Representing a Scene of Objects

Fodor Zsófia

Compute Science

2020

Contents

1. Subject specification

OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

When running the project, the user enters in a small village containing a house, with some animals, a car, a fountain representing the 'center' of the village and some wolves indicating, that it is close to a forest, where wolves may appear.

The user has the possibility to traverse this scene using the mouse and the well-known control keys WASD. The scene contains some moving, rotating objects, the user also has the possibility to move some objects in the scene, with the specified key.

There are also two light sources provided, and the color of one of them can be changed using the M key from the keyboard, form a yellowish-white(neutral) color to green and viceversa.

2. Scenario

2.1 Scene and object description

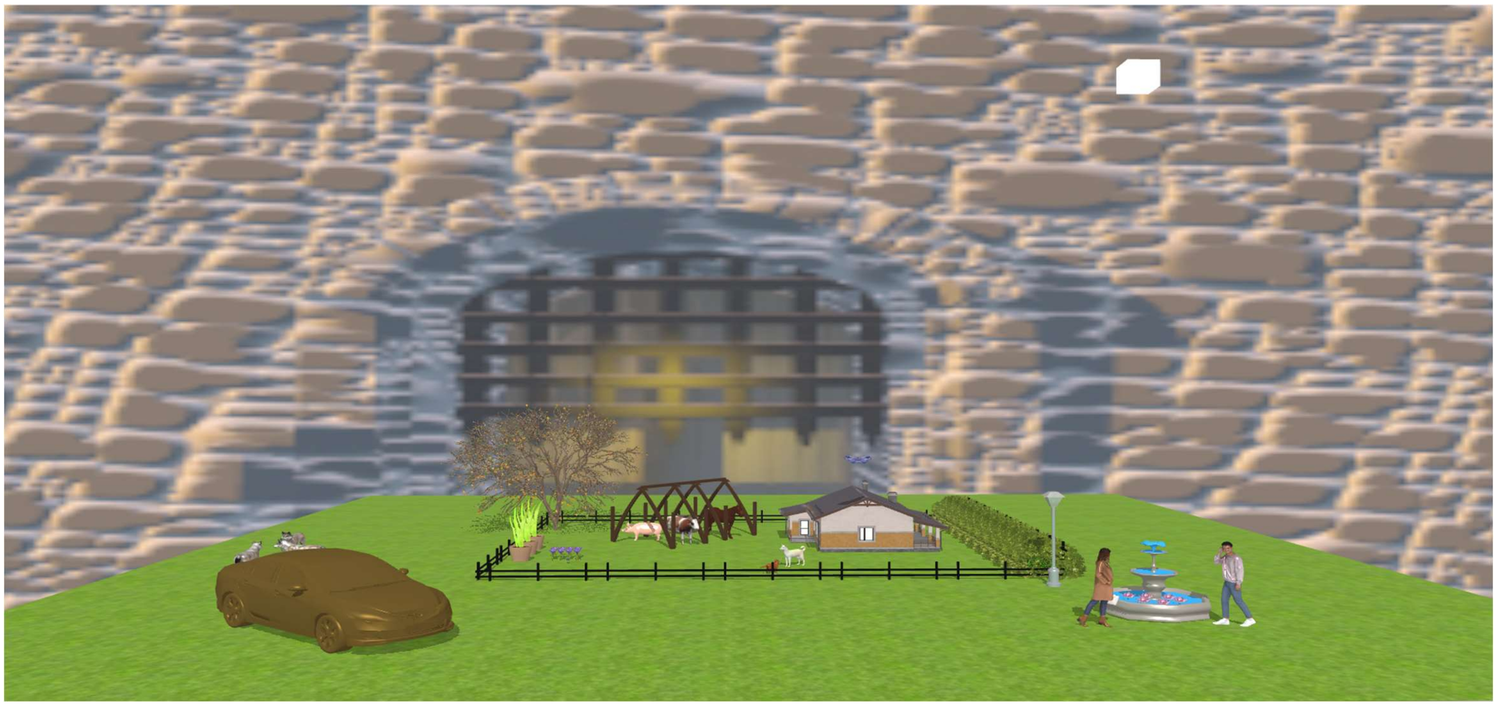
Like I specified above, the user is presented a typical village house, and its surrounding.
The house is surrounded by a fence, which has some bushes grown on it, on one of the four sides. There is a village house, in front of which there is a dog and a cat playing. There is also a barn-like object, in which the other animals are located, namely: the horse, a cow, a pig and a sheep.
There are also some plants in pots, and some flowers growing from the ground. The yard of the house has also a tree, and there is also a butterfly flying above the house.
Outside the yard, there is a red car, some wolves, representing the idea that villages are usually surrounded by a forest, and there is often the problem of wolves attacking their animals. On the other side of the scene there is a fountain, surrounded by a woman and a man speaking on the phone. The fountain contains some water lilies.
Used objects: the static objects were included as a single object created in the Blender application.
This static scene contains the following objects:

