Interactive Graphics: Homework 1

Fabian Humberto Fonseca Aponte | 1886565 | Interactive Graphics
MSc. Artificial Intelligence and Robotics
Sapienza Università di Roma
30/04/2020

In the present report is described the process of development followed in other to fulfill the challenges stated for the Homework 1 of the subject Interactive Graphics, this includes but is not limited to general description of the techniques considered.

GitHub Repo: https://github.com/fhfonsecaa/InteractiveGraphics_CartoonShade

1. Geometry vertex, normal and textures

Vertex

In order to get the vertex that will compose the final geometry and specially to avoid mistakes in the order and arrangement of those vertex at the moment of building up the solid, it was used the software blender to support its design following these steps:

Sketch the 2-dimensional morphology of the object and generate a solid by providing an additional dimension, smooth the surface with respect to the requirements:

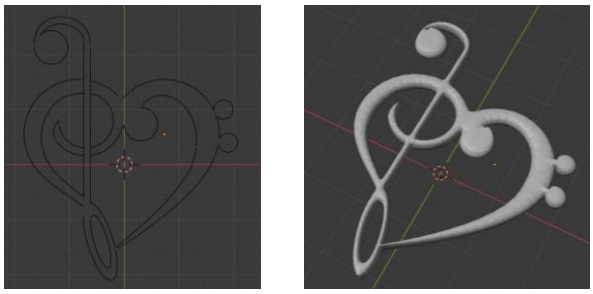

Figure 1. Planar sketch and 3-dimensional object generated from blender.

A fast inspection of the source code of the object indicated that it requires de tcomputarion of over 3500 poins to describe the 3-dimensional surface. Therefore, a much simpler form was designed using a more straight forward method, generate basid solids like piramids and cubes, change the morphology and merge them together.

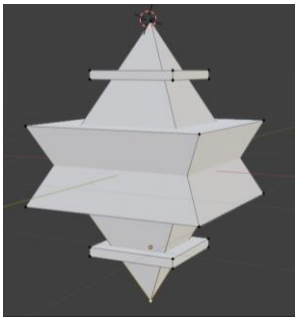

Figure 2. Simplified solid with aproximately 26 vertex.

Normal vectors

Taking into account the fact that the other matters and that in order to construct a square it will be needed two triangles, it was implemented a method to calculate the normal vectors from the three basic points that any solid will require as a primitive. This normal vector was made coincident with the other vertices of the face to homogenize the direction of the vectors whose origins are locally coplanar, that means, they belong to the same face of the solid.

Given three points p, V, and U:

$$\boldsymbol{Normal} = (\boldsymbol{V} - \boldsymbol{p}) \times (\boldsymbol{u} - \boldsymbol{p})$$

Equation 1. Normal vectors calculation from 3 points in space.

Texture component

Since the solid used for the development of this project has a combination of composite faces, that is, it has not been built exclusively with triangles. It was required to define a texture according to the difficulties related to those features.

To do this, a vector of 2D textures was defined, avoiding the CLAMP_TO_EDGE parameter, giving the composite surfaces greater regularity when constructing the mind map of the texture. Additionally, for the faces that reduce their size like the platforms, the NEAREST parameter was selected to always approximate the morphology of the upper mind map in the tree.

Figure 3. Example of texture mapping into the final solid.

2. Viewer position and perspective projection

To configure the observer's point of view, the look at function implemented in the provided library was used, keeping the distance of the eye and the angles of the spherical coordinates as free and controllable parameters.

$$eye = \begin{bmatrix} distance \times \sin(theta) \times \cos(phi) \\ distance \times \sin(theta) \times \sin(phi) \\ distance \times \cos(theta) \end{bmatrix}$$

Equation 2. Eye vector calculation with respect to spherical coordenates.

In the case of the perspective projection, the following values were used:

$$fovyView = 170 \qquad aspectView = 1 \qquad nearView = -1 \qquad farView = 0.1$$

Values that were embedded in the library function to calculate the associated transformation matrix.

