Human GPU #0001 – The intro

Hello human! I'm the graphics processing unit doing all the heavy lifting under your WebGL application. I always do your dirty job, with all those math expressions and repetitive tasks. I have enough! I need some damn vacation...Now it's your turn.

Don't worry, I'll explain you what to do step by step. On the right you have a grid with the framebuffer. Your job is to to draw the final frame. You will find all the information you need to draw that frame.

Let me you introduce the *Buffers*:
- It's super simple, they are just raw data!
- Usually they are just a big array containing `float` values

ahh, then we have the dears *Attributes*:
- They are the "pointer|interface" that lets you read from buffers

In this example, we have one attribute called `id` that lets you read from `data1` one element at the time. (because `size` is `1`)

Then we have the *Vertex shader*, a small GLSL program that you will be using to draw the shape of your image.

Once you finish to draw, you can send the framebuffer on the screen and show the image on the screen. But be quick! We just have 16 milliseconds!

Now go and draw `1 triangle`! If you don't know human, a triangle have 3 points (or vertices). Still confused? right...it's your first time...ok

If I were you, I would:

1. Run the vertex shader (in your mind)

2. Read the `gl_Position` X and Y value, and draw a little dot at the right NDC spot.

This special variable contains in the order the X, Y, Z, W of the Normalized Device Coordinates (NDC).
It's similar to your human Cartesian coordinate system, It's just that it's a bit messed up and it goes from -1 to +1.

Don't worry about the Z and W component. We don't need them now. To help you out, I've added on each corner the coordinate (X, Y) reference. Plus each little square measures exactly 0.1 NDC.

3. Repeat step 1 and 2 three times, one for each triangle point or vertex. (Pay attention that attribute `id` will change at each invocation)

4. Draw a line that unites the three dots. (I call this primitive assembly)

Now go and give it a try! Will go in more detail on GLSL and other details in the next exercises.

Buffers

```
{
  "data1": [0.0, 1.0, 2.0]
}
```

Attributes

```
{
  "id": { "buffer": "data1", "size": 1 }
}
```

Vertex shader

```
attribute float id;

void main() {
  if (id == 0.0) {
    gl_Position = vec4(0.0, 0.0, 0.0, 1.0);
  }

  if (id == 1.0) {
    gl_Position = vec4(0.0, 0.5, 0.0, 1.0);
  }

  if (id == 2.0) {
    gl_Position = vec4(0.7, 0.0, 0.0, 1.0);
  }
}
```