- 3 wolves,
- a fountain,
- a woman,

- a man,
- a car,
- a barn,
- a cow,
- a horse,
- a sheep,
- a pig,
- 3 flowerpots,
- 6 flowers,
- a tree,
- a house,
- fences,
- bushes,
- water lilies

The dynamic objects, that were added separately are the following:

- dog,
- cat,
- butterfly,
- lamp.



2.2 Functionalities

- When entering the scene, the user can **traverse the scene** using the mouse and the WASD buttons.
- The user can **change the representation of the scene** from polygonal to wireframe and pointframe, with the C, V and B buttons.
- Pressing the M button, the user will **apply another light source**, having the color green.
- Pressing the X and Z button, the user can **move the cat and the dog** objects.
- Pressing the J and H buttons, the user can **change the location of the light** source and by this also the shadows.
- Pressing the O button, a flower will grow, then wither
- A butterfly is always flying above the scene in a circular motion.

3. Implementation details

3.1 Functions

There multiple algorithms used here, that were already implemented, or presented in the laboratories.

For example loading the objects and textures mapping, was already provided in the Model3D class. Here the vertices and coordinates are read from the .obj and .mtl files.

- Animations
  When running the application, the user is placed in the middle of the scene, and with a camera rotation the whole scene is presented, then the camera goes to a distance view, so the user can traverse the scene the way he/she wants, using the keyboard and the mouse.

- Lighting
  For the lighting we use the Phong model presented in the laboratories, which uses the three different light components (ambient, diffuse, specular), to create the whole lighting effect.

- Shadow generation
  For generating shadows, I used the Shadow Mapping technique. The steps of this algorithm are the following:
    - Rendering the scene from the the light's point of view, and store depth values in depth map.
    - Rendering the scene from the camera's point of view, and we compare the depth of each visible fragment with the values stored in the depth map.

3.2 Graphics model

Objects are the base components of an OpenGL application. They are loaded in a pipeline of .obj and .mtl files in the application. In order for these objects to be loaded their normal must be computed.

3.3 Data structures

Many structures, such as Camera or Model3D were already provided. Also arrays and vectors were used for the rotation, translation, scaling of objects.

Also a struct was implemented in the fragment shader, for the second light source, containing a position vector, color, ambient, diffuse and specular components.

3.4 Class hierarchy

The used classes are the ones already included in the starting project, with the mention, that I implemented some additional functionalities. Below I will provide a brief explanation of each class:

- **SkyBox** class – used for loading & drawing the skybox.
- **Shader** class - using shaders in different situations.
- **Camera** class - transformations of the camera, such as moving and changing the direction, rotation.
- **Mesh & Model3D** work together to offer an interface for drawing textured objects onto the screen. Pipeline: Model3D reads object, Mesh contains the draw function needed to draw an object.
- **Main** class – controller of the other classes, here meet all the functionalities.

4.Graphical user interface and user manual

1. User manual

W	Move camera in front, bringing the objects closer
A	Move camera to left
S	Move camera to back, distancing form the scene
D	Move camera to right
X	Dog and cat move in opposite direction
Z	Dog and Cat move closer to each other
C	Polygonal representation
V	Wireframe representation
B	Point frame representation
J	Moving the location of the light source
H	

M	Changing the color of the second light source
Moving to the left	Moves the camera direction to the left or right, up or down
Moving to the left	Moves the camera direction to the right
Moving forward	Moves the camera direction up
Moving backward	Moves the camera direction down
O	Grow flower, if it reached a specified size, it will wither, and a new flower will appear in its place
R	Reset camera to initial position

