ANDREW PETERS 1502220

Abertay University

Tools Programming

CMP405

Contents

[Summary of features 2](#_Toc7729258)

[Controls 2](#_Toc7729259)

[Features 3](#_Toc7729260)

[Camera movement 3](#_Toc7729261)

[Camera Focus 3](#_Toc7729262)

[Picking 3](#_Toc7729263)

[Object Highlighting 4](#_Toc7729264)

[Copy, Paste, Duplicate and Delete 4](#_Toc7729265)

[Inspector 5](#_Toc7729266)

# Summary of features

When the scene first loads the user is presented with a view from the camera just above the plane, looking at a 90 degree angle so they are not looking directly up towards empty space. The user can then move around the scene using key presses to rotate, look up and down and move forwards and backwards. If the user selects edit form the menu at the top a number of options will be available. The user will be able to copy an object (if one has been selected), they will be able to paste the copied object as long but this is only able to be cone if an object had been copied previously. A duplication button has been added here as well which will allow the user to instantly duplicate a selected object. A button for an inspector has been added to allow the user to change the position, scale and rotation of objects.

When the user selects an object by clicking on it, a few things can happen. First is that the object will be highlighted to recognise the selected object. After this has been done the user can focus onto the selected object and this will turn the camera and point towards said object.

# Controls

Camera movement

* Press W to move forward
* Press A to strafe left
* Press D to strafe right
* Press S to move backwards
* Press Q to rotate left
* Press E to rotate right
* Press R to rotate up
* Press F to rotate down
* Press space to focus on selected object
* Press space again to switch back to the normal camera

Picking

* Point and click on an object on the screen to selected
* Click on another object to switch selection
* Click anywhere apart from another object to deselect

Object Highlighting

* Only happens when an object is selected
* The object selection will be highlighted
* Highlight selection will be turned off when click elsewhere on screen that is not an object

Copy

* Click on an object to select it
* Click on edit from the menu
* Click on Copy from the edit menu

Paste

* Can only be done if an object has been previously copied
* Click on edit from the menu
* Click on paste from the edit menu

Duplicate

* Click on an object to select it
* Click on edit from the menu
* Click on duplicate from the edit menu

Delete

* Click on an object to select it
* Click on edit from the menu
* Click on delete from the edit menu

Buttons

* Click the button with the smiley face to save the scene
* Click the button with the line through it to view the selected object without the wireframe on

Inspector

* Pint and click on an object to selected it
* Click on edit from the menu
* Click on inspector from the edit menu
* Type in how you want the object to be changed

# Features

## Camera movement

The camera can be given commands after the screen has been loaded, by using these commands the user can make the camera rotate left, right, up and down. It can also move forward backwards, left and right. Each of the camera movements have been added to give the user the ability to easily move around the scene and get an entire overview of the level. This would come in very handy for the content creation and the user would be able to see where objects have been placed and how they would want the level to look. By having the ability to rotate as well, gives the application more freedom and has the possibility accurately place the camera to the position the user wants.

A new camera class was made to handle the functionality of this feature. Variables were created in the header file that stored the position, rotation, the lookat etc. New variables were created in the Input commands for manual rotation that allowed for a freer movement. The update method of this class would take the rotation variables of the camera and apply a rotation rate to them when called. A function was made to return the camera position, and another returned the view matrix. Because of this only a few lines of code needed to be added to the game class, in the update only 2 things needed to be called, the camera classes update and its function to get the view matrix. The renderer just needed to be passed the position so it could it could print the camera location on screen.

## Camera Focus

The user can focus on the selected object. This will allow them to find the object they want without having to find it manually. It can be very handy for future progress in WOFFC as once the scene gets bigger, the user might let the camera drift away from the location, but this feature allows them to quickly locate a lost scene object and rotate around it to its location.

This was simple to implement as the camera class only needed to be modified slightly. A new function was made to create the lookat matrix for the object. It took in the input commands as well as vector 3 containing the object selected position. The function simply calculated the lookat direction and then used the object position as the lookat point and this was returned to the game class. The update method for this class simply did a check to see if the space bar was press and if it was then the view matrix be that of the newly created function in the camera class, if it wasn’t then it would use the other view matrix function instead.

