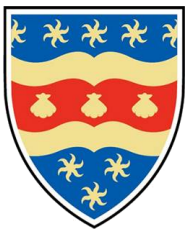


University Of Plymouth

School of Engineering,

Computing, and Mathematics

A Co-Operation Level Editor

Osbourne Laud Abraham Clark

10777267

BSc (Hons) Computer Science

Plymouth University

United Kingdom

$??^{th}$ April 2025

Acknowledgements

Abstract

This project see the successful development of a late stage prototype application that will make level creation for Co-Operation easier for users, without the users needing to know/ learn or manipulate YAML files directly. I first talk about the reasons for creating the application as well as defining the applications end goal, with a brief glimpse at the previous attempts made into the creation of this application. Followed by the permission requirements and gathering along with the legal, ethical, social and professional issues and aspects for the project. Then ”method of approach” outlines the functional and non-functional requirements needed for the application, as well as management tools used for the project, and the technologies needed for the successful development of the tool. Once all the background has been talked about i then talk about and show the implementation of the application, showing key parts of the applications code, with code snippets and images split between the appendix and inline with the section. After talking of the implementation, the evaluation and conclusion finish the document with references and appendix following.

Extra

Word Count: 7354
Code Link: https://github.com/Ozzy-King/comp3000

Contents

List of Figures

Listings

1 introduction

1.1 Background

this dissertation idea came about through a problem identified in my second academic year while developing a mod for the game Co-Operation created by MINDFEAST. the issue found was the tedious nature of creating levels. there are methods for designing levels, such as 2D layout on paper or excel, so basic level visualisation is possible. However once levels have been implemented, and have become big or start to contain a lot of objects they begin to become a mess, with attempts to quickly change a single objects becoming a massive effort. there are multiple method used for finding and changing object positioning:

- counting rows and columns in the game
- crawling through the file until you have found the correct object
- make continuous mental logs of where things have been placed

even with these methods, level designing and iteration becomes slow and tedious. My aim with this project is to create a visual, easy to use tool to be used along side Co-Operation making level creation more fluid.

1.2 Objectives

the aim of this product/application is to help facilitate the creation of levels for the game Co-Operation. the interface and controls should be easy to use and understand. the target audience are both the experienced and inexperienced players of co-operation who want to create levels for the game.due to the nature of level creation - with YAML being tedious in nature as well as including the requirements for moving around and tracking of strings and substrings - the main target users will be the inexperienced with little to no knowledge of these technologies. in reference to the inexperienced, the tool should allow for users that wouldn't normally create modified content for games to now have a chance. this opens up a previously inaccessible creative nature to the individuals as well as the ability to target the widest audience possible. This tool will also be a valuable with the experienced users who have already created mods, those starting to create mods or, have just started learning the modding process. Those who have already got knowledge of YAML can take advantage of this application, as instead of manual file modification, visual manipulation can be performed instead.

1.3 Deliverables

the deliverable will be a piece of software that can display a close 1:1 representation of the Co-Operation level, while facilitating a live tile manipulation solution to be able to edit the level directly without needing to modify the text file. the users will be able to export their created levels to their respective folders, as well as import existing level files so that levels not originally made using the tool can also be easily modified.

for object manipulation, features that are to be required are CRUD(Create, Read, Update, Delete) on objects with in the scene to have control over how the level looks. the tool should also be able to import objects in GLB format from the correct file location for use within the scene.

scene manipulation allows the users to zoom and pan the camera around the scene. for zooming the mouse wheel is used with defined minimum and maximum limits. Panning uses middle mouse click to function and its speed is defined by the current zoom level.

when importing and exporting of levels, it uses the YAML format layout described by the game documents. for the exporting requirements, the exported files must look the same in games as in editor as well as also be playable in the game. core level features are supported for importing, this includes the main features of 2d art objects (figure 14) as well as 3d art objects (figure 15) and the structure of object definition them selves (figure 16). the only feature that cant be supported without integration, of this tool into the game, would be ”mapObject” with values that aren’t ”custom” since these rely on built in objects that aren’t available (figure 17). if the object uses previous described objects it would use ”base” instead of ”mapObject”.

quality of life features that have been added is feedback to users to update them on the current state of the editor. a key quality of life feature is the object highlighting both when the user hovers their mouse over an object and when interacting with the object through clicking and dragging. the other quality of life aspect is for making the selecting of objects easier and which object is going to be placed next. this is done through middle mouse click for selecting objects, with the next object that will be placed is displayed through on screen text in the bottom left corner.

2 Legal, Social, Ethical and Professional Issues

2.1 protecting user supplied data

this application does not require personally identifiable data or any personal data. because of this, features such as security and storage are not need in the implementation. the application also doesn’t send usage data to external sources and so i wont

need to ask for consent or guarantee data privacy. this circumvents a list of possible issues that could arise relating to security, consent, and secure storage, as well as reduce the cost as there is no need to pay for third party hosting servers and cloud storage.

2.2 permission from MindFeast

there was a potential, when starting the project, that i could have stepped into a copy right infringement area which is a legal issue, as well as professional. without permission i would've be working on an application for a game that i don't legally own and copying aspect only found in their game such as the level layout method. professionally this would cause the potential issue for me to be seen as a bad actor meaning future employers potentially wouldn't trust me. MindFeast's vision for Co-Operation is to have the community make mods. This is shown through their extensive exposing of the underling base game mechanics though LUA, and YAML. while they want a community effort for mod creation using their modding API, the sort of mod/tool i wanted to make is stand-alone and so wont be using their API directly. to circumvent this issue i was required to gain permission from MindFeast as to allow me to continue with the project. i ended up messaging them on their discord server, gaining confirmation within the next couple days.

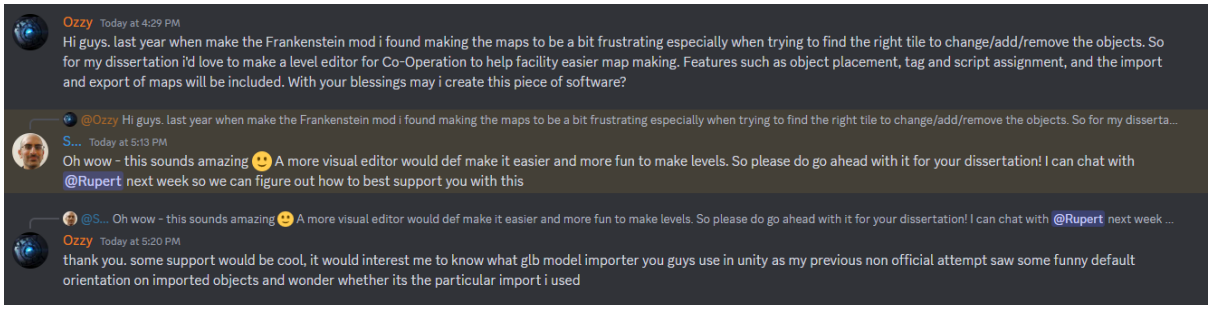


Figure 1: permission confirmation from Director of MINDFEAST

2.3 permission of previous group

for this application i needed an example mod for the testing of the tool. the need for the use of a mod is to help with the testing of the tools features. this included: importing and exporting of GLB objects, objecting manipulation as well as the importing and exporting of new and pre-existing level files. there was a possibility of asking MINDFEAST to use their levels for testing but because they use built in objects i opted to use another instead to test the implemented features as best as possible. the mod i used for the duration of this project was the frankenstien mod made in comp2003 in

second year. this was the easiest mod to use as i personally help work on it meaning on know what the mod is doing intimately as well as the file structures and the levels in the mod. i also have contact with the other members who also developed the mod and so i could easily send them messages asking if i had their permission to use their contribution. the reason i didn't create a new mod from scratch was because the application focuses on level creation as opposed to mod creation. however this application could speed up the completion of mods as it stream lines the laborious part of the mod creation.