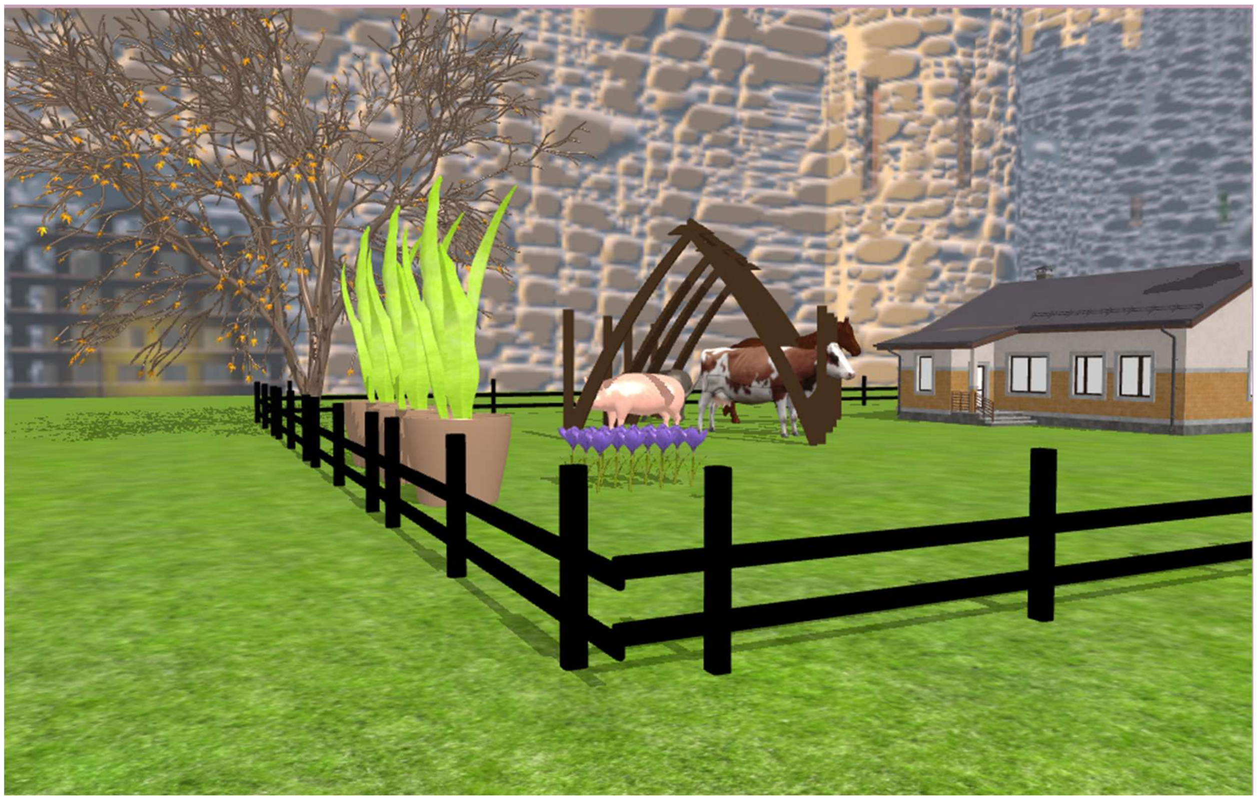


Fig2. Picture from the side of the house, representing the flowers, animals, and the tree

