Controling Rotation Speed

```
var delay = 100;

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

Menus

- Use the HTML select element
- Each entry in the menu is an option element with an integer value returned by click event

```
<select id="mymenu" size="3">
<option value="0">Toggle Rotation Direction</option>
<option value="1">Spin Faster</option>
<option value="2">Spin Slower</option>
</select>
```

Menu Listener

```
var m = document.getElementById("mymenu");
m.addEventListener("click", function() {
   switch (m.selectedIndex) {
      case 0:
         direction = !direction;
         break;
      case 1:
         delay /= 2.0;
         break;
      case 2:
         delay *= 2.0;
         break;
       }
});
```

Using keydown Event

```
window.addEventListener("keydown", function() {
  switch (event.keyCode) {
    case 49: // ’1’ key
      direction = !direction;
      break;
    case 50: // ’2’ key
      delay /= 2.0;
      break;
    case 51: // ’3’ key
      delay *= 2.0;
      break;
  }
});
```

Don’t Know Unicode

```
window.onkeydown = function(event) {
  var key = String.fromCharCode(event.keyCode);
  switch (key) {
    case ’1’:
      direction = !direction;
      break;
    case ’2’:
      delay /= 2.0;
      break;
    case ’3’:
      delay *= 2.0;
      break;
  }
};
```

Slider Element

- Puts slider on page
  - Give it an identifier
  - Give it minimum and maximum values
  - Give it a step size needed to generate an event
  - Give it an initial value
- Use div tag to put below canvas

```
<div>
speed 0 <input id="slide" type="range"
  min="0" max="100" step="10" value="50" />
100 </div>
```

onchange Event Listener

```
document.getElementById("slide").onchange =
  function() { delay = event.srcElement.value; };
```

Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

Position Input

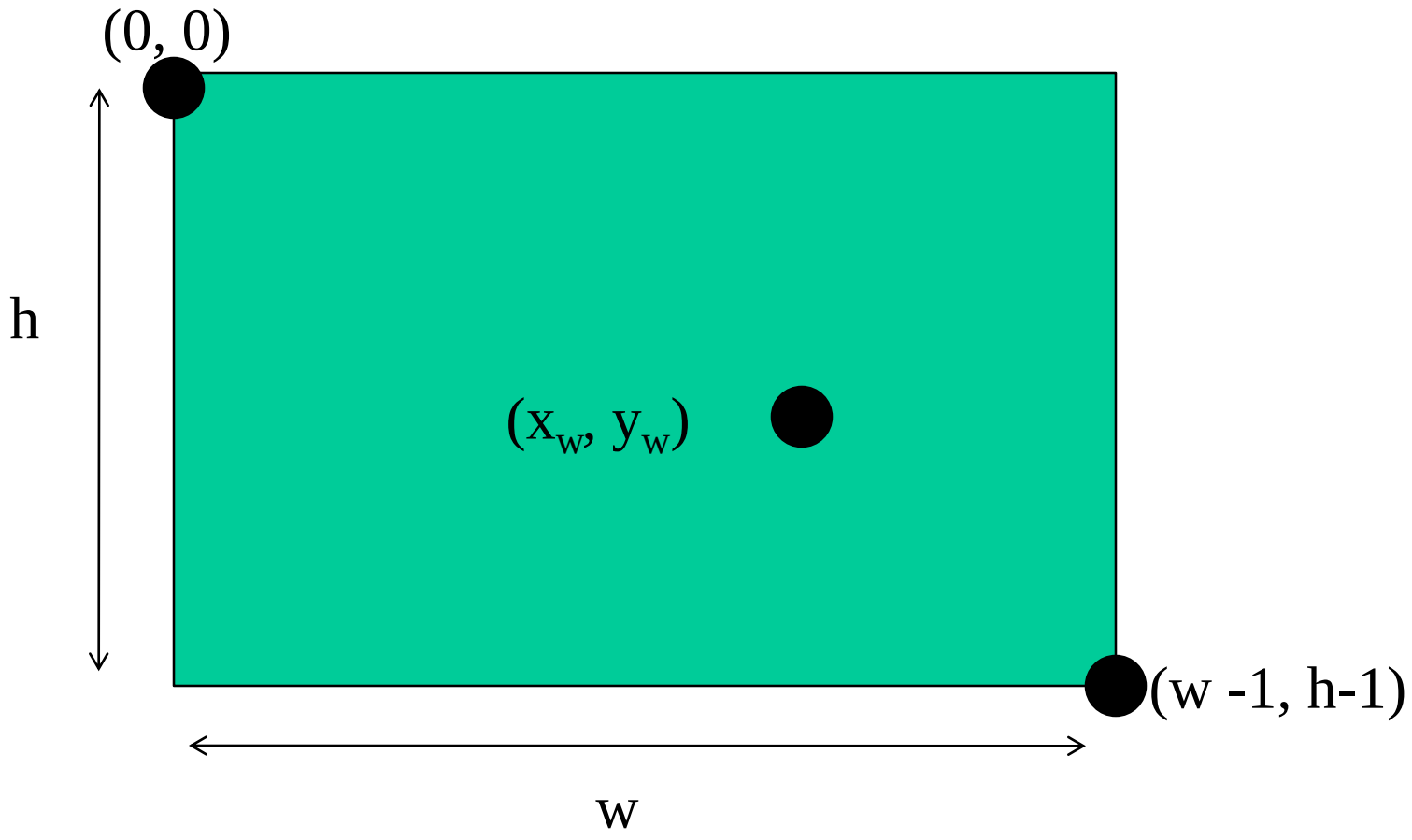
Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

Objectives

- Learn to use the mouse to give locations
  - Must convert from position on canvas to position in application
- Respond to window events such as reshapes triggered by the mouse

The University of New Mexico

Window Coordinates



Window to Clip Coordinates

$$(0,h) \rightarrow (-1,-1)$$

$$(w,0) \rightarrow (1,1)$$

$$x = -1 + \frac{2 * x_w}{w}$$

$$y = -1 + \frac{2 * (h - y_w)}{h}$$

Returning Position from Click Event

Canvas specified in HTML file of size
canvas.width x canvas.height

Returned window coordinates are event.clientX and event.clientY

