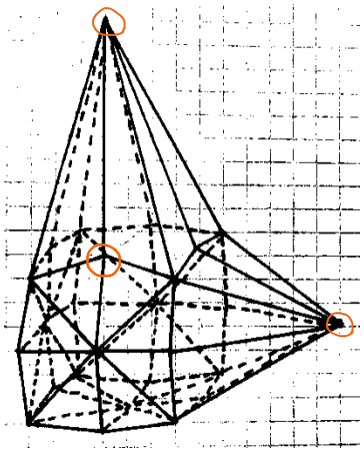
Interactive Graphics
Homework 1

Flavia Monti, 1632488

1. **Replace the cube with a more complex and irregular geometry of 20 to 30 (maximum) vertices. Each vertex should have associated a normal (3 or 4 coordinates) and a texture coordinate (2 coordinates). Explain how you choose the normal and texture coordinates.**
   The cube is being replaced with a more complex figure with a total number of vertices of 26. The new figure is like a cube where in each face there is something like a pyramid, the face on the top has the summit in a position much higher than the rest and the same with the summit of the pyramid on the right face. One vertex of the face on the top is positioned inside the 'cube'.
   Below is represented a hand-drawn of the figure, and the vertices marked by orange are the ones described before.



   Each "face" of the figure is composed by 8 triangles and each vertex of a single triangle has same normal. Each normal is composed by the formula $(c - b) \times (c - a)$ where (a, b, c) are the vertices of the triangle.
   The texture coordinates are assigned using (0,0), (0,1), (1,1) and (1,0).
2. **Add the viewer position (your choice), a perspective projection (your choice of parameters) and compute the ModelView and Projection matrices in the Javascript application. The viewer position and viewing volume should be controllable with buttons, sliders or menus. Please choose the parameters so that the object is clearly visible.**
   The ModelView matrix is generated using the lookAt. This function takes three parameters as input: eye, at and up. Eye is the parameter identifying the position of the camera and is computed with the values of phi and theta, these last two parameters define the rotation of the camera, so that al faces of the object can be observed. Phi and theta are not constant and can be updated with buttons. At and up parameters are constant.
   The Projection matrix is computed using the perspective function, this function take as input four variables, fovy, aspect, near and far. Near and far can be editable with buttons.

3. **Add two light sources, a spotlight in a finite position and one directional. The parameters of the spotlight should be controllable with buttons, sliders or menus. Assign to each light source all the necessary parameters (your choice).**
   The directional light is a light that has light rays having all same direction. It is defined by the position plus ambient and diffuse terms. The position is a vec4 where last value is 0.0 to indicate that it is a directional light.
   The spotlight is a light source having light rays in a specific direction and only what is inside a specific interval is illuminated, rest is dark. A spotlight is defined by many parameters: position, direction, cutoff and an angle theta. The angle theta is the parameter that define the radius of the interval to illuminate and cutoff is the value that determines how much the intensity of the light drops off. The formula used to compute the intensity of light of the spotlight is $I = I \cdot cos^{e}(\phi)$, where e is the cutoff and ϕ is the angle between the direction and a vector from light source with the direction to a point on the surface of the object. ϕ is calculated using dot function between DS and -LS, DS is the normalized vector of the direction of the spotlight while -LS is the normalized vector in the direction to a point in the surface. An attenuation factor is also used, in particular it is multiplied to the intensity of the spotlight. The attenuation is calculated with this formula: $\frac{1.0}{c+l \cdot d+q \cdot d^2}$, where c, l and q are respectively the constant, linear and quadratic factors and d is the distance from the light source to a point on the surface. The constant factor c is constant to 1.0, because otherwise the effects produced with the combination of the other components produces some strange illumination. Also a check on the values taken by the linear and quadratic factors is done in the fragment shader. It is noticed that if the value of one of these two components is less than 0.0 the effects that it produces are not good, in particular they produce a "circle" white (if equal to 0.0) and then black (if less than 0.0). So their values are set not to go under 0.0. In addition to these parameters also ambient and diffuse terms are defined.
   Cutoff, angle theta, direction and relative values for the attenuation can be manipulated with buttons and sliders.
4. **Assign to the object a material with the relevant properties (your choice).**
   For what regard the material properties of the object are defined two terms, the ambient one and the diffuse one.
   Specular and shininess are not defined because they are not used in the shading model.
5. **Implement a per-fragment shading model based on the shading model described at the end of this document.**
   The shading model used is the one that use the algorithm of Simple Cartoon Shade. The shading is computed in the fragment shader, so each fragment/vertex receives the corresponding result. This algorithm compute the illuminated diffuse color $C_i = a_g \times a_m + a_l \times a_m + d_l \times d_m$ and the shadowed diffuse color $C_s = a_g \times a_m + a_l \times a_m$.
   a_g is the global ambient light, a_m and d_m are the ambient and diffuse components of the object's material, a_l and d_l are ambient and diffuse components of the light. These two value (C_i or C_s) are assigned to fragment depending on the value $Max\{\bar{L} \cdot \bar{n}, 0\}$, if this value is greater or equal to 0.5 the value assigned to the fragment is C_i otherwise is C_s. The global ambient light a_g is defined as an additional variable in the Javascript file. Globally there are four ambient components: global ambient light, ambient component of the directional light,

ambient component of the spotlight and ambient component of the object's material. The values assigned to the global ambient light are very low.
C_i and C_s are calculated for the two different type of source lights and at the end are added together with the global ambient light effect on the object's material.

6. **Add a texture loaded from file (your choice), with the pixel color a combination of the color computed using the lighting model and the texture. Add a button that activates/deactivates the texture.**
   An image 64x64 is loaded in the html file and used as texture for the object. In the Javascript file the image is elaborated by configureTexture function that create the texture object and set up all the parameters.
   A button to activate and deactivate the texture is put in the html file, it modifies a flag that is checked in the fragment shader.

References:

- Angel E., Shreiner D., Interactive Computer Graphics. A Top-Down Approach with WebGL 7th edition, Pearson, 2014
- Lake, A., Marshall, C., Harris, M., & Blackstein, M., Stylized rendering techniques for scalable real-time 3d animation. In Proceedings of the 1st international symposium on Non-photorealistic animation and rendering, 2000, June
- https://www.khronos.org/registry/OpenGL-Refpages/