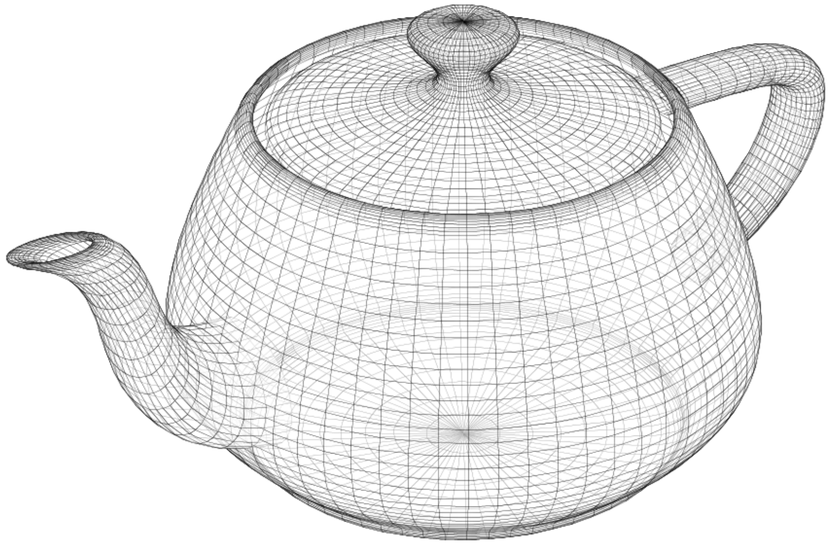
WebGL Drawing Primitives



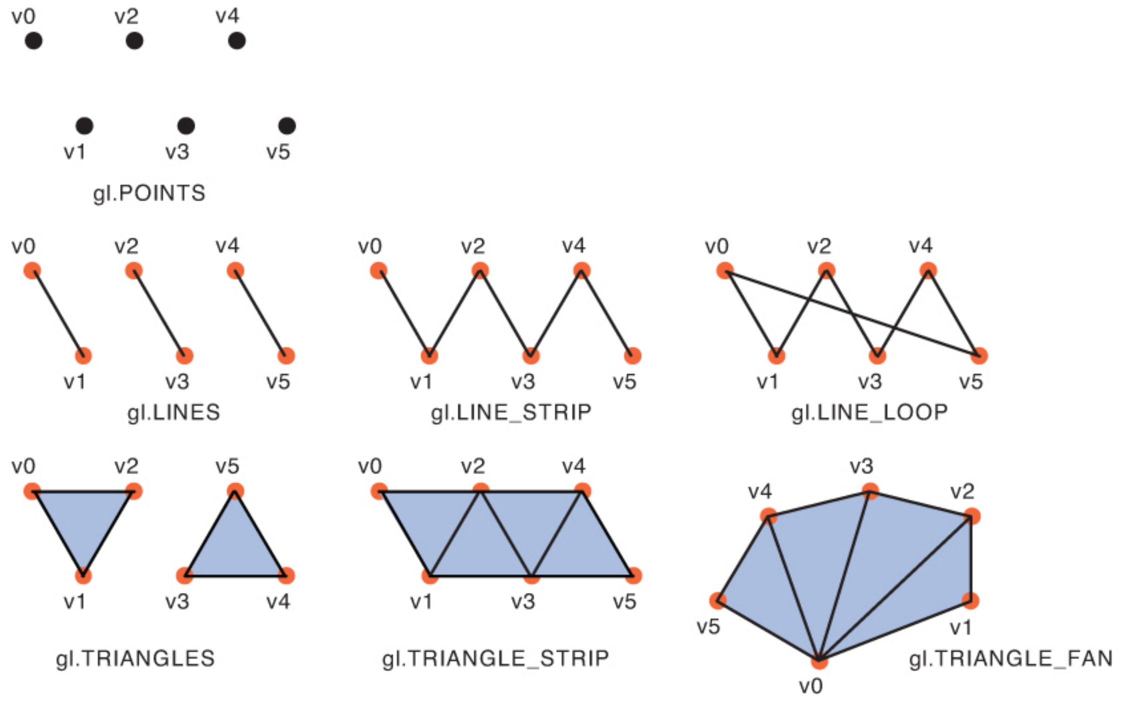
CS 418: Interactive Computer Graphics

Professor Eric Shaffer

What is a Primitive?

