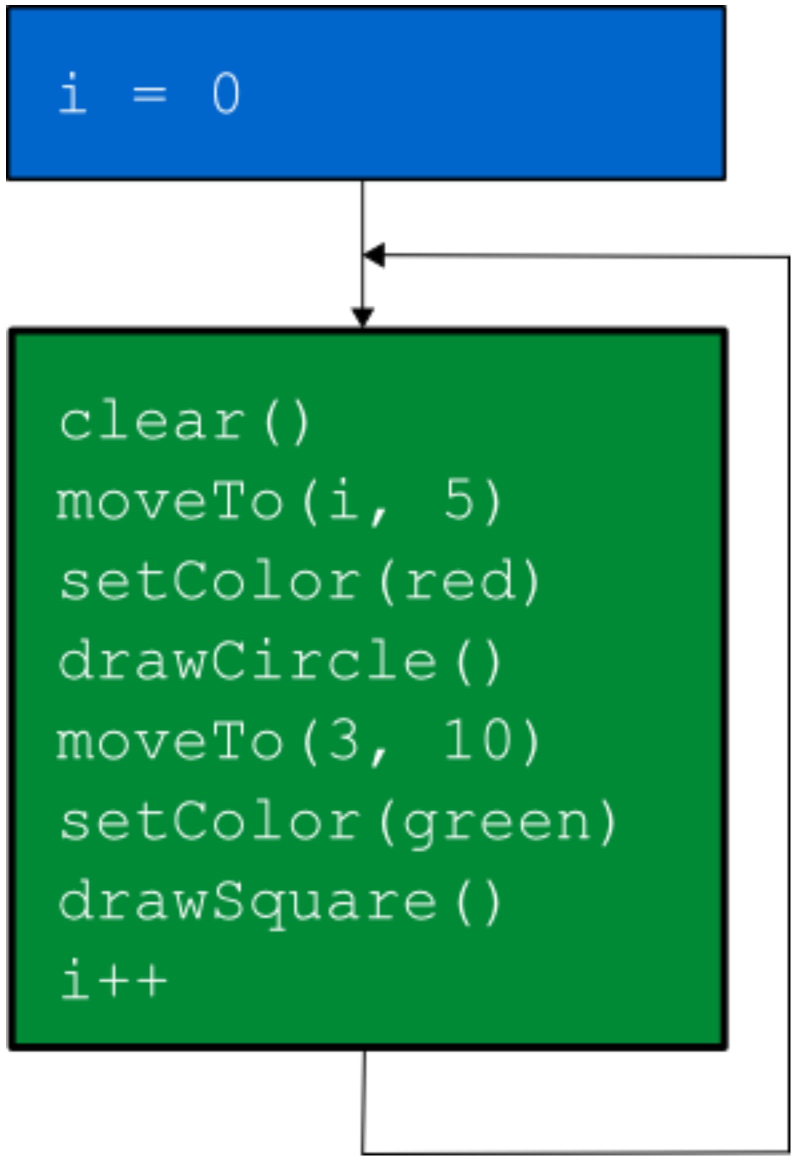
- Immediate mode:
  - Graphics commands issued execute "immediately"
  - All objects need to be drawn every frame
  - Examples: Direct3D, OpenGL, HTML Canvas
- Retained mode:
  - Calls describe objects and hierarchies
  - High-level information about objects is retained between frames
  - Examples: Most game engines, SVG, HTML DOM