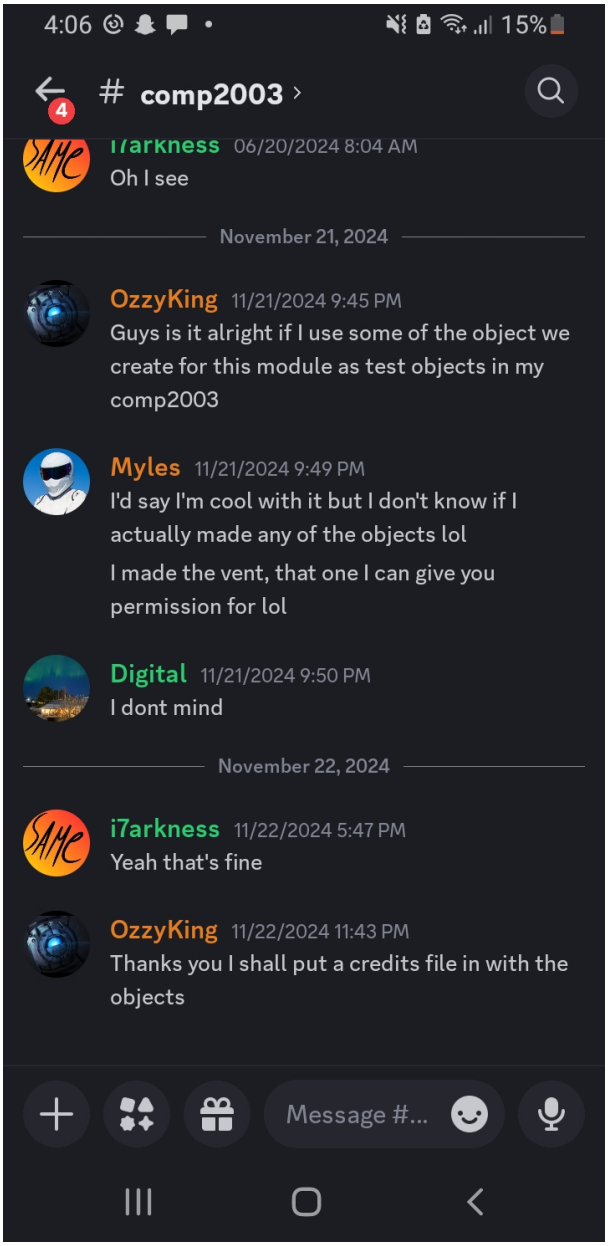


Figure 2: permission confirmation from frankenstein mod creators

3 Preparation

3.1 previous attempts

before making this project coursework, i had two other attempts at the creation of this application. the previous two varied quite a bit and the main aspect from both were brought over to the 3/current attempt which is this dissertations project.

the previous attempts were hacks. they were quickly made in an attempt to make the editing at that point easier. only one works, which happens to be the 2nd attempt, due to shortcuts i used when programming with win32.h .

3.1.1 1st attempt

the first attempt[˙**2024˙leveleditor**] used windows win32 header to create and show information to the users. it was entirely text based, only using buttons and input boxes, with no advance visual features (figure 3). the point of its creation was to quickly create an application that could be used by the team i was in for the second years comp2003 to create levels faster and easier. understandably it was never used as it was made after most of the levels had been developed and did see a vast amount of bugs due to its two day creation. it did however tackle the issue of cell identification. by using images of all letter combinations from AA to ZZ, the application placed those images in the cells (figure 4). within the game view the cells could be easily identified and then modified through the tool. the only issues with this design was that it required at least 2 windows open, and one for each cell that was getting edited, meaning the screen could become a mess of windows if multiple cells where getting edited at a time.

3.1.2 2nd attempt

after the first attempt, while it was not used, testers and viewers gave feed back and with it came the second variation[**ozzyking˙2024˙github**] (figure 5) (figure 6). the second attempt used a 2d pixel based engine called ”olcPixelGAmeEngine”. it worked with the idea that instead of the cells displaying all of its current objects, it would show a colour based on if a specific item was in the cell. this was done by creating layers so each item had it own map. this attempt did not see implementation of easily identifiable cells, instead its aim was to make level editing quicker. this was done by only needing the user to find the object in a searchable list and clicking the cell. this gave feedback though colouring the cell, telling the user the item was now in that cell. identification could have been added through the displaying of attempt 1s images, but on this attempt i was also wanting little user setup so it was more user friendly instead of requiring images to be copied into the working folder of each mod.

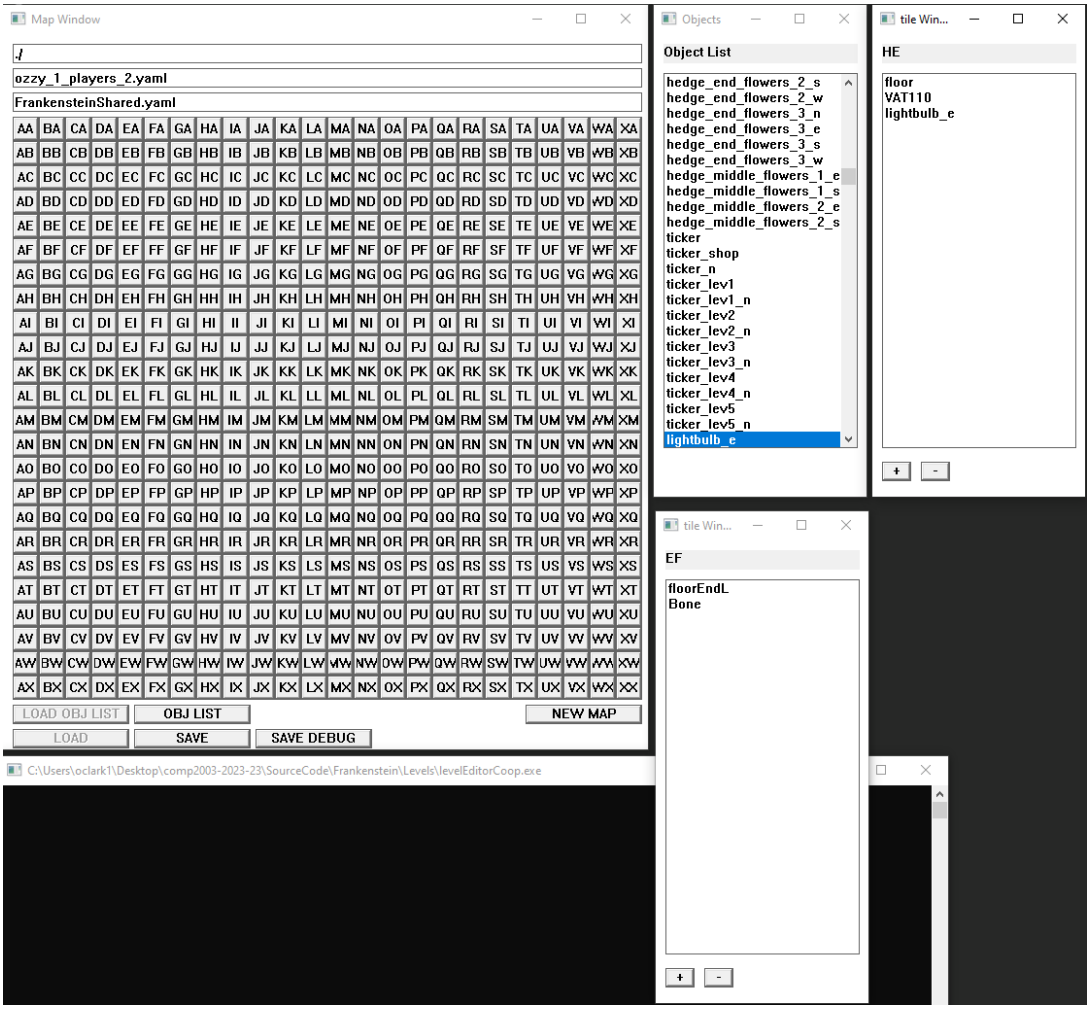


Figure 3: first editor made

3.1.3 current attempt

the third attempt is this dissertation. its aim is to have easy identifiable cells as well as make them easily editable, merging both major features from the previous attempts into one. it was understood early on, back in attempt 1, that a completely 3D solution would be best, but due to time restraint on other project this was not feasible at the time.

4 Method Of Approach

4.1 agile

through out the development of the project, agile was the outlined method of approach that was to be used for planning and completing of the dissertation project. agile first sees the creation of sprints which can take the form of a gant chart. these sprints are

Figure 4: first editors exported level file in game

then worked through one by one, with scrum/ stand-up meetings during/ at the end of each sprint. in these meetings, progress is discussed as well as general plans for the upcoming sprint. once each sprint has been completed, in term of time as opposed to tasks, future tasks are revise and either added or removed from subsequent sprints. this process aims to make sure you are completing the most amount of work that is possible by the individual. the choice of addition or removing of tasks is based on if all the tasks where complete, meaning if not all where completed then you mostly over assigned tasks and need to reduce for the next sprint.