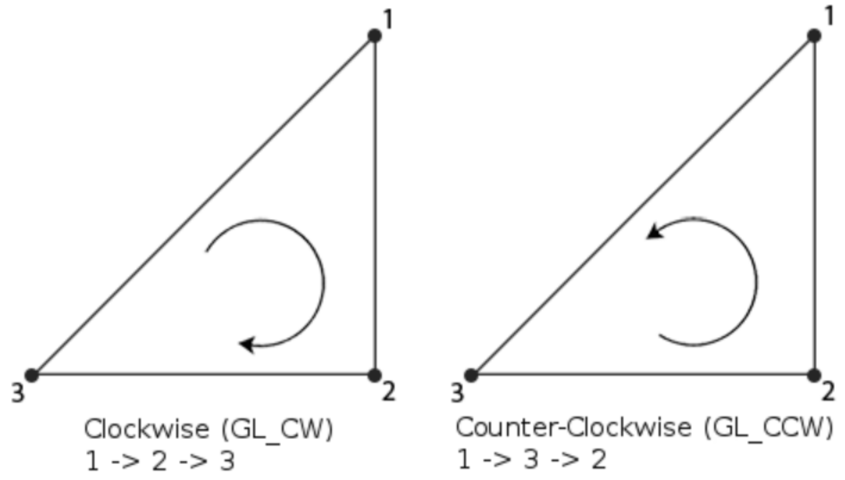
In computer graphics primitives are basic geometric shapes a system can render

WebGL primitives include



If you want to draw anything else (e.g. curves or spheres) you need to write code to render those shapes using the primitives shown here.

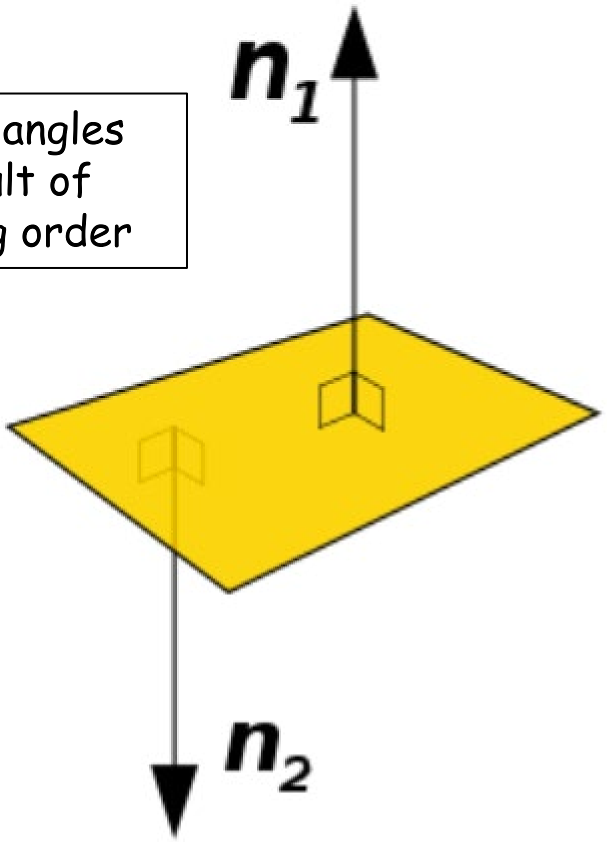
Winding Order



Clockwise (GL_CW)
1 -> 2 -> 3

Counter-Clockwise (GL_CCW)
1 -> 3 -> 2

Pro-tip: If you have black or missing triangles in a rendered scene, that may be a result of the vertices being in the wrong winding order

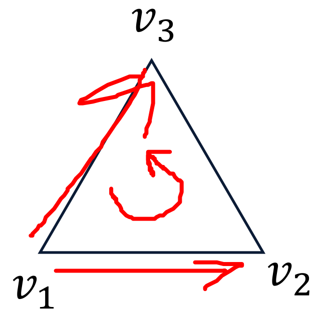


The order in which a triangle's vertices are specified can matter

It's common to specify the order as CCW when looking the outward face of the triangle
This is the default in WenGL

It is important when working with normal vectors

If a triangle's vertices v_1, v_2, and v_3 have a CCW winding order,
the cross product $\overrightarrow{v_2 - v_1} \times \overrightarrow{v_3 - v_1}$ will be the outward normal vector