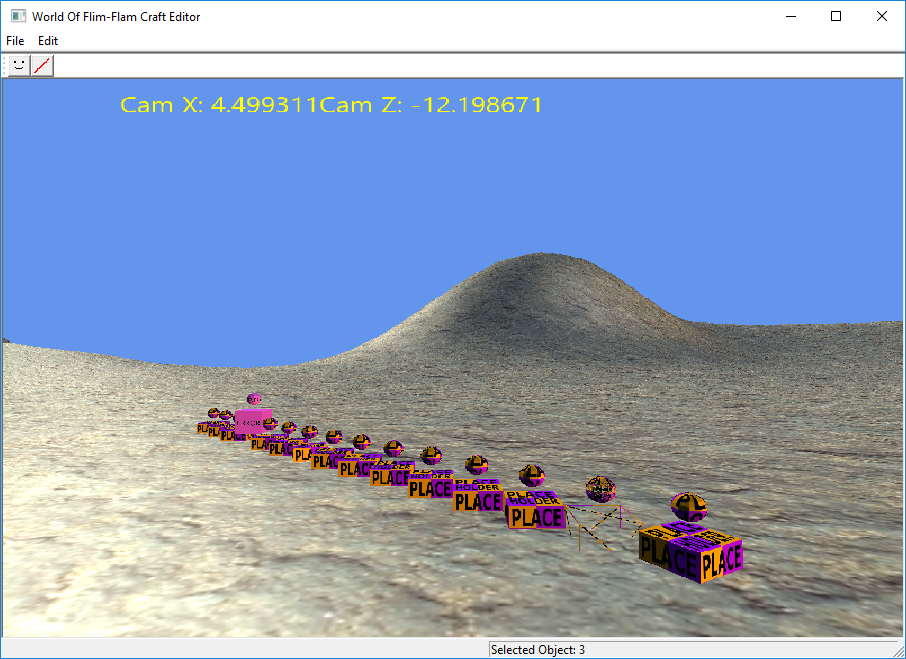
## Picking

The picking is used to select an object. When an object on screen has been clicked a notification on the bottom right had corner tells the user what object within the list has been selected. This is one of the most useful things that the user has access to. It is useful for the WOFFC game as it allows for the manipulation of objects and most of the functions within the application.

The picking feature is used for almost everything in the application. When a user points and clicks on the scene the picking function in the game class cycles though the display list to check and see if the ray from the point of the click has intersected with anything. If no object was selected prior to the click and an object was found then the object will be registered as having been selected. If there are multiple object that are found, as the ray attempts to find every object when the mouse was pressed even if some are behind the first one, the function will find the object with the closest distance to the camera making the actual one selected.

## Object Highlighting

When an object has been picked, the selected object then turns into wireframe mode. This allows for the user to see which object has been selected. When it has been deselected (by pointing and clicking somewhere else on the screen that isn’t another object), the object will return to its original solid state. To aid in usability of this the user can click on the button with a line through it to turn the wireframe mode on and off, this allows the user to see the object in its original state while also keeping it selected. This is useful within the WOFFC game as gives the user a visual display of what object has been selected.



Most of how the wireframe is turned on for the objects is done within the ToolMain class. Just after the mouse picking has been detected a check is made to make sure that that the selected object is not -1, it will then pass the parameter of the selected object to the game class where it will turn wireframe on and store it as the previous selected object. When another object is selected, the new object will have its wireframe turned on and the previous objects will be turned off. This makes sure that only the selected object has been turned to wireframe. A button was added to the resource view and when it is clicked the objects wireframe will turn off so the user can see the object again, this is done by having two function in the game class, one for turning the wireframe on and the other for turning it off. When the button is pressed a check will be made and depending on if it is true or false the function with switch it.