Immediate vs Retained mode example



Google Developer Day2009

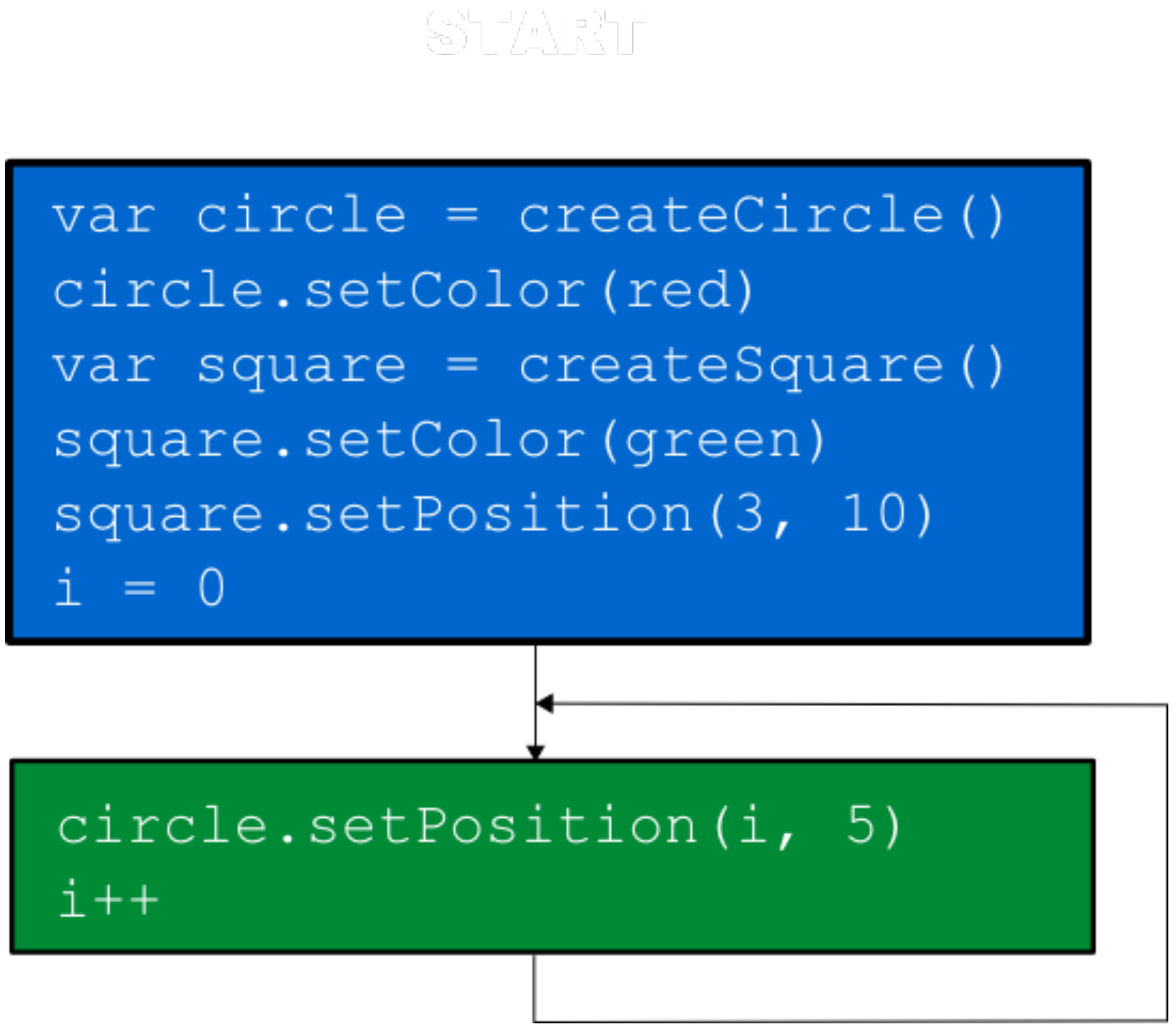
Immediate Mode



```
i = 0
```

```
clear()
moveTo(i, 5)
setColor(red)
drawCircle()
moveTo(3, 10)
setColor(green)
drawSquare()
i++
```

Retained Mode



```
var circle = createCircle()
circle.setColor(red)
var square = createSquare()
square.setColor(green)
square.setPosition(3, 10)
i = 0
```