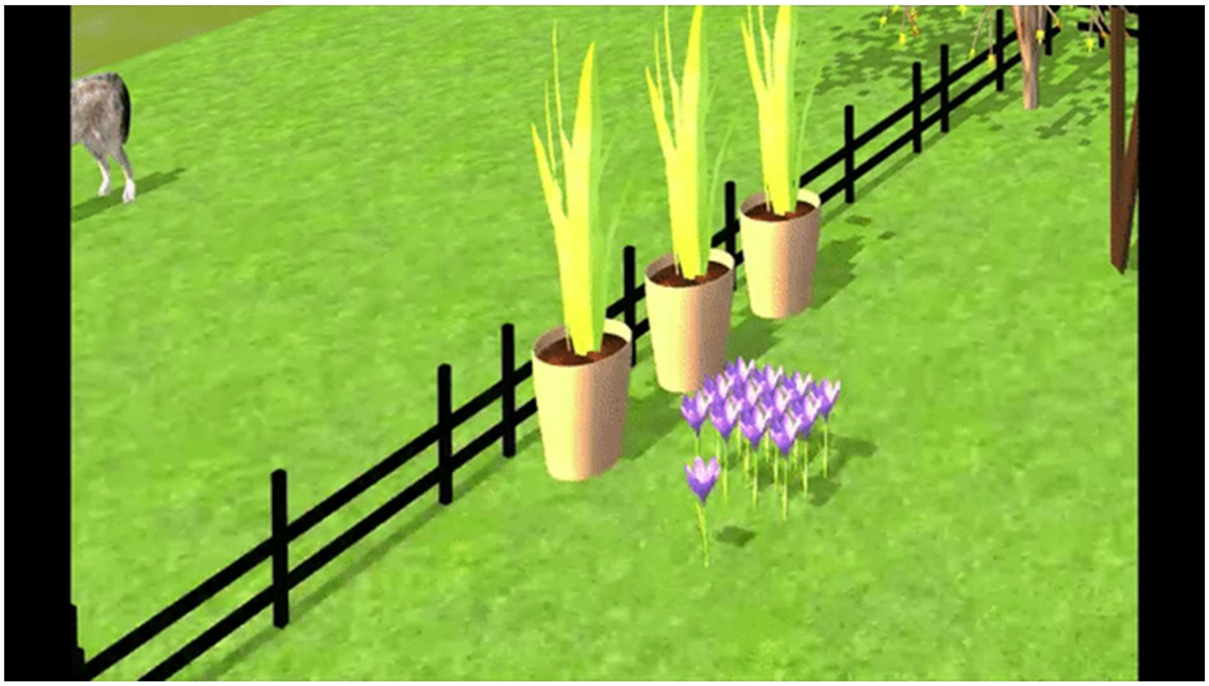


Fig3. Illustration of the flower, that grows, and after a specified size it withers.

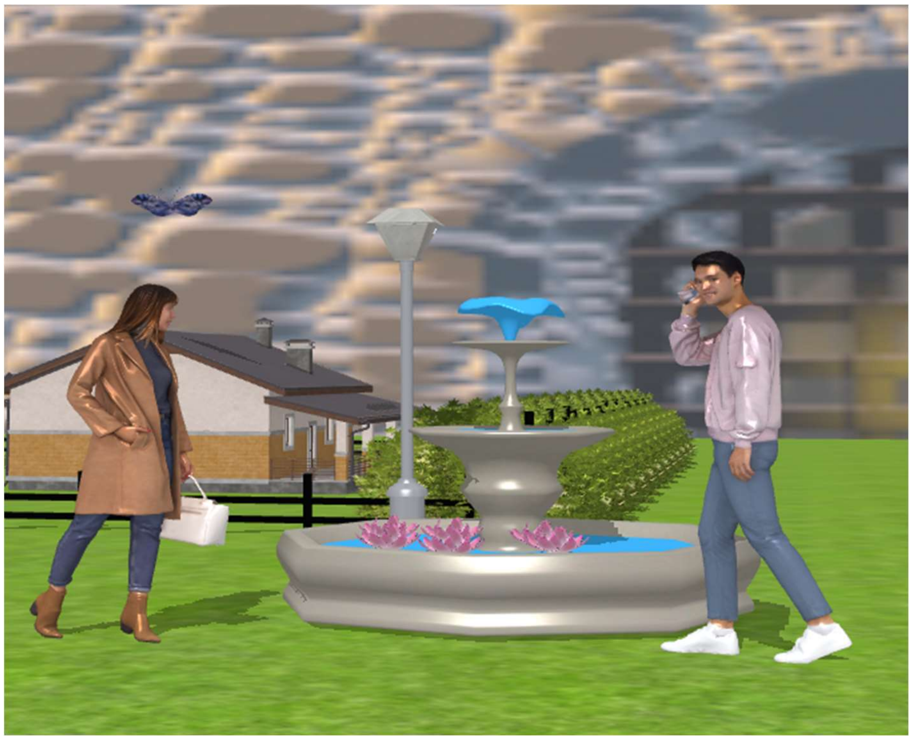


Fig4. Picture of the fountain and the man and woman beside it

5. Conclusion and possible future developments

As in case of any other project, mine also could be improved in the future. First of all, it would be a good idea, to add a point light, so that the streetlamp could be turned on or off.

Second of all, the movement of the cat and the dog could be done in a more realistic way, meaning that the legs should be animated. Another aspect that could be improved is the camera movement, to provide much more functionalities regarding it.

Another improvement could be improving the camera motion, for some reason it is very slow, can be because the amount of objects loaded.

There could also be done some collision detection, so the user cannot go through objects. Fog could be also added to the scene to make it more realistic.

6. References

1. https://www.turbosquid.com/Search/3D-Models/free
2. https://free3d.com/
3. https://www.cgtrader.com/free-3d-models/
3. https://learnopengl.com/
4. https://moodle.cs.utcluj.ro