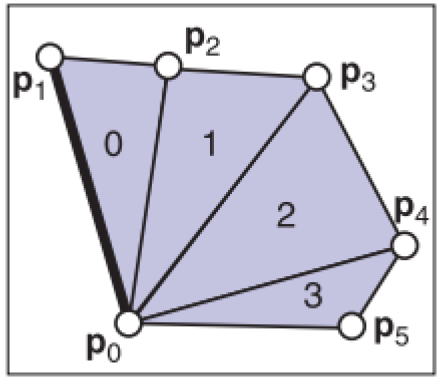


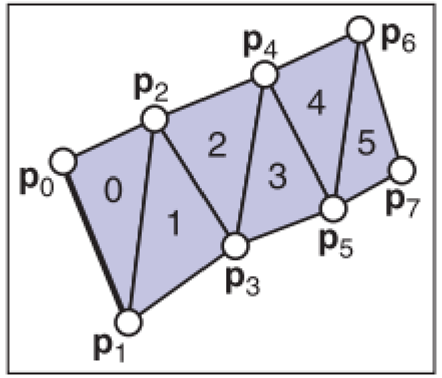
Triangle Strips and Fans

Compactness is the major goal is the design of data structures for meshes

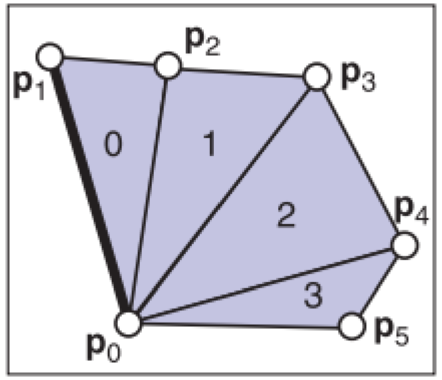
Indexed meshes most commonly used representation for this reason

Can reduce the size of the mesh even further using ***triangle strips or fans***





Triangle Fans



In an indexed mesh, the triangles array would contain [(0, 1, 2), (0, 2, 3), (0, 3, 4), (0, 4, 5)]

We are storing 12 vertex indices, although there are only six distinct vertices.

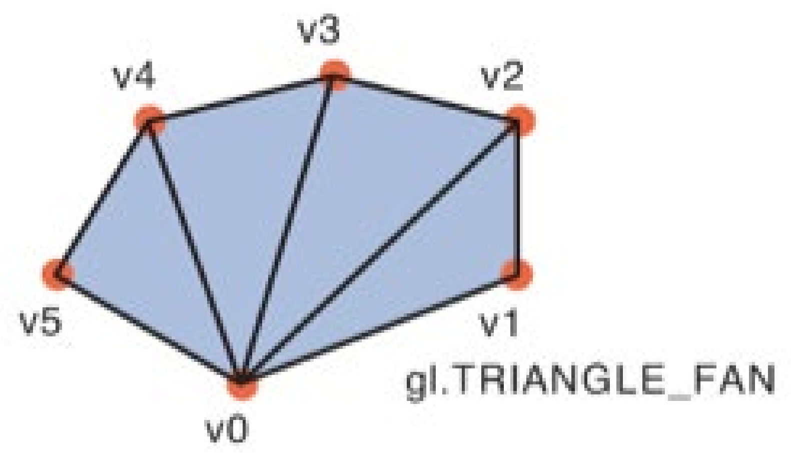
In a triangle fan, all the triangles share one common vertex
The other vertices generate a set of triangles like the vanes of a collapsible fan

The fan in the figure could be specified with the sequence [0, 1, 2, 3, 4, 5]

The first vertex establishes the center
Subsequently each pair of adjacent vertices (1-2, 2-3, etc.) creates a triangle.

A triangle fan with N vertices will generate N-2 triangles

Fans in WebGL



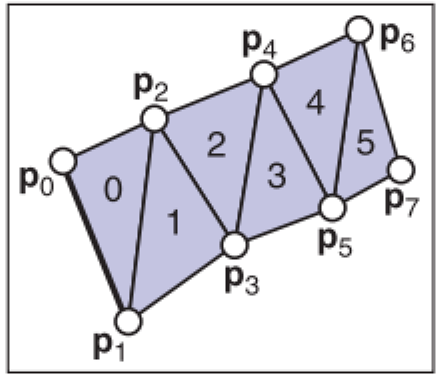
```
void gl.drawElements(mode, count, type, offset);
```

```
void gl.drawArrays(mode, first, count);
```