4.2 trello

once each off the sprints had been laid out and created on the gant chart (figure 7), the sprints and the progress for each sprint is to be documented on trello. trello is used as a way to track the progress of build an application. it shows what has been implemented, what has yet to be implemented, along with possible future additions that aren't set in stone(possible additional features). it is well suited to team environments as every one can see the apps progress al well as see clear outlines of what needs to be done and what has been done. through out the creation of the application my use of trello was limited, with a lot of the cards being added within one day. how ever i found the addition of cards and the catching up of my trello with the project was beneficial as it showed me what i had done and what i needed to done to finish. it helped keep my mind focused instead of jumping all over the place loosing track of what i have started or added to.

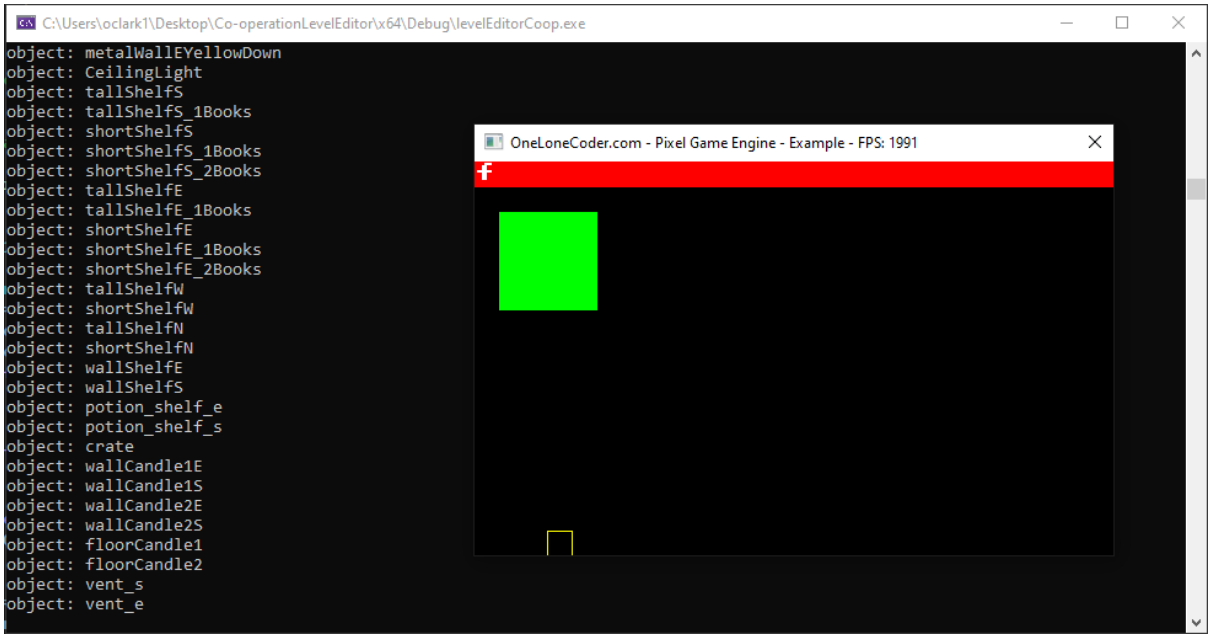


Figure 5: second attempt at making an editor

4.3 technologies

4.3.1 unity

to create this project i utilised unity for its easy to use graphics manipulation methods. From the beginning it was clear a 3D solution was required for the project as it would allow for easy understanding of visuals, easy cell identification, as well as allow for more intuitive use of the software. If 2D was chosen it would not have given the same feedback methods or ease of usability to the user. By going 3D i could give the user a closer idea to what the levels will look like in game.

4.3.2 C#

i was required to code within C# as this is unity primary coding language . this was beneficial as i am well versed in it and have used it before(both unity and C#) and so didn't have a learning curve of a new language.

4.3.3 Libraries

importing GLB objects

i have chosen to use UnityGLTF[**siccity˙2021˙github**] to import GLB object, which are compatible with Co-Operation, into unity at runtime. this also allows for hot reloading object definitions without the need of re compilation of the entire unity project meaning further object definition defining could be performed. the library was chosen over oth-

Figure 6: second attempt at making an editor

		Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9	Sprint 10	Sprint 11	Sprint 12	Sprint 13
		28-Oct-24	11-Nov-24	25-Nov-24	09-Dec-24	23-Dec-24	06-Jan-25	20-Jan-25	03-Feb-25	17-Feb-25	02-Mar-25	16-Mar-25	30-Mar-25	13-Apr-25
added tasks	TASK NAMES	10-Nov-24	24-Nov-24	08-Dec-24	22-Dec-24	05-Jan-25	19-Jan-25	02-Feb-25	16-Feb-25	01-Mar-25	15-Mar-25	29-Mar-25	12-Apr-25	25-Apr-25
	movability around 3d scene(map panning, map zooming)													
	object Crud (creation deletetion)													
	object Crud (viewing, updateing)													
	import yaml													
	export yaml													
	demonstrate cooperation													
	create level using level editor													
	implement quality of life features													
fix bugs														
	user testing													
	poster and description													
	buffer													
	buffer													

Figure 7: my created sprint plan

ers as i have used the library before which also allowed for easier setup in code and in project.

parsing YAML

YamlDotNet[**aubry˙2024˙aaubryyamldotnet**] is used be able to import and export file in YAML format without having to write a parser from scratch. This is a library built for C# and so worked easily with unity. this made importing and export much easier as there was no need to manually parse the file. the reason for choosing this library is similar to UnityGLTF, as I've previously looked into the use of this library and made implementation easier from the start.

4.3.4 Co-Operation game

Co-Operation will be used to to test if the level output file of the editor is valid to use in game and if the view within the editor it similar/ the same. this is very important to use in order to create this editor as its aim is to give real time visualisation of the level when creating and updating it

access was gained to this during comp2003, where the client, MINDFEAST, asked for a mod to be created for their game Co-Operation.

4.3.5 GitHub

GitHub has been used for version control and storage of the project files and code. currently only two branches have been made and have now been merged, with the later branch being made more towards the end of development. while version control wasn't heavily used, it was extensively used for transporting the code.

4.4 functional requirements

for the functional requirements for the project to be reached, the created application must be able to accurately visualise the imported level so that the user can see and easily understand what the level will look like in game. the required user controls that will be added will allow the user control of the camera for moving around the scene, the importing and exporting of level file which support the core aspect of the level creation with the added ability for importing of already created levels. CRUD is also required so that object can be placed within the scene as well as give the ability to move and delete these object, both the ones already loading from the level and those placed down by the user. with these feature added the application will be useable by users, allowing for importing, creating, updating and exporting of levels for the use of playing in Co-Operation

4.5 non-functional requirements

non-functional requirements will allow for quality of life feature to be added. the features include:

- the changing of panning speed based upon the current zoom distance of the camera. this is so that the camera speed stays within a controllable level, stopping the control from potentially loosing the player by unpredictable sporadic movement

- on screen hints as to the current state of selected and next placements, giving feed back to the player as to the what is going on. with this users wont feel as lost or blind when understanding which object they are currently placing or which object they have already placed.
- quick middle mouse click to select object for next placement, allowing users to quickly select/clone objects for use in different places around the scene.