```
// add a vertex to GPU for each click
canvas.addEventListener("click", function() {
   gl.bindBuffer(gl.ARRAY_BUFFER, vBuffer);
   var t = vec2(-1 + 2*event.clientX/canvas.width,
     -1 + 2*(canvas.height-event.clientY)/canvas.height);
   gl.bufferSubData(gl.ARRAY_BUFFER,
     sizeof['vec2']*index, t);
   index++;
});
```

CAD-like Examples

www.cs.unm.edu/~angel/WebGL/7E/03

square.html: puts a colored square at location of each mouse click

triangle.html: first three mouse clicks define first triangle of triangle strip. Each succeeding mouse clicks adds a new triangle at end of strip

cad1.html: draw a rectangle for each two successive mouse clicks

cad2.html: draws arbitrary polygons

Window Events

- Events can be generated by actions that affect the canvas window
  - moving or exposing a window
  - resizing a window
  - opening a window
  - iconifying/deiconifying a window a window
- Note that events generated by other application that use the canvas can affect the WebGL canvas
  - There are default callbacks for some of these events

Reshape Events

- Suppose we use the mouse to change the size of our canvas
- Must redraw the contents
- Options
  - Display the same objects but change size
  - Display more or fewer objects at the same size
- Almost always want to keep proportions

onresize Event

- Returns size of new canvas is available through window.innerHeight and window.innerWidth
- Use innerHeight and innerWidth to change canvas.height and canvas.width
- Example (next slide): maintaining a square display

Keeping Square Proportions