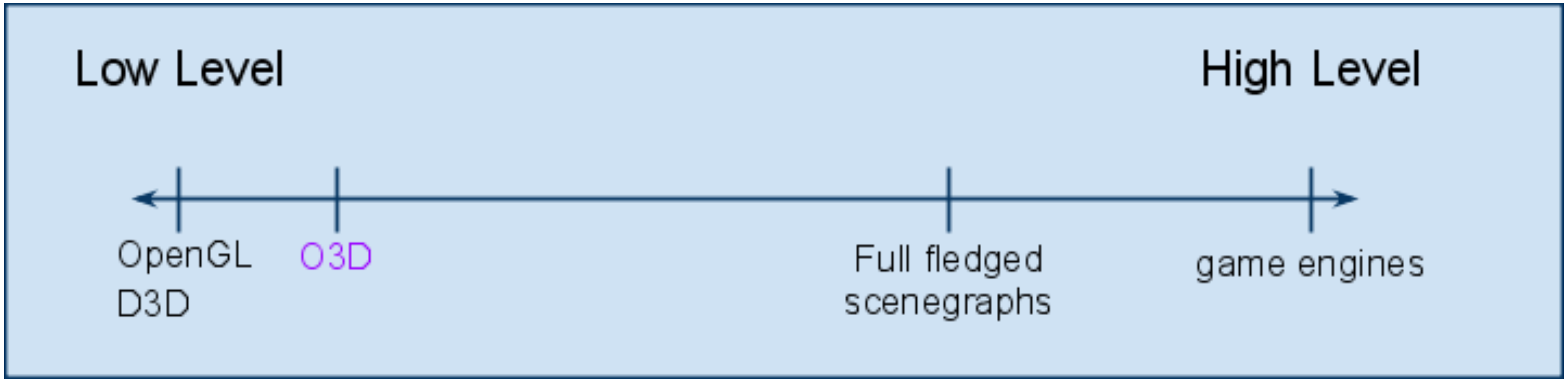
```
circle.setPosition(i, 5)
i++
```

Google Developer Day2009

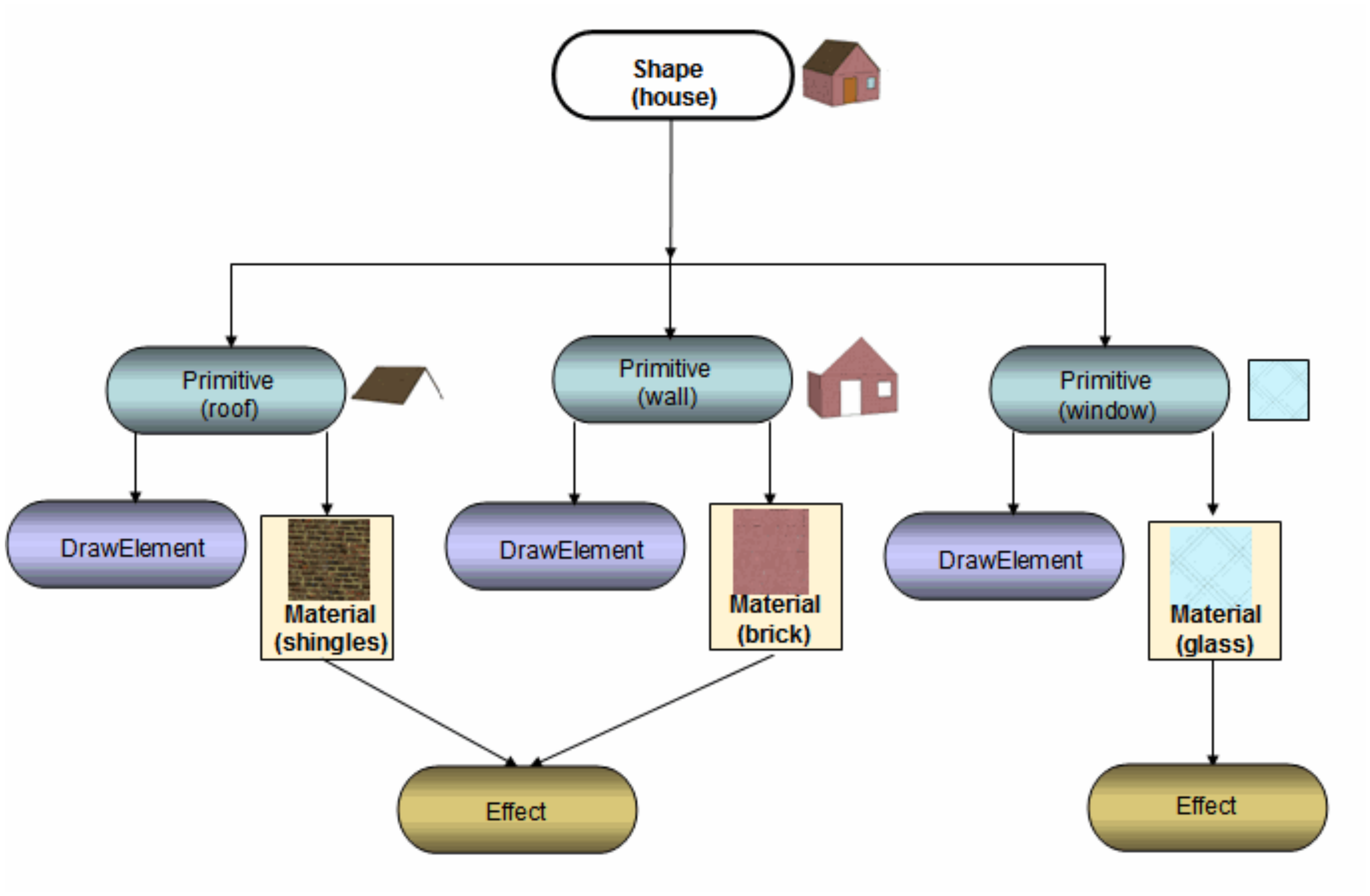
The Graphics APIs Spectrum

O3D is a **low-level, retained mode** API