5 implementation

5.1 UML

5.1.1 classes

with in the diagram shown (figure 8) it shows the interaction between classes within the application. the direction of the arrow indicates which class interacts with another class with the added 2 way interaction where two classes contain instances of each other. An example of this is ”MouseControls” containing an instance of ”GlobalResources” as to be able to use some values that the class stores. in the case of the ”MouseControls” and ”GlobalResources” interaction, values like editor state checks are used, as well as some object related resources such as materials for hovering and selecting and institution of new objects created by the user. this style of object coupling

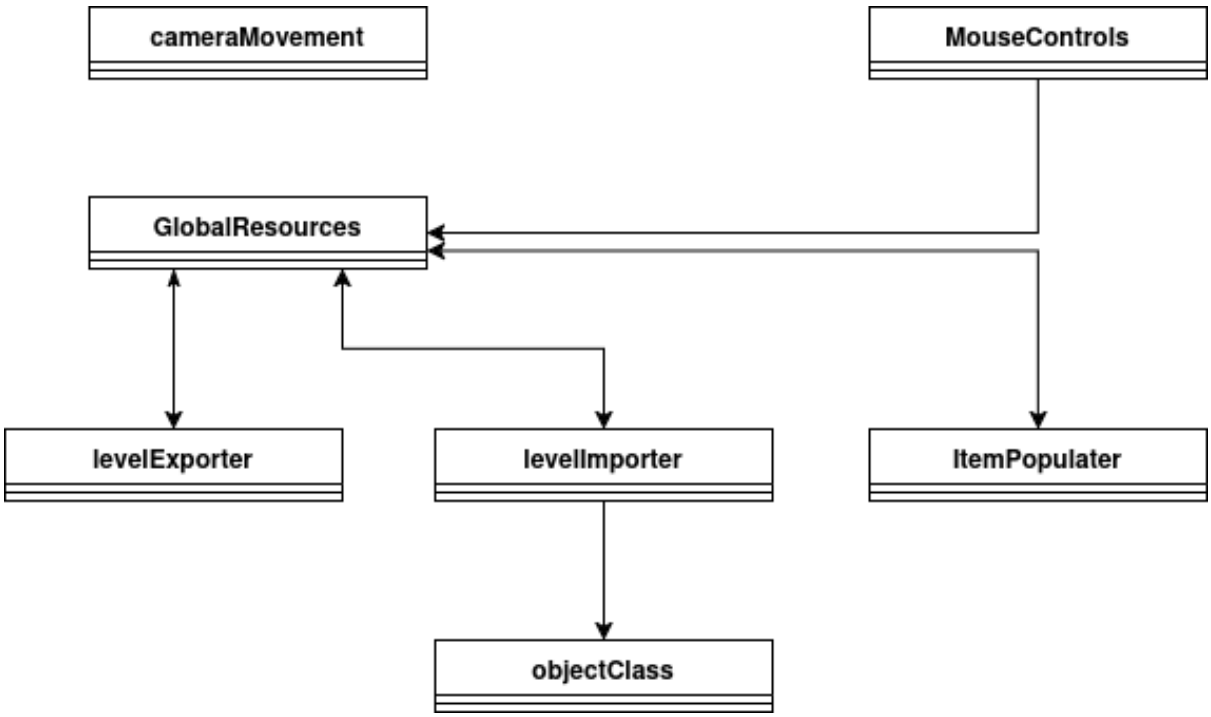


Figure 8: interaction between classes

interaction by use of referencing and not allowing child classes to communicate is called mediator design pattern. the purpose of this is to stop tight class coupling. by reducing how strict the coupling is, maintenance, refactoring and scaling become easier as once the business logic is create (the connection between the classes) only computation needs to change / be optimised. If new communication needs are needed between classes, they are added to the common mediator class which the child classes will use to communicate with each other.

5.1.2 control flow

(figure 9) shows the the basic flow initiation of the program. the main start function is found with in the ”globalResources” file, making the global resources the main controller for the entire program. it will communicate a bit with level loader to get add the required resources and make sure it is possible to actually load the level. the level loading function return a success code that global resources handles. once the file have been loaded and are in the correct variables, two coroutines(”thread-like”) get started, ”formLevelObjects” and ”PopuateObjectList”. these are the two main functions that create the level within the scene to allow for editing to start, with the ”formLevelObjects” creating the actual level within the scene and ”PopulateObjectList” displaying and controlling the list of available objects to the user.

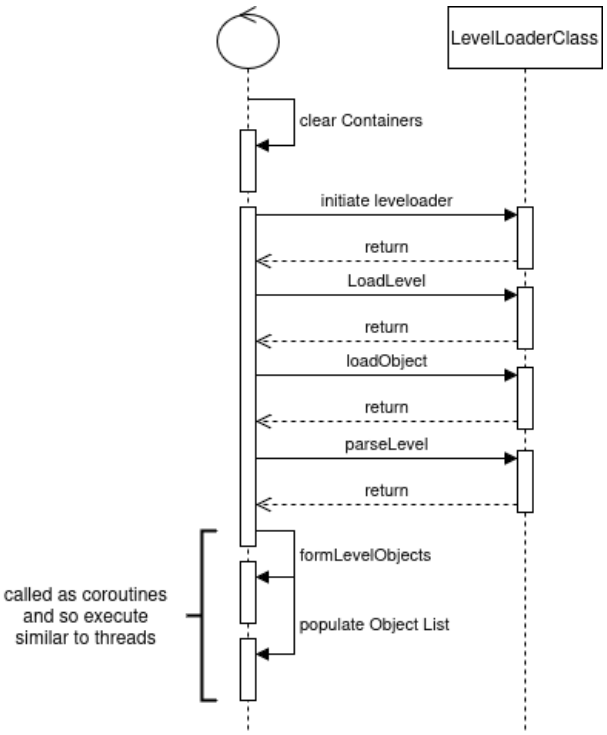


Figure 9: the basic flow of control when levels are being loaded

(figure 10) is the basic representation of the caching system. this system was put in

place to curb the long loading needed for GLB importing. while the diagram describes the system for ”Obj”(objects), a similar system is also working for the caching of images as well. large images and objects can take some time to load and so by caching them, repeated use of the same object and images can happen without significant lag occurring. the system is as follows: the resource, object or image, is first checked for in a dictionary, if the object is found it is retuned. if not it will check for the file, if it cant be found it will return a placeholder object. if the file does exist, the file will be loaded out of sight of the player and will then be returned to be used by the calling script. since just the GLB object and images are cached instead of the entire defined object within the folder, reloading is also very fast. this is due to not needing to reimport all the relevant objects in order to get the new transformations that are applied when the object are getting place within the scene.

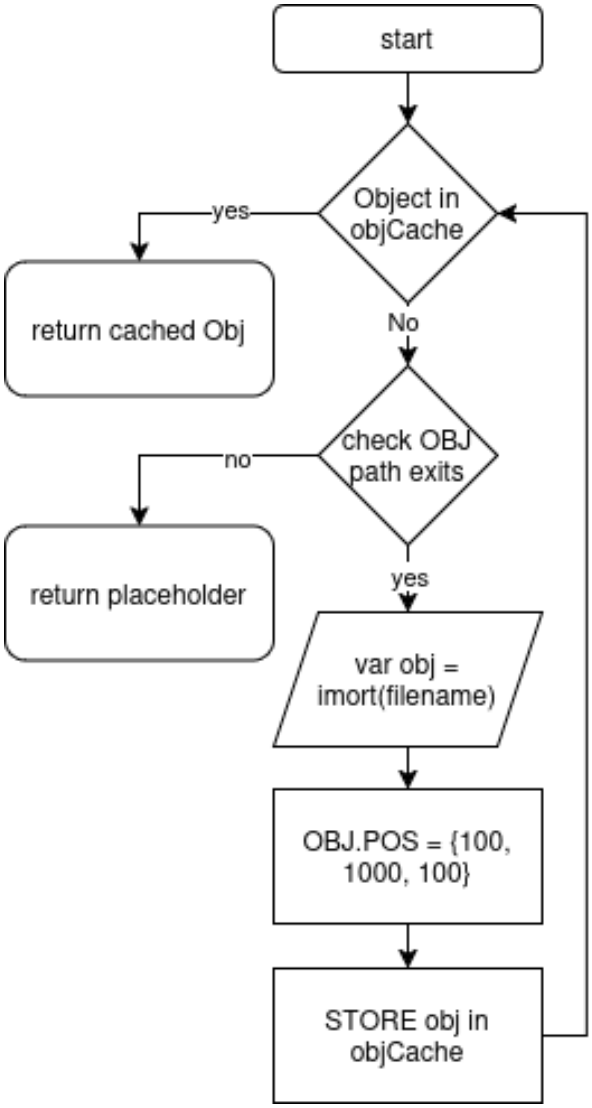


Figure 10: steps used for cacheing system

this last figure (figure 11) shows the use flow of the application. it shows how to

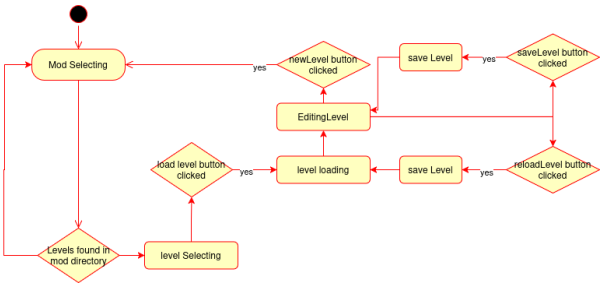


Figure 11: the use flow of the application

get to each screen and what happens after the completion of certain actions. the user has the ability to get to each state and use the features found in each one. there are clear separation between each state, for example if you are still selecting a mod all the buttons except the ”load” button will be greyed out. This reduces the chance that the user breaks the flow of the program. once the user has gotten past the level selection, the load button will turn to ”load new” which will completely clean the caching as to allow a new level file to be loaded in. this is the only way to get back to the beginning once the user started editing a file, the rest of the actions keep the user on the main editing state, with sub states that only happen momentarily.

there is no exit state put in place. the method for users to exit the application is to save their progress if they want to keep what they have created and proceed to close the application. there are ”OnApplicationQuit” methods found within the global resources file which clean the cache of any kept object and images.

