U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Computer Graphics
MIEIC

Project - Billboard

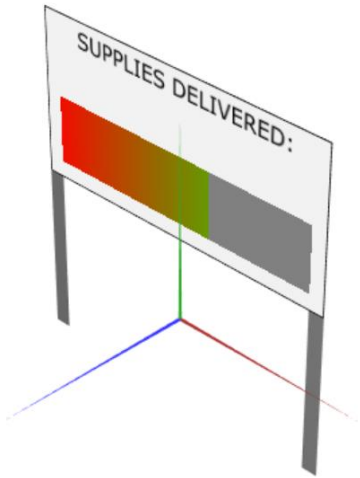
Progress bar

Teresa Matos
18-22 May 2020

MyBillboard

A billboard object containing at least four 2D planes, one of which corresponds to a **progress bar**, created with shaders:

- The **vertex shader** defines the vertices' final position as normal
- The **fragment shader** defines a colored gradient; its width matches the **% of delivered supplies**



Steps for progress bar in MyBillboard

Following the instructions in the project's worksheet to create the ***MyBillboard*** class, the steps for the progress bar are:

1. Create and apply the basic **shader files** to the progress bar's plane
2. Create **red-to-green** gradient
3. Use *nSuppliesDelivered* to define **cutoff** on the gradient

An object of ***MyBillboard*** is added to ***MyScene***, which calls its *display()* and *update()* functions

1 Shaders - Base code

Similarly to the steps for ***MyFlag***, the **examples from TP5** may be used to create the basic shaders:

texture1.vert

```
void main() {
    gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
    vTextureCoord = aTextureCoord;
}
```

uScale.frag

```
void main() {
    gl_FragColor = vec4(0.0, 0.0, 1.0, 1.0);
}
```

The plane will have a final solid color (blue)

2 Shaders – Full gradient

To create the red-to-green gradient, a final color is defined where the red and green components **vary throughout the plane, horizontally.**

The **attribute** variable `aVertexPosition` can be **passed to fragment shader** and used for this effect.

As we are not using a texture, *aTextureCoords* may not be available

billboard.vert

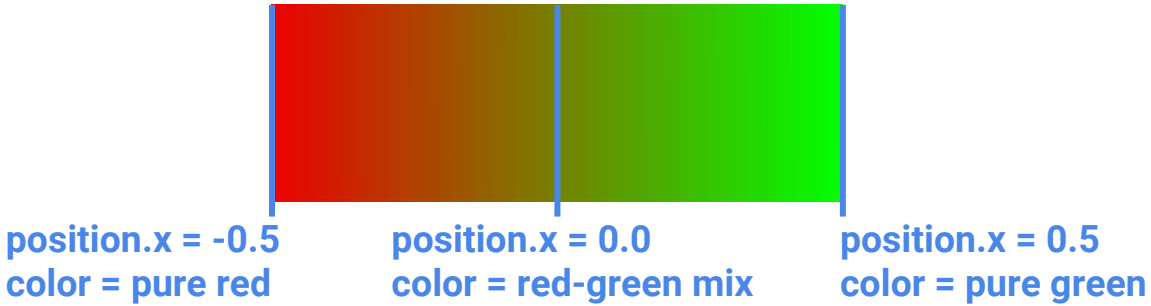
```
void main() {
     gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
     vTextureCoord = aTextureCoord;
     vVertexPosition = aVertexPosition
}
```

“Varying” variable

2 Shaders – Full gradient

In the **fragment shader**, the **final color** is obtained by defining the red and green components (`vec4(R, G, 0.0, 1.0)`), considering that:

- At the **left margin**, the color must be **pure red**
- At the **right margin**, the color must be **pure green**
- At the exact **middle** of the plane, the color must be a **mix of both**



3 Shaders – Gradient with cutoff

The *nSuppliesDelivered* variable may be passed from ***MyScene***, through the *update()* function, to the shaders as a **uniform** variable

Considering the total number of supplies to obtain a relative value:

cutoff = N delivered / 5 total

The **cutoff** corresponds to the relative width where the gradient is visible

Example:

- 0 supplies delivered – **cutoff = 0.0** – no gradient (total grey)
- 4 supplies delivered – **cutoff = 0.8** – gradient in 80% of plane's width

3 Shaders – Gradient with cutoff

The output color is:

- **Grey**, if the current position in X relative to the total width is higher than the **cutoff value**
- **Green-to-red** gradient, otherwise