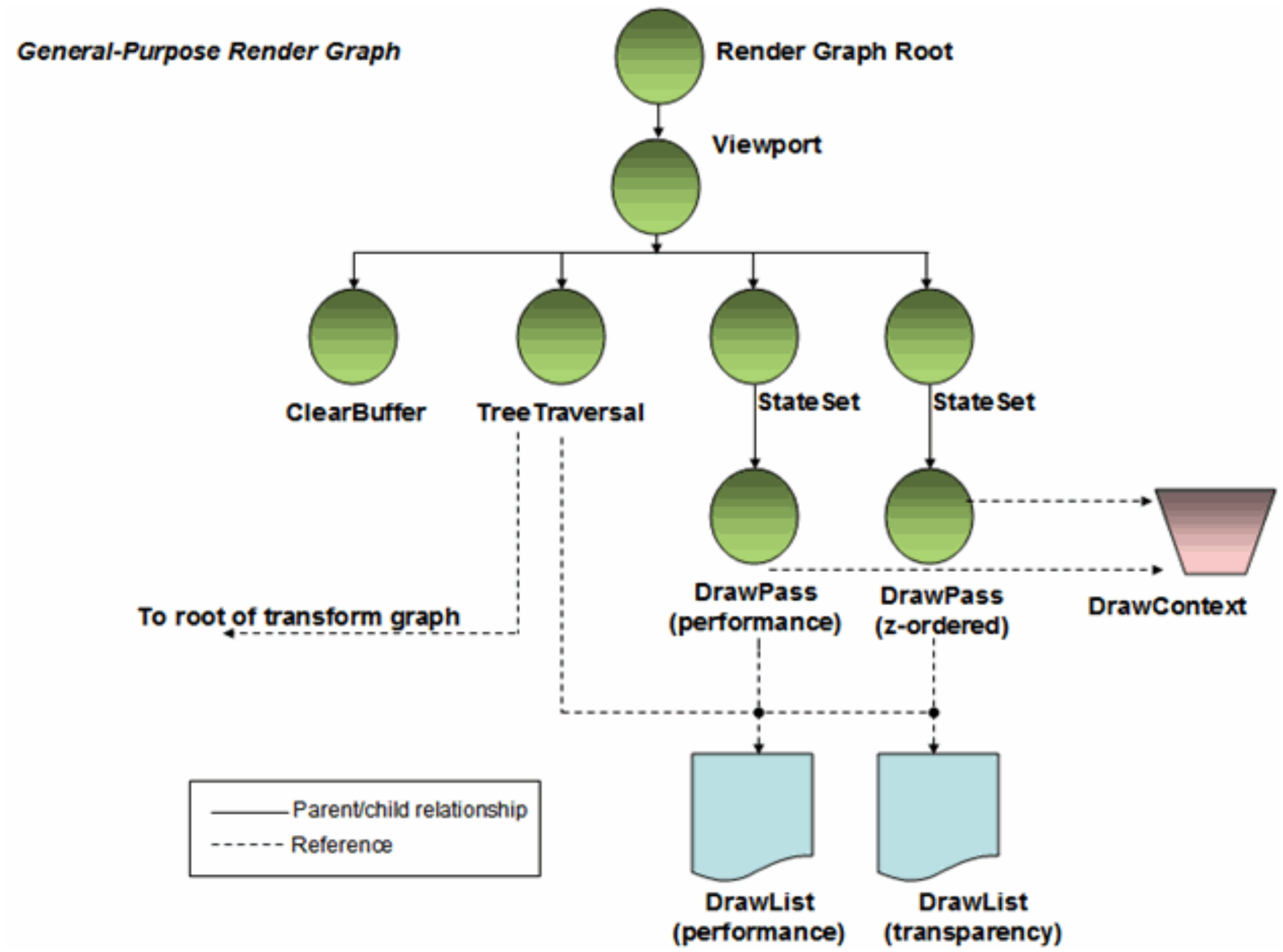
- Has most of the flexibility of an immediate mode API
- Performance critical functionality (e.g. z-sorting, culling, animation) executed in native code.



