

Documentation “Museum Heist”

Marija Markovic 01527717
Georgia Moldovan 01635273

Introduction

Museum Heist is a first-person game, in which the player's character is a thief that must roam through a museum in order to steal specific works of art. The game was developed using Visual Studio, in C++ and OpenGL.

Features

1. Camera

In this game we used a first person camera that can easily be adjusted with the help of the mouse or the touchpad.

2. 3D Geometry

The 3D Geometry used in the game, such as the museum, the paintings etc., was modelled in Blender, exported as .obj files and with the help of Assimp imported in our game. The statues were downloaded from the open-source website <https://free3d.com> but processed in blender to reduce the number of vertices and faces. Some objects like the globe, or some pedestals are created with methods provided in the framework.

3. Win/Lose Condition

The objective of the game, as its name implies, is finding and stealing some pieces of art. The player must navigate through the museum, which is more or less a maze, and find some specific objects, but has limited time to do so. If the player finds the objects before the time runs out, he wins and otherwise he loses.

4. Intuitive Controls

As many other PC games of the first-person genre, the player controls the movement with the WASD keys and the camera with the help of the mouse (or touchpad). Interaction with the objects is made possible through the left mouse button. By pressing the TAB-Key the player can switch into fullscreen mode and by pressing Q he can turn around at 180°. By pressing the B-Key the player can turn the bloom-effect on and off. By pressing the + and - signs on the numpad, the player can accelerate and decelerate his movements speed.

The controls are implemented using polling and the movement of the player is framerate-independent.

5. Textures

For texturing we used multiple DDS files that were attached to the objects as 'Materials' using UV coordinates and vertex and fragment shader.

We made use of the already implemented 'Texture' and 'Material' classes from the ECG framework for loading the textures to a specific object.

6. Illumination

For illuminating the scene we used 1 directional light and 12 point lights. The point lights are placed either in the middle of the room, in order to illuminate more paintings simultaneously, or above paintings and/or statues. The lights are then passed on as uniforms to the shader we used for texturing our objects.

7. Moving Objects

Except for the first person camera, our game has a moving globe and the hierarchical animation. The movement is framerate-independent. Globes rotation speed can be easily adjusted through a parameter, while the hierarchical animation has a constant, non adjustable movement speed.

8. 2D Text

We used a 2D text over our 3D scene to display some information that is relevant to the game, like the timer, some hints regarding the gameplay and the win/lose notifications, etc. (a very basic HUD). We used an external library called freetype.

9. Additional Libraries: PhysX, Assimp, Freetype

9.1 PhysX

We used the Nvidia PhysX engine for character control, collision detection and picking objects. We used two collision detection shapes: convex hull and triangle mesh. Triangle mesh is used for the maze as it is a single object and not an assemblage of wall box objects. Convex hull shape is used for all other objects. Capsule controller is used for the camera, and raycasting for picking objects.

9.2 Assimp

Assimp is a 3D model import library we use in our game. We have a single method that we used for loading all models, loadModel in GameObject.cpp

9.3 **Freetype** was used to render the 2D Text

Effects

1. Shadow Mapping with PCF

Our game scenery is not well suited for shadow mapping. With one light source, walls would cast shadows over the museum rooms and objects inside, and that is why we decided to display shadows only for certain objects. These objects include the statue of buddha, the statue of a woman, and rotating sphere. For implementation we used code parts from <https://learnopengl.com>.

2. Video Textures

Video texture can be seen in the first room on the right side down the hall, on the right side on the wall. Video texture consists of 5 textures changing themselves in equal time periods. This switching is framerate independent.

3. Cel Shading

Cell shading is to be seen on a couple of objects. Statue of the Easter Island Head and its pedestal, and the statue of a standing man.

4. Hierarchical Animation

Hierarchical animation consists of a central sphere moving up and down on a thin cylinder, and four cubes following it are rotating around it.

5. Bloom

We used blur on a rectangle to give the illusion of a window. As mentioned before, our scene is not made for such effects and it was a little difficult to find a meaningful way to integrate them in the game. The bloom effect is turned off at the beginning of the game, and by pressing the B-Key, the user can turn the effect on. For implementation we used code parts from <https://learnopengl.com>.

Here is a map of where in our game you can find each effect.

- 1. Video Texture
- 2.3.4. Shadow Maps
- 3. Hierarchical Animation
- 5.7. Cell Shading
- 6. Bloom