In either case, just use gl.TRIANGLE_FAN as the value for *mode*

```
vertexPositionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
var triangleVertices = [
      0.5, -0.5,  0.0,
      1.0, 0.5,  0.0,
      0.0,  0.5,  0.0,
     -0.5, -0.5,  0.0,
];
gl.bufferData(gl.ARRAY_BUFFER, new
      Float32Array(triangleVertices), gl.STATIC_DRAW);
vertexPositionBuffer.itemSize = 3;
vertexPositionBuffer.numberOfItems = 4;
....
gl.drawArrays(gl.TRIANGLE_FAN, 0,
vertexPositionBuffer.numberOfItems);
```

Triangle Strips



The triangle strips are useful for a wider range of meshes than fans.
Vertices are added alternating top and bottom in a linear strip

The triangle strip in the figure could be specified by [0 1 2 3 4 5 6 7]

Every subsequence of three adjacent vertices (0- 1-2, 1-2-3, etc.) creates a triangle.

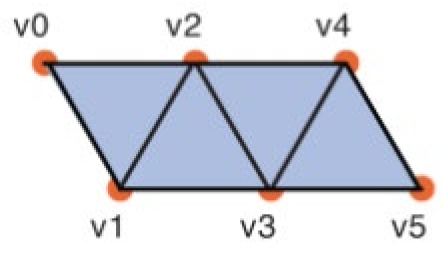
For consistent orientation, every other triangle needs to have its order reversed.

In the example, this results in the triangles (0, 1, 2), (2, 1, 3), (2, 3, 4), (4, 3, 5), etc.

For each new vertex that comes in,
the oldest vertex is forgotten and the order of the two remaining vertices is swapped.

A triangle strip with N vertices
will generate N-2 triangles

Strips in WebGL



gl.TRIANGLE_STRIP

```
void gl.drawElements(mode, count, type, offset);
```

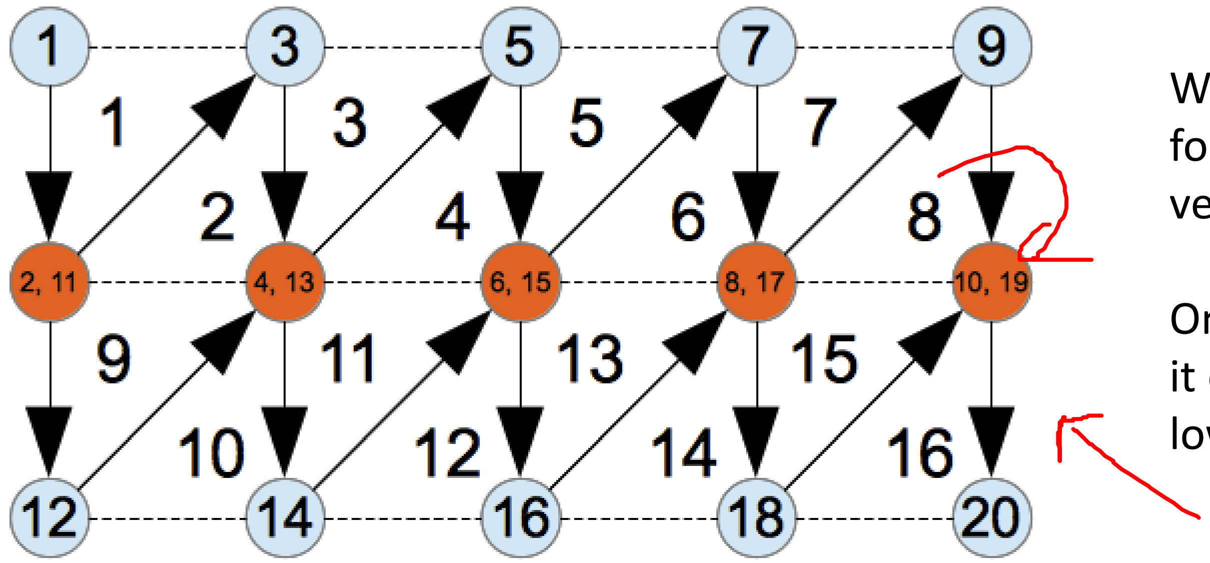
```
void gl.drawArrays(mode, first, count);
```

In either case, just use gl.TRIANGLE_STRIP as the value for *mode*

```
vertexPositionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vertexPositionBuffer);
var triangleVertices = [
    -0.5, -0.5,  0.0,
     0.5, -0.5,  0.0,
     0.0,  0.5,  0.0,
     1.0,  0.5,  0.0,
];
gl.bufferData(gl.ARRAY_BUFFER, new
    Float32Array(triangleVertices), gl.STATIC_DRAW);
vertexPositionBuffer.itemSize = 3;
vertexPositionBuffer.numberOfItems = 4;
....
gl.drawArrays(gl.TRIANGLE_STRIP, 0,
vertexPositionBuffer.numberOfItems);
```