Transform Graph Detail



A Rendergraph Example



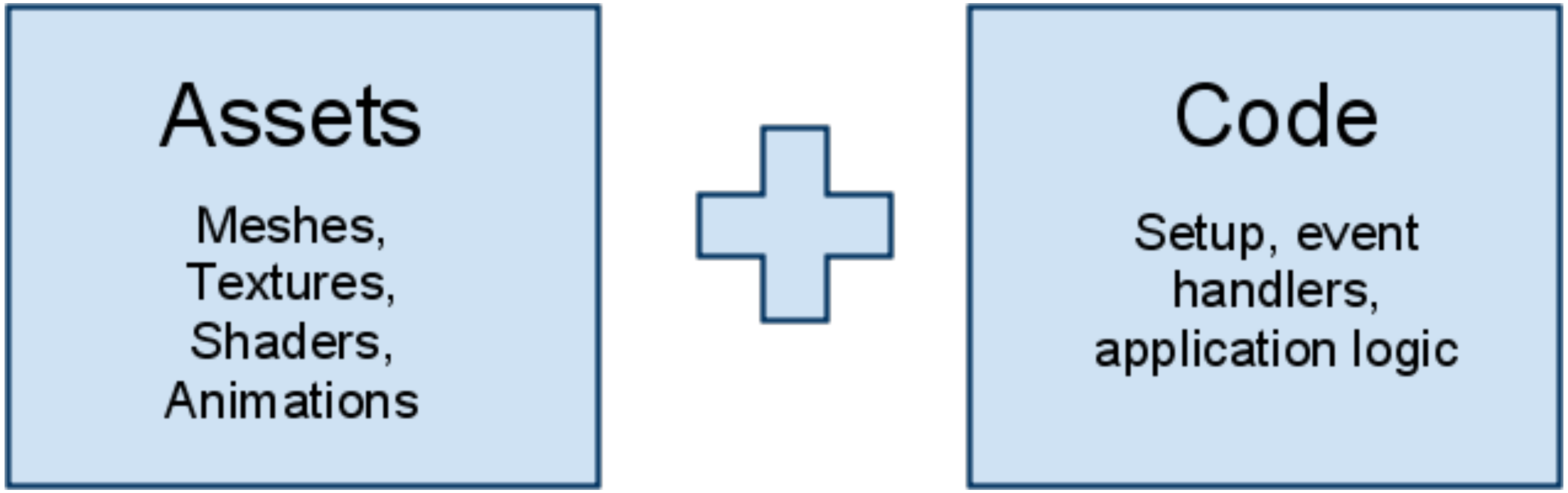
Google Developer Day2009

Programming with O3D

- 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
  - They serve as valuable sample code too!
- Over time higher-level libraries will be developed

