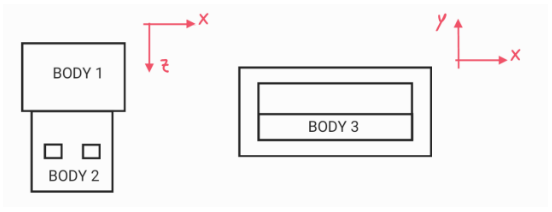
Report Homework 1

Gioele Migno - 1795826

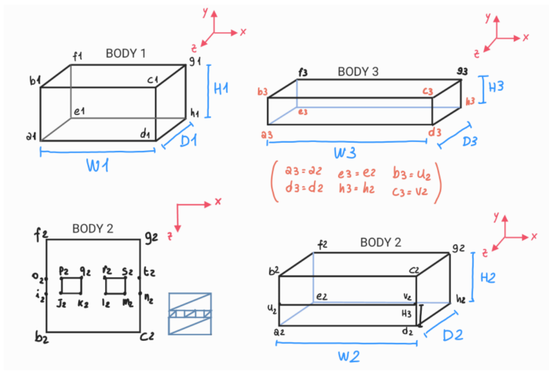
2 May 2021

1 Geometry description

The cube has been replaced with a model of a mini USB flash drive, the geometry is composed by three bodies shown in Fig. 1a, Body 1 is the case, Body 2 is the shield of the USB port and Body 3 is the pin board. Each body is built using simple quadrilaterals, the overall geometry has 32 vertices, in Fig.1b each of them are reported in black.
The normal of each quadrilateral ABCD has been computed by the cross product between the sides (normal=cross(B-A, C-B)). Since the texture used is an iron pattern, so it can be stretched without problem, the overall texture is simply mapped on each bodies' side, namely, considering a square ABCD with A at the bottom left corner, A is mapped to the texture coordinate (0,0) and C to (1,1). In a general case this method cannot be used when the image cannot be distorted to be adapted.



(a)



(b)

Figure 1: Bodies of the geometry

2 Rotation around barycenter

The barycenter (BB) of the geometry has been calculated considering the midpoint between the barycenter of the case (BB1) and the barycenter of the pin board (BB3), the weight of the shield has been neglected. The two barycenters have been computed in the following way:

$$BB1 = a1 + \left(\frac{W1}{2}, \frac{H1}{2}, -\frac{D1}{2}\right)^T = a1 + \alpha \tag{1}$$

$$BB2 = a1 + \left(\frac{W1-W2}{2}, \frac{H1-H2}{2}, -D2\right)^T + \left(\frac{W3}{2}, \frac{H3}{2}, -\frac{D3}{2}\right)^T = a1 + \beta + \gamma \tag{2}$$

The parameters like W1 D1 H1 are described in Fig.1b, the barycenters are computed with respect to $a1$ because all other vertices are defined with respect to it.
In order to easily rotate the geometry around the barycenter BB, the vertices have been calculated so as to place the barycenter at the origin of axes of the world reference frame, to do that we need to choose $a1$ such that $BB = (0,0,0)^T$ the solution is $a1 = (-\alpha - \beta - \gamma)/2$, all the operations are reported in a Jupyter Notebook (compute_vertices.ipynb).
In the HTML page, the axes of rotation, direction and the start/stop can be controlled using buttons.

3 View position and Perspective projection

The view position is defined with a spherical coordinate system, namely the eye position is computed as follows:

$$eye = (R\sin(\theta)\cos(\phi), R\sin(\theta)\sin(\phi), R\cos(\theta))^T \tag{3}$$

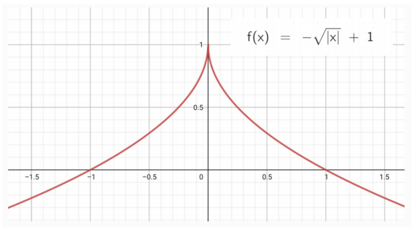
Where R indicates the radius of the sphere, namely the distance of the camera from the origin of the axes and the angles θ ϕ specify the position on the sphere surface. The eye position is used in the computation of modelViewMatrix that uses as last applied transformation the matrix built by *lookAt()* function that takes as inputs: *eye*, the position where the camera looking at (*at*), and the orientation of the camera (*up*). The parameters *at* and *up* are fixed to the origin and the y-axis respectively, instead, *eye* can be controlled with: $R \in [0.05, 10]$, $\theta \in [-90, 90]$ and $\phi \in [-90, 90]$. ProjectionMatrix is computed through *perspective()* function that uses the parameters: $fovy \in [10, 120]$ to set the field-of-view size, $aspect \in [0.5, 2]$ to define the visual ratio and, $near \in [0.01, 3]$, $far \in [3, 10]$ to determine the viewing volume, these parameters and the previous ones of *lookAt()* can be controlled in the HTML page by sliders. Using initial parameters, both the USB flash-drive and the neon tube are visible inside the view volume, by setting *near* to about 2, the object is partially outside the view.

