1800573 Marcus McGinn

Documentation

CMP 405 – Tools Programming

1. Introduction/Summary of features

As part of this assessment, it was required to use the tool framework and develop extensions to the functionality and usability of the tool. To accomplish this, research was undertaken around game development tools, specifically the unreal engine 4 as it is the most widely used in the games industry (Toftedahl, 2019). The main focus of this project in particular was adding onto the functionality of the camera and its usability in order to increase the functionality and usability of the project.

To accomplish this task several additions to the framework were added including:

* improved camera movement and control
* object selection
* object highlighting
* making the camera focus on and orbit around selected objects
* multiple different cameras
* object manipulation saving the positions of objects in the scene
* wireframe mode

Controls

W - Forwards

S – Backwards

A – Rotate Left

D- Rotate Right

Q- move up

E – move down

F-toggle wireframe mode

Mouse movement (while left mouse button is down) – Camera rotation

Double click Left Mouse Button – Select an object (double click empty space to deselect)

1 – change to camera 1 view

2 – change to camera 2 view

3 – change to camera 3 view

while object selected

W- zoom in

S – zoom out

A– Rotate Left

D- Rotate Right

Q- pan up

E – pan down

Mouse movement (while left mouse button is down) – Camera rotation

I – move object positively along the X axis

K – move object negatively along the X axis

J – move object positively along the Y axis

L – move object negatively along the Y axis

U – move object positively along the Z axis

P – move object negatively along the Z axis

N – scale object up

M – scale object down

Improved Camera Movement and Control

To improve the functionality of the camera a new camera class called “cam” was created. This new camera class uses the maths explained on the frameworks’ wiki as seen below to handle the movement of the camera.

Graphical user interface, text, application

Description automatically generated

(GitHub, 2020)

These equations are used to represent the parametric equation of a sphere and therefore, can be used to turn the camera as if it was in a rotating sphere. This allows the camera to; look up, down left and right. The controls for the camera were loosely based on the unreal engine 4s controls in the viewport (Unreal Engine, 2022). Elements borrowed from Unreals’ design include; the use of W/A/S/D keys to move the camera forward, backwards, left and right along with using the Q and E keys to move the camera up and down. However, in the case of this project, A and D don’t’ move the camera, instead they rotate the camera left and right. This was done so users could choose if they wanted to use the keyboard to turn the camera or the mouse. To rotate the camera the mouse was used similarly to Unreal so that the camera will only follow the mouse’s movements while the mouse is clicked. This allows the user to choose between moving the camera or selecting objects. It also prevents the camera from moving while the user works in another window, allowing for end users to chose when the camera moves.

Object selection and highlighting

As the purpose of this tool is for interacting with the game world, it was necessary to implement a way for the user to select objects to interact with. The code for this functionality was provided in one of the labs so it will not be covered here. The way the selection is used in this application is with the double click of the left mouse button. The reason for using the double click was to allow the user to select objects without interfering with the camera position as it would cause problems if the user was constantly accidentally selecting objects every time they wanted to turn the camera.

In order for the user to know when an object was selected, a method for highlighting the selected object was provided. This was also provided by the lecturer and will therefore, not be covered in great detail. The original example provided, highlighted objects in a blue colour that caused them to blend in with the background.

Graphical user interface

Description automatically generated

No highlight

Graphical user interface

Description automatically generated

Gold Highlight

Graphical user interface, text, application, email

Description automatically generated

CornFlowerBlue highlight against sky background

Orbit Camera

For a user to adequately manipulate objects and place them in the scene it would be useful to have the camera focus on selected objects so that a user has a clear view of what they are manipulating. To achieve this an Arcball camera was implemented, this new camera type allows the user to orbit the camera around a selected object so it can be inspected from any angle. The camera orbits around the object in a similar way to how objects in outer space orbit around planets, therefore, inspiration was taken from this process when attempting to create the Arcball camera. While an object is selected the cameras “lookat” and “lookdirection” components are set to point it at the object once lined up the camera is moved towards the selected object every frame, just as gravity attracts objects to the earth. To prevent the camera, a force must be applied away from the object to counteract the gravitational effect. This value must be larger than the gravity in order to move away. A diagram of an example of this is shown below.

Diagram

Description automatically generated

While the camera is locked in orbit around an object, the usual camera controls can be used. Q and E pan the camera up and down in relation the object, W and S partially zoom the camera in and out A, D and the mouse can all be used to rotate the camera around the orbit of the selected object. When the user wants to stop orbiting, they just have to deselect the object by clicking on an empty space. Using the same inputs as the normal camera controls was decided because the camera behaving slightly different would be less disorienting to a user than having to learn a whole new set of controls just for this specific camera mode.