Google Developer Day2009

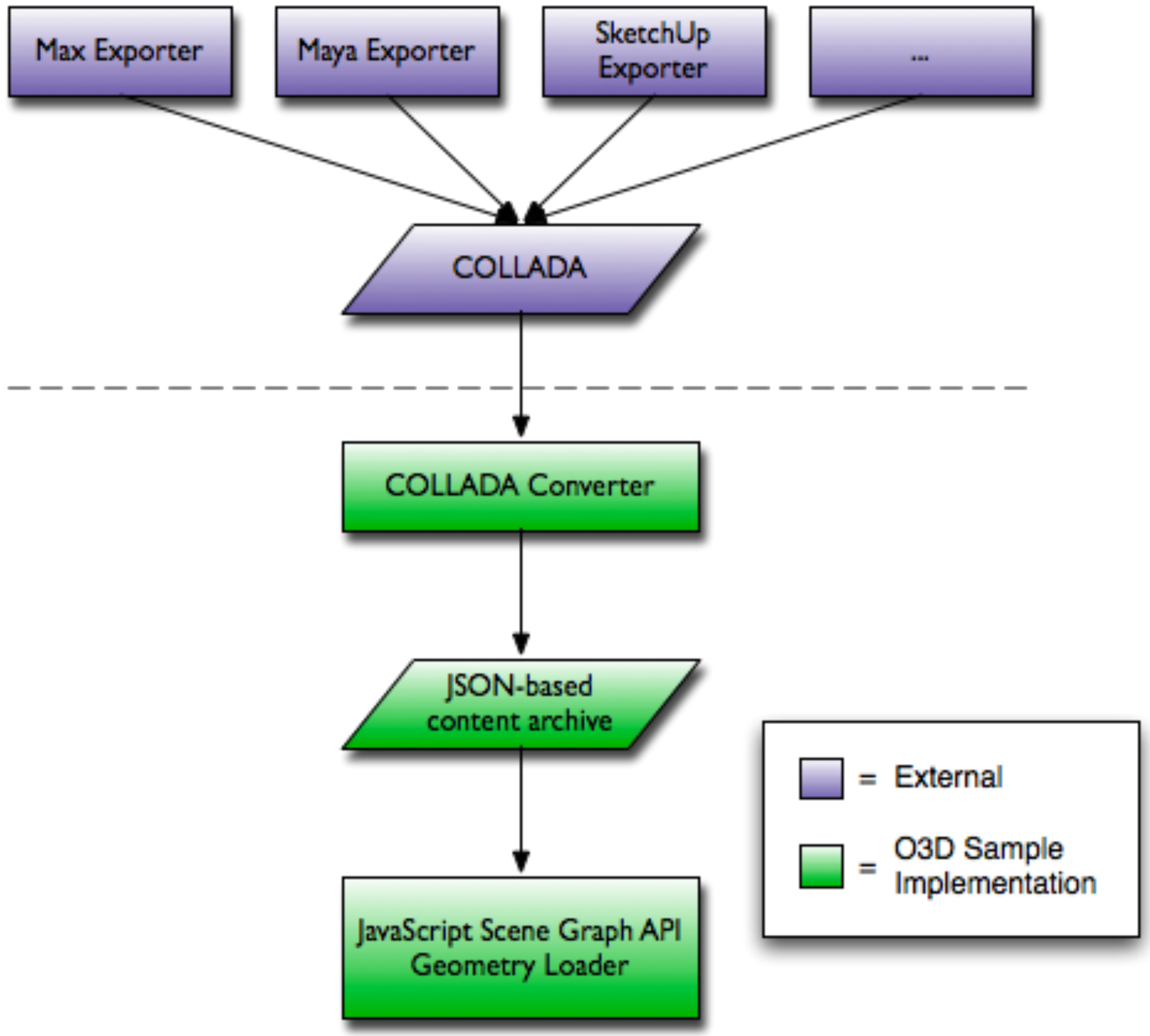
O3D Application


Assets
Meshes, Textures, Shaders, Animations
Code
Setup, event handlers, application logic

Loading assets in O3D

- O3D does NOT have its own file format
- O3D can load zipped archive files (.tar.gz) containing:
  - Text (JSON)
  - Image files (for textures)
  - Binary Buffers (for mesh data, animation, etc)
- The 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

Importing 3D models from DCC tools


Max Exporter
Maya Exporter
SketchUp Exporter
...
COLLADA
COLLADA Converter
JSON-based content archive
JavaScript Scene Graph API Geometry Loader
= External
= O3D Sample Implementation

Making an Application with O3D

Google Developer Day 2009

Step 1: Setup

Make the HTML for the webpage.

```
...
<script type="text/javascript" src="../o3djs/base.js"></script>
<script type="text/javascript">
o3djs.require('o3djs.util');
o3djs.require('o3djs.math');

...

<body onload="init();">
<div id="o3d" style="width: 800px; height: 600px;"></div>

...
```

Google Developer Day2009

Step 1: Setup cont. Initialize O3D

Initialize and make a render graph

```
//Creates the client area.
function init() {
  o3djs.util.makeClients(initStep2);
}

function initStep2(clientElements) {
  g_o3dElement = clientElements[0];
  g_math = o3djs.math;
  g_client = g_o3dElement.client;
  g_pack = g_client.createPack();
  g_root = g_pack.createObject('Transform');
  g_viewInfo = o3djs.rendergraph.createBasicView(
      g_pack,
      g_root,
      g_client.renderGraphRoot);
  ...
```

Google Developer Day2009

Step 1: Setup Cont. Make a Camera

Set the view and projection matrices

```
var g_eye = [15, 25, 50];
var g_target = [0, 10, 0];
var g_up = [0, 1, 0];

function updateProjection() {
  g_viewInfo.drawContext.projection =
      g_math.matrix4.perspective(
          g_math.degToRad(45),        // field of view.
          g_o3dWidth / g_o3dHeight,   // aspect ratio
          0.1,                        // Near plane.
          5000);                      // Far plane.
}

function updateCamera() {
  g_viewMatrix = g_math.matrix4.lookAt(g_eye, g_target, g_up);
  g_viewInfo.drawContext.view = g_viewMatrix;
}
```

Step 2: Put something on the screen.

Make some materials.

```
var redMaterial = o3djs.material.createBasicMaterial(
    g_pack,
    g_viewInfo,
    [0.2, 1, 0.2, 1]);  // green

var checkerMaterial = o3djs.material.createMaterialFromFile(
    g_pack,
    'shaders/checker.shader',
    g_viewInfo.performanceDrawList);

g_globalParams =
    o3djs.material.createAndBindStandardParams(g_pack);

g_globalParams.lightWorldPos.value = [30, 60, 40];
g_globalParams.lightColor.value = [1, 1, 1, 1];
```

Step 2: Put something on screen.

Start simple. A plane and a cylinder.

```
// Create a ground plane.
var shape = o3djs.primitives.createPlane(
    g_pack, checkerMaterial, 100, 100, 10, 10);
var transform = g_pack.createObject('Transform');
transform.parent = g_root;
transform.addShape(shape);

// Create a cylinder.
var shape = o3djs.primitives.createCylinder(
    g_pack, redMaterial, 2.5, 5, 20, 1,
    g_math.matrix4.translation([0, 2.5, 0]));
g_playerTransform = g_pack.createObject('Transform');
g_playerTransform.parent = g_root;
g_playerTransform.addShape(shape);
```

Step 3: Let the user move something.

Track the state of the keyboard.

```
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;
}
```

Step 3: Per frame processing.