5.2 YAML

5.2.1 importing

while i utilized a library for parsing and loading information from a YAML file, i was still required to understand and layout a class that represented the YAML file for with in the application. a lot of the needed information could be described with primitives such as: int, strings and float. some, however, where a little more complex and required the creation of additional classes to properly represent these structures (listing 15). one peculiar feature found in the YAML level file was anonymous objects, which are defined in the maps cell list as opposed to the object definition files. to correct this when opening the file, the cell definitions get loaded into a list of object types. if a anonymous object is found, it will be saved to the current levels object definitions to be used within the game. this was added so that level created without the editor that used anonymous objects could still be opened using the editor.

```yaml
gridObjects:
## demoing the anonymous object int the curly brackets
  AA:
    - objectOne
    - mapObject: "Custom" ##this is the custom object
      art3d:
        - model: "Art/test/placeHold.glb"
          pos:
            x: 1.5
            y: 0
            z: 0
          scale:
            x: 1
            y: 1
            z: 1
          rot:
            y: 135
## what normal grid cell definition looks like.
  BB:
    - objectOne
    - objectTwo
```

Listing 1: demonstartion of anaymous object

5.2.2 exporting

when each object is placed into the scene, it also gets added to a long list of objects that are currently in scene. this list is used when export/saving the level back to the file. it still uses the LevelFile layout class as before, but will manually populate the Grid as well as the grid cell definitions. the map wont start from 0,0 and so the maps dimensions must first be found and calculated. once the bounds have been found the grid is constructed before the creation of the cells definitions. the grid consists of 2 identifying letters from A-Z, this allows for 676 unique cells possible(listing 2). the limit can be adjusted in the future where each cell is identified by an arbitrary number of letters. once the grid has been generated the cells are then generated with the correct objects. this involves looping through each cell, and then looping through each object in scene to find the object that belongs in this cell (listing 3). this is a O(n*m) time complexity, where n is the number of objects and m is the number of cells. there are faster methods such as a single pass with multiple buckets for each cell so the objects

are read and ordered at the same time and then never accessed again. this would be faster especially for larger levels, but does also come with the cost of more memory allocated at one time. for the current use extent, the current method in use is fine, but if needed in the future i will implement the method mentioned.

```
for (int y = 0; y < levelHeight+1; y++) {
  for (int x = 0; x < levelWidth+1; x++) {
    grid += ""+(char)(x + 65) + (char)(y + 65) + (x != levelWidth ? ","
    : "");
  }
  grid += "\n";
}
```

Listing 2: grid generation

```
Dictionary<string, List<object>> gridObjects = new Dictionary<string,
   List<object>>();
for (int y = 0; y < CurrentLevelMapped.GetLength(0); y++) {
  for (int x = 0; x < CurrentLevelMapped.GetLength(1); x++) {
    List<object> cell = new List<object>();
    string gridId = "" + (char)(x + 65) + (char)(y + 65);
    if (CurrentLevelMapped[y, x] != null) {
      foreach (GameObject obj in CurrentLevelMapped[y, x]) {
        cell.Add(obj.GetComponent<ObjectAttributes>().objectName);
      }
    }
    gridObjects.Add(gridId, cell);
  }
}
```

Listing 3: code used for cell forming

5.3 camera placement

the placement of the camera is initially at vec3(0, 0, 0), rotated with vec3(0,-45,0). this is the initial setup as to allow for the correct values to be captured before changing the camera to true isometric view (with perspective not orthographic). the values which is saved before rotating the camera 35.264 degrees around x (point it down) is the forward direction vector. the camera doesn't move from this rotation, and so the forward vector can stay the same for the life time of the application. the other values captured is the forward vector after the x rotation, the right vector and the initial mouse position. the right vector, first forward vector, and mouse position is used for panning around the map, while the second forward vector after the x rotation is for the zooming functionality.

5.4 controls

5.4.1 mouse Iteration with the scene

when the users are moving their mouse around the screen, custom materials get added to object which the mouse is over so that the user can see the currently hovered over object. the method for object detection is collision triggers with ray casts. the cast will only travel till $Y = 0$ meaning there isn't a need for a large hardcoded distance and redundant checks past $Y = 0$. the reason for the method of object detection is to have accurate screen to world space conversion as well as reduce redundant checks but also allow for required checks at any distance (listing 4).

first the ray direction is gotten using the ViewportPointToRay(). then using $camera.y/ray.y$, the number of steps is found for the ray to get to $Y = 0$. a temporary vector is created to hold the z and x, which are based on the rays position when its $Y = 0$, in order to hold the the second position for distance calculation. once all the required position vectors are found the distance is calculated and used in Raycast() to specify how far the ray travels. one of the potential issues to arise from this code is division by 0 if the user moves the mouse perfectly perpendicular to the y axis as $ray.y$ will equal 0. however due to the static orientation of the camera the required mouse movement to break the raycast should never occur. in the unreachable case where the mouse is pointing up into positive y, the distance will be calculated as if the camera is pointing down Y due to the absolute function used.

```
1  RaycastHit rayHit;
2  //Ray ray = cam.ScreenPointToRay(Input.mousePosition);
3  Ray ray = cam.ViewportPointToRay(cam.ScreenToViewportPoint(I
       mousePosition));
4
5  //find point at which ray hits y = 0
6  float objToYzero = Mathf.Abs(gameObject.transform.position.y
       direction.y); //how many steps to get from objects positi
        (camera is always looging down)
7  Vector3 temp = (ray.direction * objToYzero);//could move obj
       down to y= 0 with right transofrm on x and z
8  float distance = Mathf.Sqrt(Mathf.Pow(temp.x, 2) + Mathf.Pow
       .y, 2) + Mathf.Pow(temp.z, 2));
9
10 Vector3 HitWorldPosition = ray.origin + (ray.direction * obj
11 HitWorldPosition = new Vector3((Mathf.FloorToInt(HitWorldPos
       2) * 2, 0, (Mathf.FloorToInt(HitWorldPosition.z) / 2) * 2
12
13 //cast ray to y = 0
14 bool didHit = Physics.Raycast(ray, out rayHit, distance);
```

Listing 4: calculate distance for ray to travel

5.4.2 Middle mouse click

middle mouse clicking facilitates the panning and zooming movement for the camera, as well as the cloning functionality to place more object of the same type. the panning and zooming was quite simple to implement. the panning method first takes the mouse movement delta from the previous frame and the current one. the movement change is then multipied by the respective direction transform and is then divided by the movement speed, finally being applied the to the current cameras position (listing 5). the zooming feature uses the mouse wheel when scrolling, and adds or subtract from the current zoom based on the direction of wheel scroll. when the zoom is changed, the panning speed is also change with a further away zoom being slower that a closer zoom. this makes sure the speed doesn't go negative or too fast, meaning limits are put in place so that there is a lower bound(currently 0, which is the camera load in position) and the upper bound(which is the lower bound +30 zoom steps)(listing 6).

```
Vector2 currentMouse = new Vector2(Input.mousePosition.x, Input.
    mousePosition.y);
 Vector3 difference = mouseScreenPos - currentMouse;
 gameObject.transform.localPosition += right * difference.x /
    panningSpeed;
 gameObject.transform.localPosition += forward * difference.y /
    panningSpeed;
```

Listing 5: Camera movement code

```
bool valid = false;
                if (Input.mouseScrollDelta.y > 0 && currentScrollDist -
    2 >= scrollDistMin) {
                    currentScrollDist -= 1;
                    valid = true;
                }
                if (Input.mouseScrollDelta.y < 0 && currentScrollDist +
    2 <= scrollDistMax)
                {
                    currentScrollDist += 1;
                    valid = true;
                }

                if (valid) {
                    gameObject.transform.localPosition += up * Input.
    mouseScrollDelta.y;
                }
                panningSpeed = maxPanningSpeed / scrollDistMax * (
    scrollDistMax - currentScrollDist); //update panning speed based on
    zoom,
```