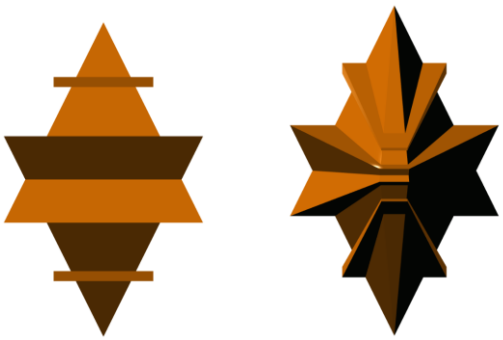

Figure 4. Example of a lateral point of view and its perspective projection.

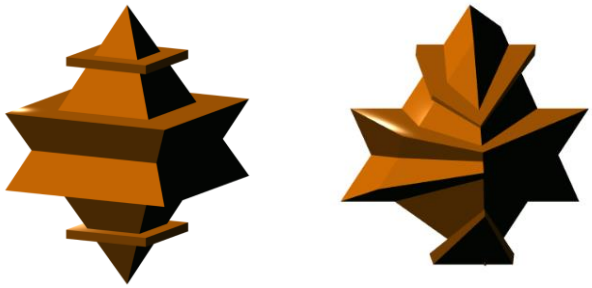

Figure 5. Example of a and aproximate isometric point of view and its perspective projection.

3. Spotlight and directional light

For the implementation of lighting (both directional and spotlight) a unification system was implemented to have the possibility of using all lights on the same object or none of them, giving the following results:

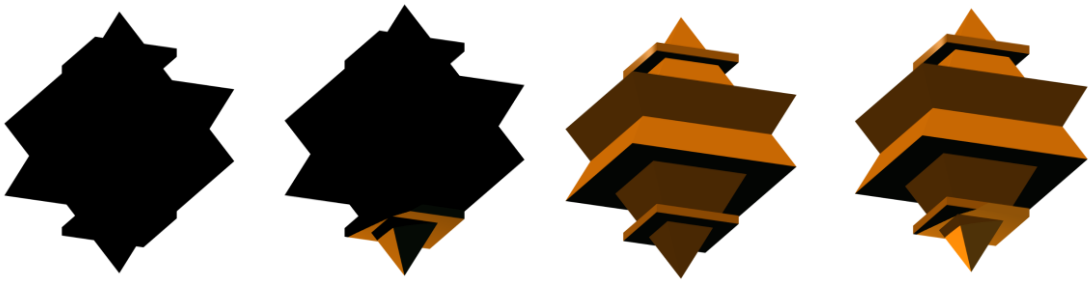

Figure 6. Respective examples of no light, spotlight, directional light and directional together with spotlight.

In the case of the spotlight it is possible to rotate it around a single axis and modify the limit of incidence on the object.

Figure 7. Example of modified aplication of the spotlight by changin the orientation and limit.

4. Material properties

Assigning values to material properties has a direct influence on the final lighting values for any shading model. These values can over saturate the color image or provide an appropriate contrast of the different light sources.

Figure 8. Example different material properties under directional light.

5. Cartoon shading

In the case of the cartoon shading, the implementation was extended to maintain the influence of the light sources, for which it was necessary to define a new branch in the if else conditional tree and to calculate the constants C_i and C_s together with the use of the parameter K_d for conditional filtering.

$$C_i = a_g \times a_m + a_l \times a_m + d_l \times d_m$$

$$C_s = a_g \times a_m + a_l \times a_m$$

$$K_d = \max\{\bar{L} \cdot \bar{n}, 0\}$$

Equation 3. Set of equations for cartoon shading model.

Figure 9. Example of cartoon model for the combination of the possible light sources.

6. Texture mapping

As a final stage in the development of this project, the mapping of textures was integrated within the unification of light sources, giving the following results:

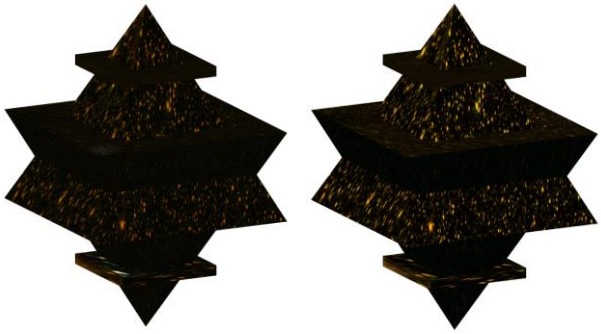

Figure 10. Example texture mapping under directional light for Phong model (left) and cartoon model (right).