```
window.onresize = function() {
  var min = innerWidth;
  if (innerHeight < min) {
    min = innerHeight;
  }
  if (min < canvas.width || min < canvas.height) {
    gl.viewport(0, canvas.height-min, min, min);
  }
};
```

Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

Picking

Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

Objectives

- How do we identify objects on the display
- Overview three methods
  - selection
  - using an off-screen buffer and color
  - bounding boxes

Why is Picking Difficult?

- Given a point in the canvas how do map this point back to an object?
- Lack of uniqueness
- Forward nature of pipeline
- Take into account difficulty of getting an exact position with a pointing device

Selection

- Supported by fixed function OpenGL pipeline
- Each primitive is given an id by the application indicating to which object it belongs
- As the scene is rendered, the id's of primitives that render near the mouse are put in a hit list
- Examine the hit list after the rendering

Selection

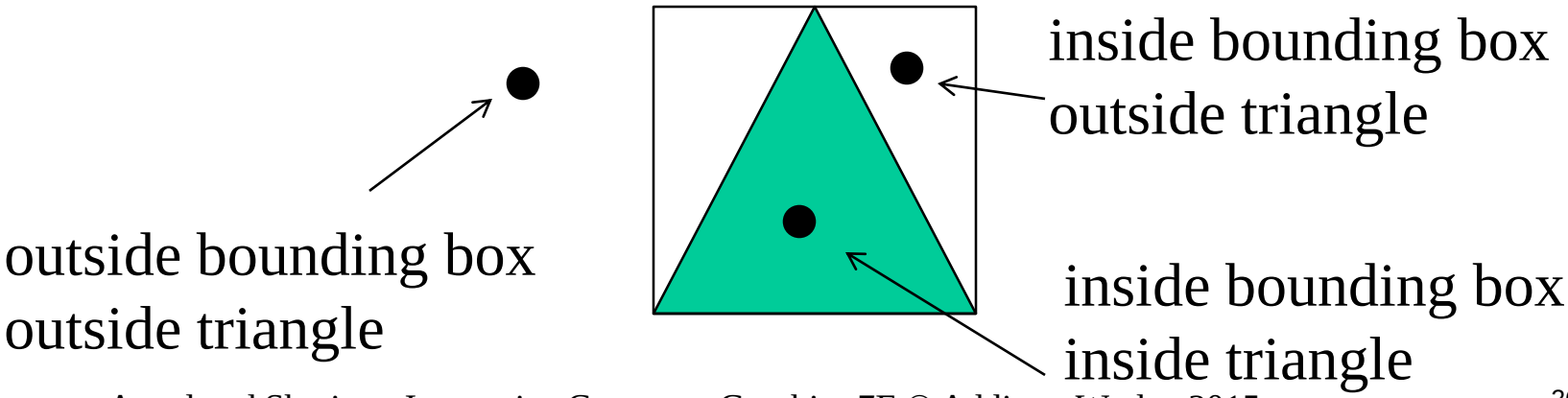
- Implement by creating a window that corresponds to small area around mouse
  - We can track whether or not a primitive renders to this window
  - Do not want to display this rendering
  - Render off-screen to an extra color buffer or user back buffer and don't do a swap
- Requires a rendering which puts depths into hit record
- Possible to implement with WebGL

Picking with Color

- We can use gl.readPixels to get the color at any location in window
- Idea is to use color to identify object but
  - Multiple objects can have the same color
  - A shaded object will display many colors
- Solution: assign a unique color to each object and render off-screen
  - Use gl.readPixels to get color at mouse location
  - Use a table to map this color to an object

Picking with Bounding Boxes

- Both previous methods require an extra rendering each time we do a pick
- Alternative is to use a table of (axis-aligned) bounding boxes
- Map mouse location to object through table



Introduction to Computer Graphics with WebGL

Ed Angel

Professor Emeritus of Computer Science

Founding Director, Arts, Research, Technology and Science Laboratory

University of New Mexico

Geometry

Ed Angel
Professor Emeritus of Computer Science
University of New Mexico

Objectives

- Introduce the elements of geometry
  - Scalars
  - Vectors
  - Points
- Develop mathematical operations among them in a coordinate-free manner
- Define basic primitives
  - Line segments
  - Polygons

Basic Elements

- Geometry is the study of the relationships among objects in an n-dimensional space
  - In computer graphics, we are interested in objects that exist in three dimensions
- Want a minimum set of primitives from which we can build more sophisticated objects
