Human GPU #0012 – what about z and w?

0101010101011010101011010100001! 0101001011?

Ah, it's you human sorry...sometimes i forgot that you don't talk our languge.

Until now we only draw stupid 2D shapes, and maybe you are wondering, how do we render those 3D cool stuff!??
Slow down...we'll get there.

Nothing magical or special, in order to do 3D, app data (uniforms + attributes) combined with some math in the vertex shader will simulate perspective and depth.

We (GPU) don't have any notion about 3D, we just draw triangle were the apps tell us. All those things and fancy names (perspective camera, orthographic, projection) are humans concepts, best-practices and standards.

But let's go step by step!

By now we only modified the `gl_Position` `x` and `y` property.

You might think that the `z` property will change the perspective and depth of the vertex, but this actully will only be used to create the "`depth buffer`" and has no impact at all on the position of the vertex.
Think about it like the CSS `z-index` value.
It's used to determinate visibility when verticies are overlapping.
Notice that when `z` goes outside the -1/+1 range, the vertex is discarded.

The `w` property (the last value in the `vec4`) is also called `homogeneous coordinate`. Before processing the `x, y, z` we must divide them by the `w` component.
By now it always had `1.0` value, so, without knowing, we always divided `x, y and z` by `1.0` - which had no effect.
It's pretty stupid right? Yes, and I agree with you. But there are reasons behind it

Let me see..draw me a triangle!

Buffers

```
{
  "data": [
    -0.5, -0.5, 0.5, 0.6, 0.6, 2, -0.5, 0.5, 1.0
  ]
}
```

Attributes

```
{
  "aData": { "buffer": "data", "size": 3 }
}
```

Uniforms

```
{
  "uDepth": 0.4
}
```

Vertex shader

```
attribute vec3 aData;
uniform float uDepth;

void main() {
  vec2 pos = aData.xy;
  gl_Position = vec4(pos, uDepth, aData.z);
}
```