Listing 6: Camer zooming code

the cloning functionality comes in handy when the user wants to copy an object to another location. the process of decide whether the user is panning or copying through the middle mouse click, is by checking the mouse position change between button down the button up. if the change is less than or equal to a change of 0.5 then it will count as a copy(listing 7). this has been implemented like this so that the user wont accidental clone a object while panning around the level, even if its a small corrective pan.

```
//if middle mouse button is clicked
if (Input.GetMouseButtonDown(2)) {
  beginDrag = newMouse;
}
else if (Input.GetMouseButtonUp(2)) {
  endDrag = newMouse;
    if (Vector3.Distance(beginDrag, endDrag) <= 0.5f) {
      globalResources.CurrentObjectSelectID = lastHoverObj.name;
  }
 }
```

Listing 7: Copy Object

5.4.3 right mouse click

right and left click also have functionality attached to them. right click has a simpler feature and was recommended during an impromptu testing session i did with a class colleague. when they were selecting and placing items down, they wanted to delete items. at this point i was instructing them on the controls as there was no read me. before they asked me for the controls, they instinctually right clicked to remove the object. this sort of control is similar to the control found in the game minecraft as its a secondary control for manipulating the same thing, which is also where the middle mouse click shortcut came from. before the delete object control was right click, it was left click and hold on the object while pressing DELETE button on the key board. the process of finding and implementing this new control is beneficial as it showed me there was an easier control interface for deleting as well as showing the value of knowledge gained through testing, making even impromptu testing quite valuable. this change also moves most of the controls to the mouse. because the controls are all in one place they are easier to use, meaning the user doesn't need to remember lots of keyboard shortcuts, or look down to find the right key to click.

the code that is used (listing 8), has changed. before it was a lot more clunky. first left mouse click needed to be checked for and then a key down action on the letter E on the keyboard. now after changes, if no other action is triggered the right mouse click it checked and can easily and quickly delete the object.

```
//if right mouse buttons is clicked
if (Input.GetMouseButtonDown(1)) {
  globalResources.CurrentLevel.Remove(lastHoverObj);
  Destroy(lastHoverObj);
  lastHoverObj = null;
  removeHoverText();
}
```

Listing 8: delete Object with right click

5.4.4 left mouse click

left click is the last of the main controls with the rest being combination shortcuts that attach to the left click action. the left click both allows the movement of objects already found within the scene as well as the placement of objects with in the scene. both controls attached to the left click have stayed the same through out development.

hovering over an object while left clicking and holding keeps the object snapped to the cursor and the grid. this control is for moving the object around. while hovering over and moving objects, the object name will float above the object at a set point. this is to show the user the name of the object so that they know which object was place. this is needed due to multiple object potentially having the same visuals. the object will also be highlighted while the being dragged around, indicating to the user which object was picked up and if its the correct one. listing 9 is the code used for both the object placing and object hovering. every loop if E is held down it will move which ever object is currently selected, which will either be null or the currently placing object, to the mouses current location. once the object is placed with left mouse click, the currently selected object variable gets set to a duplicate object leaving the previous one in place.

the other control attached to the left click is the placement function. this is accessed while the keyboard key E is held down. while in this mode left clicking will place the object into the scene.

```
if (Input.GetKey(KeyCode.E))
{
  placeing = true;
  //if its the first time pressing e
  if (Input.GetKeyDown(KeyCode.E))
    {
    removeMaterial(lastHoverObj, globalResources._hoverObj);
    removeMaterial(lastHoverObj, globalResources._selectrObj);

    setLastHoverObj(globalResources.createObject(globalResources.
   CurrentObjectSelectID));
    addMaterial(lastHoverObj, globalResources._hoverObj);

    removeHoverText();
  }
  //move object to mouse position
  lastHoverObj.transform.position = HitWorldPosition;
  //if clicked place objectand create a new one to move to mouse
   position
  if (Input.GetMouseButtonDown(0)) {
    removeMaterial(lastHoverObj, globalResources._hoverObj);
    removeMaterial(lastHoverObj, globalResources._selectrObj);
    globalResources.CurrentLevel.Add(lastHoverObj);

    setLastHoverObj(globalResources.createObject(globalResources.
   CurrentObjectSelectID));
    addMaterial(lastHoverObj, globalResources._hoverObj);
  }
 }
```

Listing 9: object placement code with highlighting

5.5 Reverse Engineering of Level Layout

during the course of the applications creation i needed to reverse engineer the placement algorithms for the objects defined in the object definitions. each object can contain 3d art objects in GLB format, 2d art in png format and/or inbuilt objects that are compiled into Co-Operations base game. while developing i did not have access to these inbuilt objects, and being illegal to rip the objects from the actual game, i chose to instead represent these missing object with a placeholder object. the placeholder is a plain white capsule which is also used for objects that have no artwork attached. the reverse engineering was required for accurate level representation. Under UK law, specifically Section 50B(1) and (2) of the Copyright of Designs and Patents Act 1988 (CDPA), decompiling a program is permitted if it is done to create an independent

program that can work with the original program.

the method used for reverse engineering was passive observation/ visual analysis. this does not require decompiling the original application, instead only viewing of the game occurred with with period code tweaking to properly represent the level. once the algorithm for placement was created, it ended up being a single function that takes in a ObjectClass object. it first constructs the 3d art aspect and then the 2d art aspect of the defined object (code 14).

5.5.1 object placement

one of the main aspect while reverse engineering was the object placement within the scene, not necessarily the art placement in relation to the object. this issue came about due to my initial placement of the camera. within Co-Operation, +z is going from top left to bottom right and +x is going from top right to bottom left. this was a small issue as the camera within the scene was already facing the positive direction of both x and z axises. this meant i needed to reverse the object positioning in the map to the positioning within the scene. (figure 12) shows how the axies differ between the game and level format, and with in the applications scene.

code 10 also shows what steps are needed when setting the position to get the correct scene placement. the correct placement was achieved through simply reversing the x and z of the transforms with the -(minus) operator.

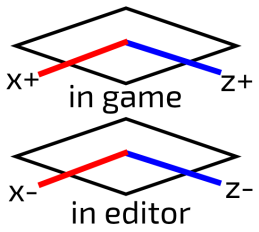


Figure 12: Demonstration of axis orientation

```
1 Temp.transform.position += new Vector3(-objsArt.pos.x, objsArt.pos.y, -
    objsArt.pos.z);//position offset
```

Listing 10: C# code for correcting object placement from file to scene

as well as the objects placment on in the scenes grid, the placement of the models and images within the objects where also slightly affect by this. this was again caused by again for the same reason as before; however due to solving the previous issue through the negation of the 2 different axises, this issue did not take as long to solve.

5.5.2 object orientation

object orientation was also a little peculiar when getting the right rotation on the object. the 3d object had an initial rotation of 90 degrees, while the 2d art work has the initial rotation of -90 degrees. once each of these initial rotations were found, it was then the issue of figuring out the required degrees of rotation for north, east, south, and west. through trial and error, the correct order of directions was figured out for the implemented enumerator. the enumerator was created so that each value could just be multiplied by 90 to get the correct rotation degree. once the direction rotation is correctly applied, the final rotation transformations could be added. this was just simply rotating around the respective axis by the specified degrees.

a behaviour that was found was that base objects defined in a object wont use their own rotation, but instead inherits from the object using them as base. this required an extra argument/function to be added to be able to handle this functionality correctly.

```
1    enum _dir {
2        east = 3, north = 2, west = 1, south = 0
3    }
4    public float DirToAngle() {
5        if (dir == "north") { return (float)_dir.north * 90; }
6        else if (dir == "east") { return (float)_dir.east * 90; }
7        else if (dir == "south") { return (float)_dir.south * 90; }
8        else{ return (float)_dir.west * 90; }
9    }
```

Listing 11: direction definition and conversion label

5.6 added features

5.6.1 cacheing for faster loading

it was found through testing that when placing new objects there was a slight lag spike. this was identified as a resource loading issue as it only occurred when placing heavily detailed objects. once the issue was identified it was remedied though a cacheing system that uses C# dictionary class to store already created object.

the method is as follows. when an object is about to be created, it is first checked to see if it has already been loaded in using the files name as a look up reference in the dictionary. if it hasn't, it get imported into the scene through the GLFT importer library i am using. once in it is moved to an off screen location. this is so it cant be seen by the user, and is then copied to be placed in the cell where the mouse pointer is located.

this method has been done for GLFT object as well as the images as both caused cause lag spikes each time they were getting imported into the scene. by doing this,

it considerably improved the speed meaning levels formed a lot quicker and is almost instant when reloading them.

