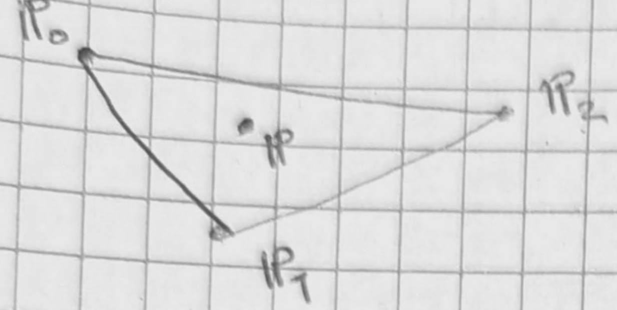
7 - Triangular meshes

Barycentric coordinates

$$\mathbb{P} = \alpha \mathbb{P}_0 + \beta \mathbb{P}_1 + \gamma \mathbb{P}_2 \qquad \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$$

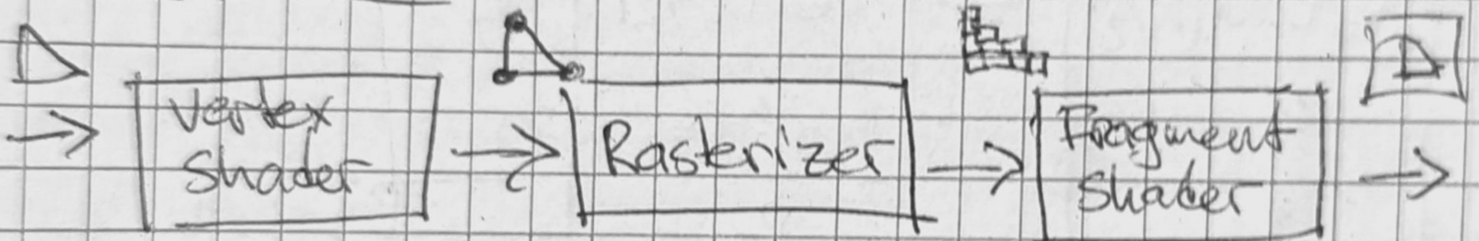


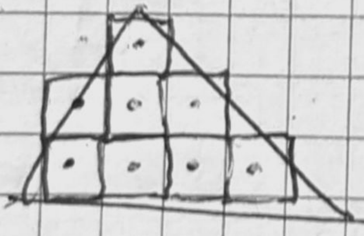
constraints: $\alpha + \beta + \gamma = 1$ ← on the plane

$$\left.\begin{array}{l} 0 \le \alpha \le 1 \\ 0 \le \beta \le 1 \\ 0 \le \gamma \le 1 \end{array}\right\} \text{inside the triangle}$$

Rasterizer can interpolate anything, like colors and normals.

GPU pipeline





Rasterizer conceptually calculates the middle of the "pixel". In reality it also computes the screen space derivatives, i.e., amount of change between each point/pixel in sets of 2x2. This is hidden from final image.

Triangular meshes

- List of vertices
  - x, y, z position per vertex
- List of triangles
  - vertex indices

Many kind of formats, .obj is one which fits how GPU works. We have list of triangles so we don't need to repeat vertices for neighboring triangles.

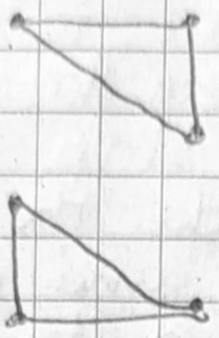
OpenGL

Each position repeated 6 times on average

- vertex attributes

positions

0	x	y	z
1	x	y	z
2	x	y	z
3	x	y	z
4	x	y	z
5	x	y	z



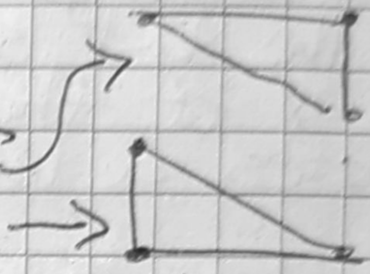
```
glDrawArrays(GL_TRIANGLES, 0, 6);
```

positions

0	x	y	z
1	x	y	z
2	x	y	z
3	x	y	z

elements

0	0	1	2
1	0	2	3



```
glDrawElements(GL_TRIANGLES, 6,
               GL_UNSIGNED_INT, 0);
```

Better to store positions once and reference them in an elements array.

Element buffer object

Only have one of this

```
GLuint ebuffer;
glGenBuffers(1, &ebuffer);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, ebuffer);
glBufferData(GL_ELEMENT_ARRAY_BUFFER,
             sizeof(unsigned int) * 6,
             elem,
             GL_STATIC_DRAW);
...
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, ebuffer);
glDrawElements(GL_TRIANGLES, 6, GL_UNSIGNED_INT, 0);
```

NB! .obj files allow to specify different triangle vertices for different attributes (pos, color etc.). The graphics API do not allow this.

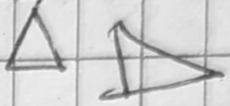
Triangle strips

We can also use triangle strips with
```
glDrawArrays(GL_TRIANGLE_STRIP, 0, 4);
```

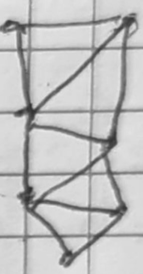
This is a more ~~the most~~ efficient way of drawing triangles, but the vertices needs to be ordered so the next vertex drawn creates a triangle with the previous two.
If we have several strips to be drawn we can use a trick in order to not paying the cost of calling glDrawArrays multiple times. We create a single strip where when we change strip, we add the last vertex of the last strip two times, and also the first vertex of the new strip two times. This creates a line which will not render.

~~We also have~~ GL_TRIANGLE_FAN which uses first and last vertex with the current.

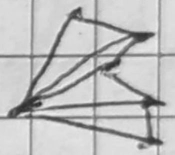
GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



All of these have a glDrawElements version, and glDrawElements(GL_TRIANGLE_STRIP ...) is the most efficient.