Setup an on render callback.

```
initStep2(...) {
  ...
  g_client.setRenderCallback(onRender);
}

function onRender(renderEvent) {
  var elapsedTime = renderEvent.elapsedTime;
  handleMoveKeys(elapsedTime);
};
```

Step 3: Move something.

Let the user move the player with the keyboard

```
function handleMoveKeys(elapsedTime) {
  var directionX = 0;
  var directionZ = 0;

  if (g_keyDown[37] || g_keyDown[65]) { directionX = -1; }
  if (g_keyDown[39] || g_keyDown[68]) { directionX = 1; }
  if (g_keyDown[38] || g_keyDown[87]) { directionZ = -1; }
  if (g_keyDown[40] || g_keyDown[83]) { directionZ = 1; }

  g_playerXPosition += directionX;
  g_playerZPosition += directionZ;

  g_playerTransform.identity();
  g_playerTransform.translate(
      g_playerXPosition, 0, g_playerZPosition);
}
```

Google Developer Day2009

Step 4: Make it frame rate independent

```
var MOVE_VELOCITY = 25;  // in units per second.

function handleMoveKeys(elapsedTime) {
  ...

  g_playerXPosition += MOVE_VELOCITY * directionX *
                       elapsedTime;
  g_playerZPosition += MOVE_VELOCITY * directionZ *
                       elapsedTime;

  g_playerTransform.identity();
  g_playerTransform.translate(
      g_playerXPosition, 0, g_playerZPosition);
}
```

Step 5: Make it move relative to the camera.

Compute a move matrix.

```
function computeMoveMatrixFromViewMatrix(viewMatrix) {
  var cameraMatrix = g_math.matrix4.inverse(viewMatrix);
  var xAxis = g_math.cross([0, 1, 0],
                           cameraMatrix[2].slice(0, 3));
  var zAxis = g_math.cross(xAxis, [0, 1, 0]);
  return [
      xAxis.concat(0),
      [0, 1, 0, 0],
      zAxis.concat(0),
      [0, 0, 0, 1]];
}

function updateCamera() {
  ...
  g_moveMatrix = computeMoveMatrixFromViewMatrix(g_viewMatrix);
};
```

Step 5: Make it move relative to the camera.

Use the move matrix to move the player

```
function handleMoveKeys(elapsedTime) {
  ...
  var moveTranslation = g_math.matrix4.transformPoint(
      g_moveMatrix,
      [MOVE_VELOCITY * directionX * elapsedTime,
       0,
       MOVE_VELOCITY * directionZ * elapsedTime]);

  g_playerXPosition += moveTranslation[0];
  g_playerZPosition += moveTranslation[2];

  g_playerTransform.identity();
  g_playerTransform.translate(
      g_playerXPosition, 0, g_playerZPosition);
}
```

Step 6: Make the camera follow the player.

Easy. Just set the target.

```
function moveCamera() {
  g_target = [g_playerXPosition, 10, g_playerZPosition];
  updateCamera();
}

function onRender(renderEvent) {
  moveCamera();
};
```

Google Developer Day2009

Step 7: Smooth the camera movement.

Make it play catch up.

```
function moveCamera() {
  var newTarget = [g_playerXPosition, 10, g_playerZPosition];
  g_target = g_math.lerpVector(g_target, newTarget, 0.03);
  updateCamera();
}
```

Step 8: Add some action.

Make him jump

```
function handleMoveKeys(elapsedTime) {
  ...
  if (g_canJump) {
    if (g_keyDown[32]) {
      g_jumping = true;
      g_canJump = false;
      g_playerYVelocity = JUMP_VELOCITY;
    }
  } else {
    if (g_jumping) {
      g_playerYVelocity += GRAVITY * elapsedTime;
      g_playerYPosition += g_playerYVelocity * elapsedTime;
      if (g_playerYPosition <= 0) {
        g_playerYPosition = 0;
        g_jumping = false;
      }
    } else {
      if (!g_keyDown[32]) {
        g_canJump = true;
      }
    }
  }
  ...
  g_playerTransform.translate(
      g_playerXPosition, g_playerYPosition, g_playerZPosition);
}
```

Google Developer Day2009

Step 9: Add effects.

