# Summary of Features

The features included in the project are as follows: A multiple object select/highlight system added to the mouse picking tool, as well as a deselect-all option. The ability to manipulate the position, rotation and scale of any selected objects. To avoid cluttering key bindings the current mode is selected from a window, and the selected mode is also reflected in the colour of highlighted objects. Camera enhancements include a snap-to-object feature and an arc-ball camera. The snap-to-object feature also works when selecting items from the item select window, making it easy to find objects. An extra window is also included which displays the information of a currently selected object. Once moved it is also possible to save all object’s statuses with the save button.

# Definition of Controls

## Mouse/Selection Controls

Ctrl + LMouse – Select Object (Adds more items to the selected objects list)

Ctrl + X – Deselect all objects

Ctrl + RMouse – Snap to the clicked object (Single object only)

Ctrl + N – Create a new default object

Ctrl + M – Select all objects (‘A’ interferes with camera and manipulation, ‘M’ could stand for ‘Many’ or ‘Multi’)

Ctrl + C – Copy selected objects

Ctrl + V – Paste previously selected objects

Backspace – Delete selected objects

## Camera Mode (Also active when ‘C’ is held)

W/A/S/D – Directional Controls

Q/E – Pan left/Pan right

R/F – Upwards/Downwards movement

Mouse Movement – Camera Rotation/Tilt

O (with single object selected) – Activate Arc-Ball camera

## Move/Rotate/Scale Mode

W/S – Increase/decrease x axis values

A/D – Increase/decrease z axis values

R/F – Increase/decrease y axis values

Hold Shift with any of these to edit the values faster

## Windows

Mode selection Window – Click the button for the mode you want, checkmark will be beside the current mode (default camera mode)

Object Data Window – Click ‘Update’ button to get most recent data for the selected object. Can only show one object’s data at a time.

Select Object Window – Same as default, except jumps the camera to the selected object, and the object is then highlighted.

# Feature Explanations

## Mouse Related

### Highlight Selected Objects

Selected objects are highlighted by creating another model which takes the existing values of the object and scales them up by 1.1 times. The wireframe of this model is then used to show which items are selected. The highlighted objects are stored in a vector of display objects just as the actual objects are, allowing for the number of highlighted objects to scale dynamically with the number of selected objects. The colour of the highlight is set depending on the current mode. White means Camera mode, Red means Move mode, Green means Rotate mode and Blue means Scale mode. The colour of the highlight is set in the render stage by using the built-in fog system to colour the highlights.

The highlight system is probably the most necessary and versatile of the features added, to the point where I almost forgot to write about it because it feels so baseline. It allows the user to see not only which objects they have selected, but the state that those objects are currently in.

### Multi-Select

Objects are selected by clicking them while holding ctrl. This highlights them, adding them to a vector list of highlighted objects. Any object that is highlighted is considered to be selected for the purposes of object manipulation. If an item is already selected and another is ctrl-clicked it will be added to the list, highlighted, and then will be manipulated if the correct mode is active.

Multi-select greatly speeds up the process of moving, rotating, scaling or otherwise editing large numbers of objects by reducing the number of times an action would have to be repeated.

### Clear Selection

Hitting ctrl + X will clear the list of selected objects. It is bound to this key to avoid deselecting items when clicking around the viewscreen, as this is something I personally find mildly irritating when using editors.

## Camera Related

Mouse movement is activated at any time by holding down the ‘C’ key. This is optional as it is more often useful to be able to pick out an object accurately than move the angle of camera view.

### Vertical Movement

Vertical movement in the camera is calculated by taking the cross product of the camera’s look direction. This means that vertical movement is relative to the angle of the camera, and as such is not always moving along the global y axis. This is useful as without it the verticality of edited levels is highly limited. The key bindings of R and F were chosen because of their usage in Guns of Icarus, where using them feels very natural in tandem with classis WASD movement keys.

The inputs for this movement are handled inside the camera class along with the WASD movements. This is done by passing the m\_InputCommands into the camera class to ensure that updates are only done when the camera mode has been selected, or ‘C’ is being held down.

### Snap to Object

The snap-to-object movement is controlled from within the camera class, however it is also able to be turned on from outside the class so it can be used with the list menu. The snap itself works by moving the camera to the location of the object and setting the lookAt point to the object. The camera is then moved away from the object to allow it to be observed without it filling the camera. The camera’s rotation/tilt will not be affected by the snap, meaning that if the user is looking up at a 45⁰ angle at the time of activating the snap they will then appear underneath and behind the object.

This is useful to have when used in combination with the selection window as it allows the user to jump to the location of any object at any location of the chunk without having to manually find it using the regular camera movement.

### Arc-Ball

The Arc-Ball is made by simply changing the lookAt point to the selected object. This only works if just a single item is selected. This means that the camera will remain focused on the object selected and allows for a full 360⁰ view of the object. This makes the W/S keys control the distance from the object, and A/D/R/F control the position of the ‘orbit’. There is currently an issue where if the user holds R or F for too long the camera will break. This could likely be fixed by capping the values associated, however this is not a huge issue.

The Arc-Ball feature is useful as it makes it much easier for a user to fully view an object from all angles.