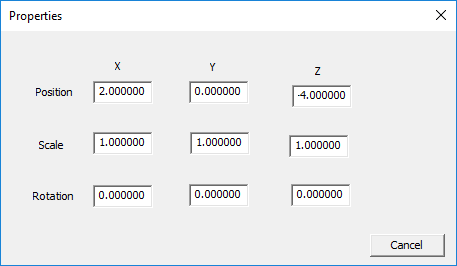
## Copy, Paste, Duplicate and Delete

These features allow the user to replicate an object in the screen. To do this the user must click on an object within the scene and go into the edit menu, from here the user can click on copy which will take a copy of the scene object, the paste button in the edit menu will put the copied object in the scene and will move it slightly on the x axis so that it can be seen. To add some friendly usability a replicate button was added in the edit menu as well this does a similar function to the copy and paste however it instantly carries out both function without a step in between. This can be used to quickly create content within WOFFC without having to go through a creation process if the objects are going to be the same.

These features follow relatively the same steps to create. First, in the resource view a new button was added to the edit menu, from here the button was given a command to run a function when pressed. Each of these functions then called another function from tool main. The paste function will do an error check to make sure that an object has been selected, from here a new scene object was created that was made equal to the selected object. The Paste feature then adds the copied object to the scene graph, as it is the last one in the vector, .Back() is used to gain access to it and is moved +1 to the right. It is given its own ID as it had the same ID as the object it was copied from, making it its own unique object. After this is all done the scene is rendered again to view the changes made. Duplicate has almost the exact same code as the Copy and Paste, it just misses the extra steps in between as they are not required. Finally the Delete feature gets the first element of the scene graph and adds on the selected object ID, once the object has been found, .erase() is called to delete the object and the scene graph is updated ad rendered.

## Inspector

When the user wants to move, scale or rotate and object in the scene they can do it through the inspector. When this menu pops up the user can select an object, after this they can use the text fields to change it however they wish. This is especially useful as it gives the user the possibility of setting up the scene how they would like in an easy simple way. When the inspector opens it will show the values that are currently assigned to the object giving a clear outline of the sizes the user should use without guessing.



This feature was implemented first by creating a dialogue box in the resource view. Static text was used to identify which of the variables would be changed on the object and edit controls were used as the input fields. Then an object inspector class was made that followed the same steps as the select dialogue class. CEdit variables were made, one for each of the input boxes as well as functions to modify them. The functions would convert the string to a float and use these with the scene graph and the currently selected item that were passed in previously. The new values replace the current values that have been entered and pass it all to the MFC Main class. The MFC Main class will finally pass it as a reference to the tool main where it will take the selected object and transform it with the new values, rendering the scene again to view the changes.

# Conclusion

While the project managed to gain some good features, it failed to achieve what was initially planned out. The attempt for the project first was to try and move objects around the scene so that tower walls could be created and snapped together, this did not work however as when the implementation was attempted, I was unable to get the mouse curser to move the objects. This was due to the fact that when the curser moved slightly the objects ended up spiralling out of control, this also happened when attempting to change the camera position through mouse movements. This forced the project away from its initial goal and other features were added instead.

In terms of future development, there is plenty of room for improvement. More features could be added, especially in the inspector. A preview of what the object looks like could be included as well as having the ability to change the textures and object itself. Once this has been included, understanding how the information is used will allow for a create object feature to be implemented.

While the inspector does allow for the objects to be manipulated, if the user clicks away from the window the project crashes. This was because it is the selected object that is being used with the window however, when an area of the scene is selected that is not an object the window tries to alter a scene object with a -1 value which doesn’t exist. A check could be put in place to make sure that it will not try to modify anything that hold this value, stopping this bug. One final bug that I am not sure why is happening is that when the inspector window is open there is a memory leak making the program run a little slower until the window has been closed. An improvement could be made where a slider bar could be used to alter the values of the object however this might not work as the values would have to be limited.

Improvements can be made to the camera class as when the selected object has been focused the user can move around it but the values are completely skewed making it swing round in random ways. Taking in the object position and moving the camera to that location with an offset works well but I felt for this feature it was nice to just focus at it from the camera position. Also there is a bug that when moving this camera the other camera lookat position is updated as well so when switching back, it causes the camera to be turned in a very strange way. Maybe by locking the values when the spacebar has been pressed will fix this.

On a good note however the features all work as intended, the highlight allows the user see what object is selected, the object selected can be focused on, each of the function for copy paste and delete work perfectly and the inspector functions as intended.