5.6.2 linux compatablility

Cooperation Level Editor started development on windows as that was the operating system used be mostof the computer i had access to. how ever i at ome point was limitited to justlinux and could not access a windows computer. due to this i was required o make linux compatible with the editor. at first i touhg it must be as easy as getting the porject onto linux and compiling it on the linux os. this wasnot the case and i proccedded to research the potential issues stoppon the porject from running. when the project was first ran, there where no error messages and at somepoint when running the code there was a silient error. i found out the issues was the directory seperator wich was ”/” on linux. however this was handy as windows also can use ”/” and i had already been useing it through out my code. the only issue was in the ”populateLevelDropDown()” in GlobalResources.cs. this issues was remided with Path.DirectorySeparatorChar as this was the directory seperator char for the spesifc operating system.

```
1    public void populateLevelDropDown()
2    {
3        levelDropdown.ClearOptions();
4        LoadButton.GetComponent<Button>().interactable = false;
5
6        string workingDir = inputFilePath.GetComponent<TMP_InputField
     >().text;
7        if (!Directory.Exists(workingDir)) { return; }
8
9        LoadButton.GetComponent<Button>().interactable = true;
10       List<string> filenames = new List<string>(Directory.GetFiles(
     workingDir + levelDir, "*.yaml"));
11       for (int i = 0; i < filenames.Count; i++) {
12           filenames[i] = filenames[i].Split('/')[^1];
13       }
14       levelDropdown.AddOptions(filenames);
15   }
```

Listing 12: before '\\' changed to'/'

```
1    public void populateLevelDropDown()
2    {
3        levelDropdown.ClearOptions();
4        LoadButton.GetComponent<Button>().interactable = false;
5
6        string workingDir = inputFilePath.GetComponent<TMP_InputField
     >().text;
7        if (!Directory.Exists(workingDir)) { return; }
8
9        LoadButton.GetComponent<Button>().interactable = true;
10       List<string> filenames = new List<string>(Directory.GetFiles(
     workingDir + levelDir, "*.yaml"));
11       for (int i = 0; i < filenames.Count; i++) {
12           filenames[i] = filenames[i].Split('/')[^1];
13       }
14       levelDropdown.AddOptions(filenames);
15   }
```

Listing 13: after '\\' changed to '/'

6 Testing

6.1 smoke testing

through out the development of the application i did manual smoke testing. these are simple tests that are done to check the basic functionality of the application. they are

performed so that the implementer knows the created feature is initially working and so further unit tests can be written to test all the required edge tests.

this is a list of basic features tested with smoke testing. this is only for the basic functionality and so dosnt cover edge cases for particular functions. only the expected output is tested for when using the correct controls, and so doesn't take into account potentially broken control sequences.

expected	result
Objects when hovered over are highlighted white	Objects get highlighted white
Objects when hovered and clicked highlight green	Objects get highlighted green
Objects, when hovered, clicked and moved get moved around the scene	Objects get moved around the scene
Moved object are changed in the save file	Objects are reflected in the save file
When key "E" is held down, the placing object is shown	Object gets shown
While placeing, the placing object follows the mouse	The placing object follows mouse
When left clicking while holding "E", the object is placed in the scene	Object gets placed within the scene
Objects when right clicked are removed	The object gets removed from scene
Objects when removed do not show up in the saved level	Objects placed get removed from file
Objects when middle mouse clicked are selected to be the next object to be placed	The objects become the next placing object when middle clicked
When placing an object after middle clicking an object, the middle clicked object will be the placing object	The placing object becomes the previously middle clicked object
saved levels will look similar in game as in editor	saved levels look similar in game as in editor
loaded levels will look similar in editor as in game	loaded levels look similar in editor as in game
levels can have other YAML files imported as includes	YAML files can be loaded into the currently edited level
objects from imported include files can be used in the editor	objects in include files are found and are presented to the user to use in the editor

Figure 13: Smoke tested features

6.2 unit tests

once the initial application was created, i created a suvery to be able to give to tester of the application.

7 Evaluation

in this project, i successfully created a functioning application that could import and export levels which could be modified and were still playable within the game Co-Operation. the application adhered to the level specifications in that features described in the modding guide could be visible in the level editor. the application required reverse engineering that was successfully undertaken in order to achieve the 1:1 level

visualisation. with the implemented controls and procedures for map and object loading and manipulations, users can now use the tool to independently design and create levels with little to no understanding of YAML or the correct layout required.

within the application all the functional and mentioned non-functional requirements where added. this helped polish the product as quality of life features were added such as middle mouse clicking for copying, next placing objects name is displayed, and the object highlighting functionality for feed back to the user about the currently focused on object. with a good user experience, the application is easier to use and understand. this means the target audience stated at the beginning, being people with no knowledge of YAML, can now have an easier time creating their own levels. the secondary demographics, which are the individuals who already have mod creation experience, will also benefit from this application. this is because it allow for quick level creation as well as easy switching between the editing of levels, meaning other versions of the same level can be adjusted quickly. i did two sets of tests, one with a experience user and another with individuals who have never touched CO-Operation modding before. the experienced users were able to create levels in under 10mins and produce almost complete version of a level with all three variants with in an hour. the inexperienced was able to select, place, move and delete object from the scene, and then view it the level within the game.

with all the great progress made there where some aspect of development that could have gone a lot better. agile was the assigned method to use through out the project for management and completion of tasks. i didn't strictly keep to the agile methodology, i instead dipped between waterfall and agile. however part way through the development i ended up spending some time sorting out trello with the previous sprints i had completed. by doing this i as able to clearly see what features and aspects i had added meaning i could see what features needed to be added afterwards. though without the proper use of agile i believe more features could have been added with a longer period of polishing.

another issue that did rear it head was the realisation of anonymous object in the cell definitions them selves. at first i thought that i could ignore this issue as if i didn't support it would mean exported level wouldn't use it and i wouldn't need to handle it. however when looking through older levels as well as the game base levels, they were used quite a bit. this meant i needed to start supporting them as i wanted to have compatibility with level that weren't originally made with the tool. this lead to the conversion of a class list to a "object" list with the required conversion made further down the parsing chain. i ended up having the objects get stored in the levels file with the name anonymous_¡randomNumber¿ so that if there were multiple objects they wouldn't be saved with the same name. this way works and it does successfully parse the objects out, however i feel the names should me more descriptive. the name

formation could be ¡direction¿_¡firstObjFileName¿_¡randomNumber¿. the reason for this is that finding the objects would be easier within the object list as it would describe the object more instead of just a random name.

8 Conclusion

the objective for this application was to create a tool that could let people, with varying knowledge and experience in YAML and Co-Operation level design, design and create levels easily and faster than what would useable be possible through manual text file manipulation. this goal was met and did allow easy creation of levels from both experience and the inexperienced users. however due to the planning method, features and important fixes were not implemented. this consisted of the tracking and fixing of the memory leak, and image view of objects so users know what they look like before placing them. with most functional requirement being implemented with a lot of non-functional features also appearing, this was a successful project in my opinion with the production of a open-source application.

there is a potential to make the loading speeds faster. the majority of the waiting time is for the loading of GLB object into the scene. i chose to use GLTFUtility as i used it before and it had straightforward use within the code. while using it was easy there where other libraries out there that are potentially faster. one made by the khronosgroup, the royalty-free company who directs interpretability standards of 3d formats, as well as a library called glTFAST which from the name could potentially speed up the loading times. this is a future plan as complete overhaul of the object importing would need to be done, making sure that objects get imported the correct way laid out by the library.

due to the open-sources nature of this project there is great potential for extensive updates and support. due to the openness of the project, both the community and i can contribute to the progression of the project afterwards. there is also a potential that MINDFEAST might contribute/take the projects code for implementation into their game.

9 appendix

```
1 public GameObject createObject(string name)
2     {
3         Vector3 newPos = new Vector3(0, 0, 0);
4 
5         ObjectClass obj = allObjects[name];
```

Art2D

Defines a custom 2D texture to be added to an object, displayed as a quad, or as a decal projected onto a surface. One or more `art2D` can be added to a level object.