Degenerate Triangles

Strips are limited…some meshes seemingly cannot be specified using a single strip



We can easily strip the triangles formed by the upper two rows of vertices

Once we finish triangle 8 9 10 it does not seem like can add the lower row to the same strip

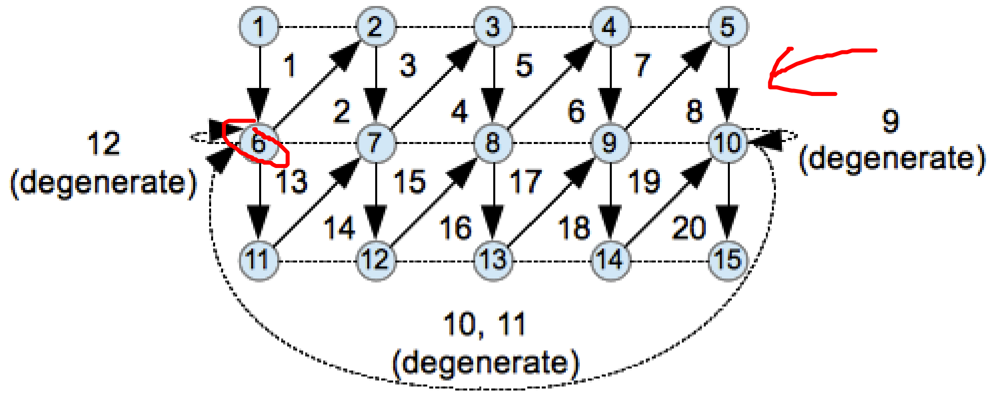
Using two strips means having duplicate vertices (the orange ones) that appear in both

It also incurs the extra overhead of having more than one buffer that needs to be transferred

…but there we can use a single buffer if we use ***degenerate triangles***.

Degenerate Triangles

A degenerate triangle has two or more vertices that are identical…so it has zero area



```
indexBuffer = {
    1, 6, 2, 7, 3, 8, 4, 9, 5, 10, 10, 6, 6, 11, 7, 12, 8, 13, 9,
14, 10, 15
}
```

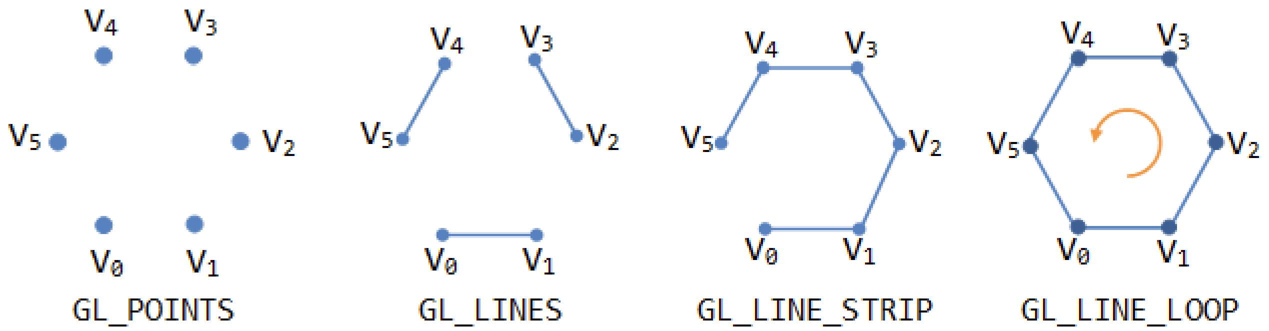
Sequence of triangles linking the two rows into one strip is then

- …
- Triangle 8 = 5, 9, 10
- Triangle 9 (degenerate) = 5, 10, 10
- Triangle 10 (degenerate) = 10, 10, 6
- Triangle 11 (degenerate) = 10, 6, 6
- Triangle 12 (degenerate) = 6, 6, 11
- Triangle 13 = 6, 11, 7

Obviously, the generate triangles waste some storage space and processing for no visible effect

But, for something like a large grid of triangles the savings of keeping a single strip may be worth it.

Point and Line Primitives in WebGL



WebGL can draw lines…sort of

The rendered quality is usually poor and varies between browsers

Lines may disappear in 3D when viewed from certain angles

It's common to use small triangle strips rather than lines for better visual quality