## Manipulation of Objects

### Move/Rotate/Scale Objects

The main form of object manipulation are the 3 ways to interact with objects. Move, Rotate and Scale all have their own class to handle inputs, and the update for this class will only be called while in the appropriate mode. As mentioned in the controls section, the W/S keys manipulate the x axis, the A/D keys manipulate the z axis, and the R/F keys manipulate the y axis. In use, this means that moving an object acts like moving any 3rd person character with a WASD control scheme, with R/F allowing editing along the remaining axis. Selected objects are edited by first creating a list which points to the objects which are to be edited. This pointer list is then passed through to the appropriate update function and the changes are applied. After this the highlights are updated to match. If the shift key is held while manipulating an object the operation will be done faster.

Object manipulation is a critical part of the project, as it allows for more complex scenes to be created. The ability to operate on multiple objects are once is an excellent time saver as it means repetitive tasks become less so.

### New Object

Using Ctrl + N the user can add a new default object to the scene. This object is made in the same way that the display list is built, except with default file paths and values instead of reading values in from the scenegraph. Adding new objects is important as it means there is no real limit to the number of objects in the scene.

### Copy/Paste

Ctrl + C adds the currently selected objects to a vector list. This list is cleared before new items are added. Once the list is populated, the user can paste the items with Ctrl + V. This offsets the positions of the copied objects and gives them new IDs before adding them to the list of display objects, as well as the list of highlighted objects. This makes finding the new objects fairly easy as they are the highlighted ones. This is advantageous as it copies every detail of the copied object, including textures etc to reduce the amount of time spend making objects of a certain type as once one is created it is easy to create any number more.

### Delete

The list of display objects is checked from highest to lowest, with any matches in the highlight list being erased from the display objects list. The ID numbers of objects in the display list are then reset to avoid issues when selecting, and the highlighted objects list is cleared as when this is not done it can lead to crashes as non-existent objects are checked for their ID.

## Windows

All windows have been made so that they can be closed using the cross button without causing the program to crash upon reopening the window.

### Object Information Window

This window outputs the object ID, Xpos, Ypos, Zpos, Xrot, Yrot, Zrot, Xsca, Ysca, Zsca and Snap-to-ground value of the currently selected object. The only downside to the window is that it has to be manually updated using the update button, meaning that any movement done in real time on an object will not show up.

This is a useful feature as it allows for the user to see the item in more detail than just looking at the physical model. This could also be extended to show more details of the object if they were added.

### Mode Select Window

This window comprises of 4 buttons and 4 check boxes. Each button corresponds to a mode that the user can be in. The currently selected mode has a check mark next to it. In Camera mode the WASD and RF keys move the camera. In the other 3 modes the keys manipulate the selected objects. This is achieved by referencing an int in ToolMain which then sets an int in Game to let the renderer know what inputs should do, as well as what colour highlights should be used.

This feature saves on key bindings, as it would otherwise be very cluttered or convoluted to switch between modes. Alternative options would likely be to hold multiple keys at a time which is inconvenient. Alternatively, the modes could be bound to number keys, although this would be a similar solution that wouldn’t make the currently selected mode as obvious unless the user already know the colour scheme of the highlights.

### Object Selection Window

This is the object selection window that came with the template project. However, it has been upgraded so that upon selecting an object the camera will snap to that object’s location as well as highlighting the object. The snap only happens for one frame to avoid locking the camera in place, and to achieve this a bool is handled like the int in the Mode Select Window which ensures the camera is immediately free to move again. A list of the display objects is now passed in place of scene objects so that it is up to date even before a scene is saved.

These are minor improvements to the selector, however they definitely add to the ease of moving around the chunk, as being able to select any item and move to its location is much faster and more accurate than manual camera movement. It can also be immediately followed by pressing ‘O’ to turn on the Arc-Ball camera as only one item will be selected at this point.

## Miscellaneous

### Object Saving

When the save button is pressed a copy of the display objects from Game is passed to ToolMain. From here the freshly cleared database is filled with new objects, the values of which are set from the existing display objects. This allows any saved changes to be observed even after closing the project, an essential part of any tool.

### Select All Objects

Ctrl + M selects and highlights all objects in the scene. This is achieved by first clearing the list of highlighted objects, which is then filled with a new highlight object for each existing display object. The advantages of selecting all objects instantly are mainly time saving, as previously each item would have to have been selected individually which is highly inefficient.

# Conclusion

Overall, the project went fairly well in regards to the number of small improvements that have been made both for usability and world editing. The main downside is a lack of any large features, as this means the system as a whole remains rather generic. Were I to redo the project I would like to make a few small tweaks, such as making mode switches able to be done via number keys as well as the window. More mouse features would likely help the flow of the program as well, as the mouse is fairly under-utilised. A lot of time was wasted initially on getting MFC related features to work despite the majority of the grading being done based on adding/enhancing interactions with the world. This meant that I did not have as much time to polish the application’s features as much as I would have liked. Having said that, I would also have liked to do more of the manipulation through tool windows, but due to a focus on completing the essentials this was never fully realised. This can be seen in the Object View Window, which was initially supposed to take inputs too, except it had to be pushed aside to allow enough time to fully implement basic features such as Copy/Paste and Delete. Although the project went fairly well it lacks a centrepiece which would act as a proper demonstration piece, and as such leaves me a tad underwhelmed.

# References

The fog used to colour the highlights was from code was received from another student, who received it from Matthew Bett, the module leader. To clarify, this code is as follows:

m\_highlightObject.m\_model->UpdateEffects([&](IEffect\* effect)

{

auto fog = dynamic\_cast<IEffectFog\*>(effect);

if (fog)

{

fog->SetFogEnabled(true);

fog->SetFogStart(0);

fog->SetFogEnd(1);

fog->SetFogColor(Colors::Yellow);

}

});