Multiple cameras

When building a scene it would likely be useful for a user to be able to see certain angles with the press of a button as it could give them greater context for where objects are placed in the scene and help with the repositioning of objects. To do this, two new cameras were added, one which floats directly above the scene looking down and one far away to the right of the scene. Both perspectives give a full view of the scene from their respective angles which could help a user place objects within the context of the scene as a whole rather than just the first person view of the original camera. Below are the views of cameras two and three.

Camera two view

A picture containing text

Description automatically generated

Camera Three View

Chart

Description automatically generated with medium confidence

To implement these cameras, more members of the Cam class were created called “cam2” and “cam3” respectively. Each camera was given a unique position and was pointed towards the centre of the scene. A variable called “camselection” was used to control which camera is on display by passing their respective components into the “CreateLookAt” function. To change which camera is currently displayed the user can press the number keys to choose. Pressing one activates the default first person camera, pressing two activates camera two and pressing three activates camera three.

Object Manipulation and Saving

To manipulate selected objects is very simple as it only requires for their attributes to be changed with a keypress. While an object is selected, pressing the I and K keys will move it along the X axis, J and L will move them across the Y axis and U and O will move the object along the Z axis. This layout of keys is very similar to the controls for the regular camera. This is because it may be beneficial to a user for the movement of objects to be similar across the board. The size of objects can also be changed using similar methods. When the N and M keys are pressed, the objects will grow and shrink in size. This is done by increasing their scale value.

Regular size

Graphical user interface

Description automatically generated

Increased scale

Graphical user interface

Description automatically generated

Decreased scale

Graphical user interface

Description automatically generated

Moving and manipulating these objects would be pointless if the application was not able to save any work done. The initial framework provided, lacked this functionality as it saved from the scene graph rather than the display graph. To remedy the issue, a pointer was used to store the display graph and have it overwrite the data of the scene graph. This ensures that when the application saves it saves an up to date version of the scene.

Wireframe mode

Wireframes are a valuable tool for debugging as they can be used to tell when models have not been loaded correctly or if the overlap in some way (Sanders, 2020). Implementing wireframes was extremely simple as the draw function includes a parameter for controlling if models are rendered with wireframes or not. To make the models appear as wireframes a Boolean was used to toggle them on and off.

Wireframe offA screenshot of a video game

Description automatically generated

Wireframe On

Chart, map

Description automatically generated

Conclusion

Throughout the course of this projects development, I learned a great deal about tools programming including what goes into it, the types of considerations that must be made and ultimately how they work. I think the project was successful when it comes to adding the necessary functionality for use as a tool and that the camera was especially well fleshed out. There are a few ways the project in its current state could be improved for example, the extra cameras are totally stationary so it may be useful to make it so the user can adjust the position and orientation of the extra cameras. Being able to select and manipulate multiple objects at once is a useful piece of functionality that would greatly benefit the project if added. In future I think I would choose to focus on elements of tools programming that do not involve the camera as much, specifically the use of UI and Menus as they are an incredibly important aspect to consider because they can increase the usability of the project and make the control of the project easier for a user to understand. This opinion on UI and menus is backed up by an article on IGex Solutions’ website which stated “The UX/UI Design of the application improves the user experience and customer satisfaction that ultimately helps increase the number of users of the specific application.” (The Importance of UI/UX Design - iGex Solutions, 2019) The overall use and importance of UI and Menus is something I’d like to build on in my career as a programmer going forward to develop better programming tools in the future.

References

Toftedahl, M. (2019) *Which are the most commonly used Game Engines?*. [online] Available at: < > [Accessed 2 May 2022].

GitHub (2020) *Vector Camera: Moving from only Yaw rotation to Yaw and Pitch · savantguarde/WFFC-Edit Wiki*. [online] Available at: <https://github.com/savantguarde/WFFC-Edit/wiki/Vector-Camera:-Moving-from-only-Yaw-rotation-to-Yaw-and-Pitch> [Accessed 7 March 2022].

iGex Solutions. 2019. *The Importance of UI/UX Design - iGex Solutions*. [online] Available at: <https://www.igexsolutions.com/blog/the-importance-of-ui-ux-design/#:~:text=UX%2FUI%20Design%20plays%20an,users%20of%20the%20specific%20application.> [Accessed 8 May 2022].

Sanders, A., (2020\_ *What Is a Wireframe in 3D Animation?*. [online] Lifewire. Available at: <https://www.lifewire.com/what-is-a-wireframe-140516> [Accessed 19 May 2022].

Unreal Engine (2022)  *Viewport Controls*. [online] Available at: <https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/LevelEditor/Viewports/ViewportControls/> [Accessed 14 February 2022].