THE UNIVERSITY of NEW MEXICO

Introduction to Computer Graphics with WebGL

Ed Angel

Animation

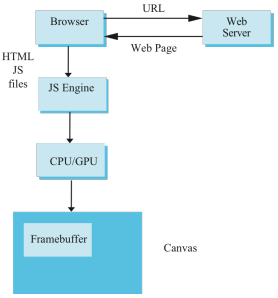
THE UNIVERSITY of NEW MEXICO

Callbacks

- Programming interface for event-driven input uses *callback functions* or *event listeners*
  - Define a callback for each event the graphics system recognizes
  - Browsers enters an event loop and responds to those events for which it has callbacks registered
  - The callback function is executed when the event occurs

THE UNIVERSITY of NEW MEXICO

Execution in a Browser



Execution in a Browser

- Start with HTML file
  - Describes the page
  - May contain the shaders
  - Loads files
- Files are loaded asynchronously and JS code is executed
- Then what?
- Browser is in an event loop and waits for an event

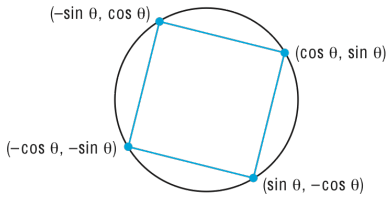
onload Event

- What happens with our JS file containing the graphics part of our application?
  - All the "action" is within functions such as init() and render()
  - Consequently these functions are never executed and we see nothing
- Solution: use the onload window event to initiate execution of the init function
  - onload event occurs when all files read

```
-window.onload = init;
```


Rotating Square

Consider the four points



Animate display by rerendering with different values of θ

Simple but Slow Method

```
for(var theta = 0.0; theta <thetaMax; theta += dtheta; {

vertices[0] = vec2(Math.sin(theta), Math.cos.(theta));
vertices[1] = vec2(Math.sin(theta), -Math.cos.(theta));
vertices[2] = vec2(-Math.sin(theta), -Math.cos.(theta));
vertices[3] = vec2(-Math.sin(theta), Math.cos.(theta));

gl.bufferSubData(…….

render();
}
```


Better Way

- Send original vertices to vertex shader
- Send θ to shader as a uniform variable
- Compute vertices in vertex shader
- Render recursively

Render Function

```
var thetaLoc = gl.getUniformLocation(program, "theta");

function render()
{
     gl.clear(gl.COLOR_BUFFER_BIT);
     theta += 0.1;
     gl.uniform1f(thetaLoc, theta);
     gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
     render();
}
```

Vertex Shader

```
attribute vec4 vPosition;
uniform float theta;

void main()
{
   gl_Position.x = -sin(theta) * vPosition.x
        + cos(theta) * vPosition.y;
   gl_Position.y = sin(theta) * vPosition.y
        + cos(theta) * vPosition.x;
   gl_Position.z = 0.0;
   gl_Position.w = 1.0;
}
```


Double Buffering

- Although we are rendering the square, it always into a buffer that is not displayed
- Browser uses double buffering
  - Always display front buffer
  - Rendering into back buffer
  - Need a buffer swap
- Prevents display of a partial rendering

Triggering a Buffer Swap

- Browsers refresh the display at ~60 Hz
  - redisplay of front buffer
  - not a buffer swap
- Trigger a buffer swap though an event
- Two options for rotating square
  - Interval timer
  - requestAnimFrame

THE UNIVERSITY of NEW MEXICO

Interval Timer

- Executes a function after a specified number of milliseconds
  - Also generates a buffer swap

```
setInterval(render, interval);
```

- Note an interval of 0 generates buffer swaps as fast as possible

THE UNIVERSITY of NEW MEXICO

requestAnimFrame

```
function render {
    gl.clear(gl.COLOR_BUFFER_BIT);
    theta += 0.1;
    gl.uniform1f(thetaLoc, theta);
    gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
    requestAnimFrame(render);
}
```


THE UNIVERSITY of NEW MEXICO

Add an Interval

```
function render()
{
    setTimeout( function() {
      requestAnimFrame(render);
      gl.clear(gl.COLOR_BUFFER_BIT);
      theta += 0.1;
      gl.uniform1f(thetaLoc, theta);
      gl.drawArrays(gl.TRIANGLE_STRIP, 0, 4);
   }, 100);
}
```