- We will need three basic elements
  - Scalars
  - Vectors
  - Points

Coordinate-Free Geometry

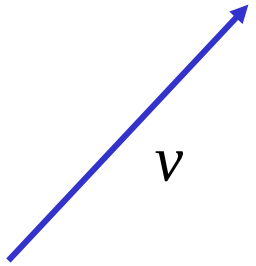
- When we learned simple geometry, most of us started with a Cartesian approach
  - Points were at locations in space **p**=(x,y,z)
  - We derived results by algebraic manipulations involving these coordinates
- This approach was nonphysical
  - Physically, points exist regardless of the location of an arbitrary coordinate system
  - Most geometric results are independent of the coordinate system
  - Example Euclidean geometry: two triangles are identical if two corresponding sides and the angle between them are identical

Scalars

- Need three basic elements in geometry
  - Scalars, Vectors, Points
- Scalars can be defined as members of sets which can be combined by two operations (addition and multiplication) obeying some fundamental axioms (associativity, commutivity, inverses)
- Examples include the real and complex number systems under the ordinary rules with which we are familiar
- Scalars alone have no geometric properties

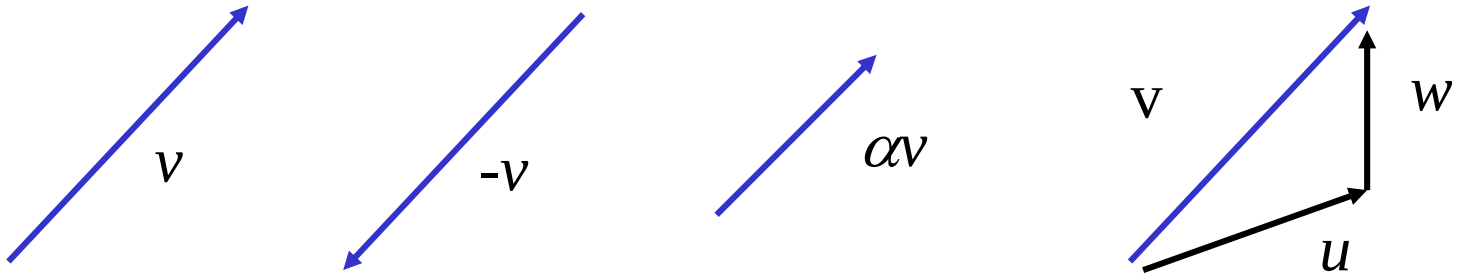
Vectors

- Physical definition: a vector is a quantity with two attributes
  - Direction
  - Magnitude
- Examples include
  - Force
  - Velocity
  - Directed line segments
    - Most important example for graphics
    - Can map to other types



Vector Operations

- Every vector has an inverse
  - Same magnitude but points in opposite direction
- Every vector can be multiplied by a scalar
- There is a zero vector
  - Zero magnitude, undefined orientation
- The sum of any two vectors is a vector
  - Use head-to-tail axiom



Linear Vector Spaces

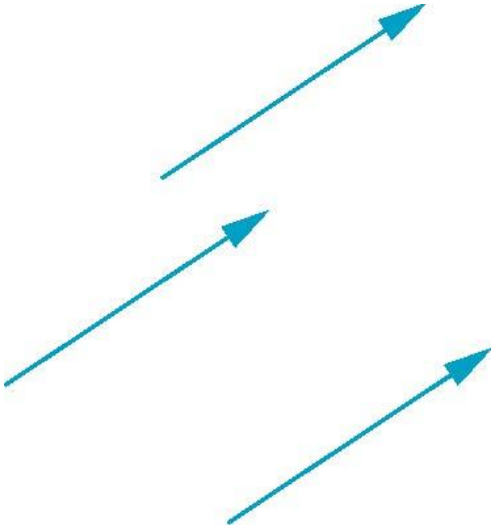
- Mathematical system for manipulating vectors
- Operations
  - Scalar-vector multiplication $u=\alpha v$
  - Vector-vector addition: $w=u+v$
- Expressions such as

  $v=u+2w-3r$

Make sense in a vector space

Vectors Lack Position

- These vectors are identical
  - Same length and magnitude

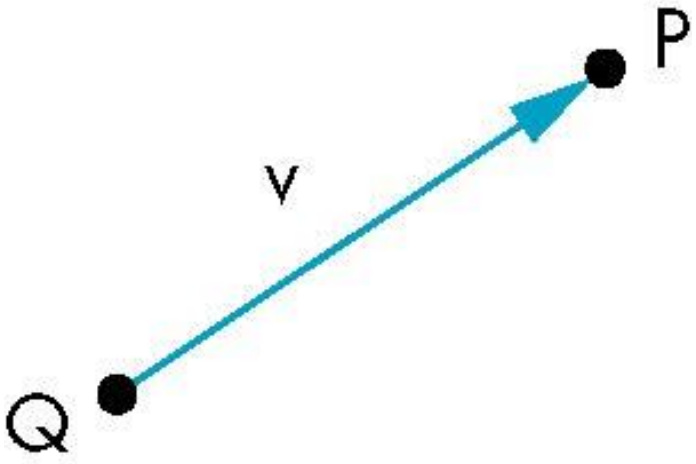


- Vectors spaces insufficient for geometry
  - Need points

The University of New Mexico

Points

- Location in space
- Operations allowed between points and vectors
  - Point-point subtraction yields a vector
  - Equivalent to point-vector addition



$v = P - Q$

$P = v + Q$

Affine Spaces

- Point + a vector space
- Operations
  - Vector-vector addition
  - Scalar-vector multiplication
  - Point-vector addition
  - Scalar-scalar operations
- For any point define
  - $1 \bullet P = P$
  - $0 \bullet P = \mathbf{0}$ (zero vector)

The University of New Mexico

Lines

- Consider all points of the form
  - $P(\alpha)=P_0 + \alpha \mathbf{d}$
  - Set of all points that pass through P_0 in the direction of the vector $\mathbf{d}$