4 Neon light

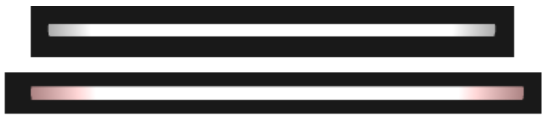
The neon light is modeled as a horizontal cylinder placed on the y-axis.
The function *build_cylinder()* used to draw the cylinder has been developed from scratch, it builds the object with the base on xy-plane at z=0, to do that it essentially starts from the points of a circle divided in several sectors (given by *compute_base_circle()*), then increasing the z-coordinate of each point it approximates the body of cylinder with quadrilaters of base equals to the size of the circle's sector and the height equals to a parameter *dh*, naturally at the end each quadrilater is built by two triangles. Once drawn the body, it builds the two bases using *draw_circles()* function that approximates them with triangles. The cylinder is placed horizontally on y-axis inside the render function.
Inside the cylinder, three lights with the same properties have been placed, their colors are white for the specular component and gray for the diffuse and the ambient ones, the proprieties chosen are the following ones:

```
var lightAmbient = vec4(0.4, 0.4, 0.4, 1.0);
var lightDiffuse = vec4(0.6, 0.6, 0.6, 1.0);
var lightSpecular = vec4(1.0, 1.0, 1.0, 1.0);
```

The emissive component has been computed assigning the color to each vertex/fragment (it depends on the render mode) considering the distance between the z-coordinate of the points and each light in world coordinates, more specifically the function used is shown in Fig2a, observe that the distance, represented by x, is always inside the interval [-1, 1], so the output is between 0 and 1. In Fig.2b are shown two examples, the first represents the final appearance of the neon light inside the scene and the second uses a red ambient light to highlight the effect.



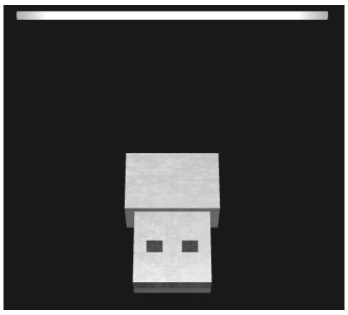
(a)



(b)

Figure 2: Function used for emissive term of the neon light and two examples

In the HTML page the neon light can be turned on/off using a button, when the light is power off, only the ambient component is used during the rendering of the tube and the USB flash drive, Fig.3 shows the scene with the light turned on and turned off. As you can see from the figures, the neon tube has a different material ambient property, and the internal face of the USB flash drive is rendered as a back face in order to highlight the holes.


(a)

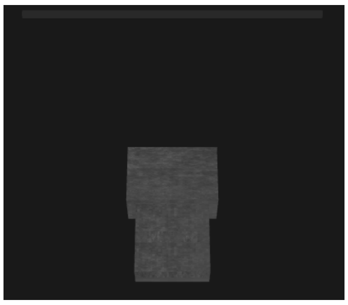

(b)

Figure 3: Scene with light turned ON/OFF

5 Material of the object

In order to give to the object appearance of iron, the material's properties have been set to the following values:

```
var materialAmbient = vec4(0.2, 0.2, 0.2, 1.0);
var materialDiffuse = vec4(0.3, 0.3, 0.3, 1.0);
var materialSpecular = vec4(1.0, 1.0, 1.0, 1.0);
var materialShininess = 100.0;
```

6 Shading models

The shading model has been implemented both per-vertex and per-fragment, in the HTML page the mode can be toggle using a button, the current modality is shown by a label (PER-VERTEX / PER-FRAGMENT).
In order to keep simple, efficient, and easily manageable the GLSL code of the shaders, the neon tube and the USB flash-drive use two different programs, moreover each of them use two different programs according to the shading model, so, overall the GLSL code is composed by four different programs.

7 Browser compatibility

The code has been tested in Ubuntu 18.04 using a python3 server (*$ python3 -m http.server*) and Google Chrome v.90.0.4430.93. Using Firefox v.88.0 the texture is not properly applied in per-vertex shading model.