Create a particle emitter

```
function initStep2(clientElements) {
  ...
  g_particleSystem = o3djs.particles.createParticleSystem(g_pack, g_viewInfo);
  g_poofEmitter = g_particleSystem.createParticleEmitter();
  g_poofEmitter.setState(o3djs.particles.ParticleStateIds.ADD);
  g_poofEmitter.setColorRamp(
      [1, 1, 1, 0.3,
       1, 1, 1, 0]);
  g_poofEmitter.setParameters({
    numParticles: 30,
    lifeTime: 0.5,
    startTime: 0,
    startSize: 5,
    endSize: 10,
    spinSpeedRange: 10},
    function(index, parameters) {
      var angle = Math.random() * 2 * Math.PI;
      parameters.velocity = g_math.matrix4.transformPoint(
          g_math.matrix4.rotationY(angle), [25, 2.5, 0]);
      parameters.acceleration = g_math.mulVectorVector(
          parameters.velocity, [-1.5, 1, -1.5]);
    });
  g_poof = g_poofEmitter.createOneShot(g_root);
```

Google Developer Day2009

Step 9: Add Effects

Make a poof of dust when he lands.

```
function handleMoveKeys(elapsedTime) {
  ...
    if (g_jumping) {
      g_playerYVelocity += GRAVITY * elapsedTime;
      g_playerYPosition += g_playerYVelocity * elapsedTime;
      if (g_playerYPosition <= 0) {
        g_playerYPosition = 0;
        g_poof.trigger(
            [g_playerXPosition,
             g_playerYPosition,
             g_playerZPosition]);
        g_jumping = false;
      }
```

Google Developer Day2009

Step 10: Load a character.

```
function initStep2(...) {
  ...
  var transform = g_pack.createObject('Transform');
  g_playerTransform = transform;
  var playerPack = g_client.createPack();
  o3djs.scene.loadScene(g_client, playerPack,
                        g_playerTransform,
                        'assets/character.o3dtgz', initStep3);
}

function initStep3(playerPack, playerParent, exception) {
  o3djs.pack.preparePack(playerPack, g_viewInfo);
  o3djs.material.bindParams(playerPack, g_globalParams);
  g_playerTransform.parent = g_root;
  ...
}
```

Step 11: Orient the character.

Making him point in the direction of movement.

```
var g_playerDirection = 0;
...
function handleMoveKeys(elapsedTime) {
  ...
  if (directionX != 0 || directionZ != 0) {
    var moveTranslation = ...
    var targetDirection = Math.atan2(moveTranslation[0],
                                     moveTranslation[2]);
    g_playerDirection = g_math.lerpRadian(
      g_playerDirection, targetDirection, 0.2);
    ...
  }

  g_playerTransform.identity();
  g_playerTransform.translate(
      g_playerXPosition, g_playerYPosition, g_playerZPosition);
  g_playerTransform.rotateY(g_playerDirection);
}
```

Step 12: Animate the character.

Setup an animation parameter.

```
var IDLE_START_TIME = 247 / 30;
var IDLE_END_TIME = 573 / 30;
var IDLE_TIME_RANGE = IDLE_END_TIME - IDLE_START_TIME;
var g_animTimer = IDLE_START_TIME;

function initStep2(...) {
  ...
  var paramObject = playerPack.createObject('ParamObject');
  g_animParam = paramObject.createParam('animTime',
                                        'ParamFloat');
  o3djs.scene.loadScene(g_client, playerPack,
                        g_playerTransform,
                        'assets/character.o3dtgz', initStep3,
                        {opt_animSource: g_animParam});
  ...
```

Google Developer Day2009

Step 12: Animate the character.

Update the animation parameter.

```
function onRender(renderEvent) {
  ...
  handleAnimation(elapsedTime);
};

function handleAnimation(elapsedTime) {
  g_animTimer += elapsedTime;
  if (g_animTimer >= IDLE_END_TIME) {
    g_animTimer = g_math.modClamp(g_animTimer,
                                  IDLE_TIME_RANGE,
                                  IDLE_START_TIME);
  }
  g_animParam.value = g_animTimer;
}
```

Google Developer Day2009

Step 13: Add more animation.

Walk, Jump, More Idles...

```
Repeat step 12 for various actions.

See step13.html

diff step12.html and step13.html to see what changed.
```

Google Developer Day2009

Step 14: Load a background.

```
function initStep2(...) {
  ...
  var loader = o3djs.loader.createLoader(initStep3);
  loader.loadScene(g_client, playerPack, g_playerTransform,
                   'assets/character.o3dtgz', prepareScene,
                   {opt_animSource: g_animParam});
  var worldPack = g_client.createPack();
  g_worldTransform = worldPack.createObject('Transform');
  loader.loadScene(g_client, worldPack, g_worldTransform,
                   'assets/background.o3dtgz', prepareScene)
  loader.finish();
}
function prepareScene(pack, sceneRoot, exception) {
  o3djs.pack.preparePack(pack, g_viewInfo);
  o3djs.material.bindParams(pack, g_globalParams);
  sceneRoot.parent = g_root;
}
function initStep3(...) {
```

Questions?

http://code.google.com/apis/o3d

Google
Developer
Day2009

Google
Developer
Day2009