Required Fields:

Field	Type	Description
texture	string	The name of a .png or .jpg/jpeg file placed in the package's Art/2D folder (or subfolders), e.g. "MyImage.png" or "SubfolderName/MyImage2.png".
OR		
textures	List of string	Use this if multiple texture files are required. This is sometimes needed when defining a Spine character (displayType = "spineAnimation") e.g. ["MyImage.png", "MyImage2.png"]

Optional Fields:

Field	Type	Description
normalMapTextures	List of string	A list of names of .png or .jpg/jpeg normal map texture files placed in the package's Art/2D folder (or subfolders). Note: Normal maps are currently only applied to Spine characters (displayType = "spineAnimation"). The number of normal maps should match the number of textures. e.g. ["NormalMap.png"] (a single normal map for a single texture) ["NormalMap.png", "NormalMap2.png"] (if there are two textures, etc.)
displayType	string	Can be set to: "quad" (a standard textured quad) OR "billboard" (a textured quad that faces the camera) OR "decal" (a projection of a 2D texture), OR "spineAnimation" (an animated Spine character) Default value: "quad"
metallic	float	A value between 0 and 1, sets the metallic value of the shader. Default value: 0.1
smoothness	float	Same as above, but for the shader's smoothness value. Default value: 0.6
transparent	boolean	Only used when displayType is set to "quad" or "billboard". When true, a transparent material will be used, with no shadow casting. When false, an opaque material will be used, with alpha clipping and shadow casting. Default value: true
projectionDistance	float	Only used when displayType is set to "decal". Determines how far the decal will be 'projected'. For adding decals to walls (and for most cases) this can be left as is. Default value: 1.5.
pos	Vector3Optional	Position relative to the centre of the tile the object is placed on. -1 or +1 in x/z represent the edges of the tile. For quad & billboard displayTypes, y = 0 represents floor height. For decals, y = 0 represents a height of gridSize * 0.5, so decals will project at the horizontal and vertical centre of a wall by default. Default value: (x: 0, y: 0, z: 0)
rot	Vector3Optional	Rotation applied on top of the object's set direction (dir). Default value: (x: 0, y: 0, z: 0)
scale	Vector3Optional	The scale of the 2D texture. Default value: (x: 1, y: 1, z: 1)

Figure 14: supported features of art2d

```
6         GameObject HolderObj = new GameObject();//holds al the models
  for object
7         HolderObj.AddComponent<ObjectAttributes>().objectName = name;
8         HolderObj.name = name;
9         HolderObj.transform.position = new Vector3(newPos.y, 0, newPos.
  x);
10
11        bool visible = false;
12        //take account of base obejcts, id and mapObject wont work as
  both can get resolved to in game objects uavaliable for viewing
13        if (obj._base != null && obj._base.Count > 0)
14        {
15            foreach (string baseObj in obj._base)
16            {
17                bool visCheck = instintateObjAsBase(baseObj, allObjects
  [baseObj], newPos, HolderObj, obj.DirToAngle());
18                if (visCheck && !visible)
19                {
20                    visible = true;
```

Art3D

Defines a custom glTF/glb model to be added to an object. One or more `art3D` can be added to a level object.

Required Fields:

Field	Type	Description
`model`	`string`	The name of a .glTF/.glb model file placed in the package's Art/3D folder (or subfolders), e.g. "MyModel.glb" or "SubfolderName/MyModel2.glb".

Optional Fields:

Field	Type	Description
`pos`	`Vector3Optional`	Position relative to the centre of the tile the object is placed on. -1 or +1 in x/z represent the edges of the tile. Default value: (x: 0, y: 0, z: 0)
`rot`	`Vector3Optional`	Rotation applied on top of the object's set direction (dir). Default value: (x: 0, y: 0, z: 0)
`scale`	`Vector3Optional`	The scale of the model. Default value: (x: 1, y: 1, z: 1)
`faceCamera`	`boolean`	Whether the 3D model should always rotate to face the game camera. Default value: false

Figure 15: supported features of art3d

```
21                }
22            }
23        }
24
25        //display obejcst and images, if nothing renders then
  palceholder(capsule) to show the object
26        //import each object used
27        if (obj.art3d != null)
28        {
29            foreach (Art3d objsArt in obj.art3d)
30            {
31                visible = true;
32                GameObject Temp = ImportGLTF(workingDirectory + "/" +
  objsArt.model);
33                Temp.AddComponent<ObjectAttributes>().attributes3d =
  objsArt;
34                foreach (Renderer rend in Temp.GetComponentsInChildren<
  Renderer>())
35                {
36                    MeshCollider col = rend.transform.gameObject.
  AddComponent<MeshCollider>();
37                    col.convex = true;
38                    col.isTrigger = true;
39                    SkinnedMeshRenderer skinnedRenderer = rend as
  SkinnedMeshRenderer;
40                    if (skinnedRenderer != null)
41                    {
42                        // Create a new mesh and bake the skinned mesh
  into it
43                        Mesh bakedMesh = new Mesh();
44                        skinnedRenderer.BakeMesh(bakedMesh);
45                        // Assign the baked mesh to the Mesh Collider
46                        col.sharedMesh = null; // Clear old mesh
```

```
reference
                        col.sharedMesh = bakedMesh;
                    }
                }

                Temp.name = obj.dir;

                //CenterPivotAtBottomMiddle(Temp);

                Temp.transform.position = new Vector3(newPos.y, 0,
newPos.x);

                Temp.transform.position += new Vector3(-objsArt.pos.x,
objsArt.pos.y, -objsArt.pos.z);//position offset
                Temp.transform.rotation = Quaternion.Euler(0, 90, 0);//
rotate around y to get it into north east south west
                Temp.transform.Rotate(new Vector3(0, obj.DirToAngle(),
0));//rotate around y to get it into north east south west
                Temp.transform.Rotate(new Vector3(objsArt.rot.x,
objsArt.rot.y, objsArt.rot.z));//added roation for inital direction

                Temp.transform.localScale = new Vector3(objsArt.scale.x
, objsArt.scale.y, objsArt.scale.z);
                //Debug.Log(obj.dir);
                Temp.transform.parent = HolderObj.transform;
            }
        }
        if (obj.art2d != null)
        {
            foreach (Art2d objsArt in obj.art2d)
            {
                visible = true;
                GameObject Temp = ImportImage(workingDirectory + artDir
 + art2dDir + "/" + objsArt.texture);
                Temp.AddComponent<ObjectAttributes>().attributes2d =
objsArt;
                MeshCollider collider = Temp.GetComponent<MeshCollider
>();
                collider.isTrigger = true;

                Temp.name = obj.dir;

                //CenterPivotAtBottomMiddle(Temp);

                Temp.transform.position = new Vector3(newPos.y, 0,
newPos.x);

```

```
                Temp.transform.position += new Vector3(-objsArt.pos.x, objsArt.pos.y, -objsArt.pos.z);//position offset

                Temp.transform.rotation = Quaternion.Euler(0, -90, 0); //rotate around y to get it into north east south west
                Temp.transform.Rotate(new Vector3(0, obj.DirToAngle(), 0));//rotate around y to get it into north east south west
                Temp.transform.Rotate(new Vector3(-objsArt.rot.x, objsArt.rot.y, -objsArt.rot.z));//added roation for inital direction

                Temp.transform.localScale = new Vector3(objsArt.scale.x, objsArt.scale.y, objsArt.scale.z);

                MeshRenderer quadMeshRenderer = Temp.GetComponent<MeshRenderer>();
                quadMeshRenderer.material.SetFloat("_Metallic", objsArt.metallic); // 3 = Transparent mode in Standard shader
                quadMeshRenderer.material.SetFloat("_Glossiness", objsArt.smoothness); // 3 = Transparent mode in Standard shader


                //use this to support billboarding
                //if billboard or not quad(dafault to billboard if invalid)
                if (objsArt.displayType == "billboard" || objsArt.displayType != "quad")
                {
                    Temp.AddComponent<BillboardScript>();
                }

                //Debug.Log(obj.dir);
                Temp.transform.parent = HolderObj.transform;
            }
        }
        if (!visible)
        {
            GameObject Temp = Instantiate(placeHolder);
            Temp.transform.position = new Vector3(newPos.y, 0, newPos.x);
            Temp.transform.parent = HolderObj.transform;
        }
        return HolderObj;
    }
```

Listing 14: C# code for correct placement of object art from file to scene

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

using YamlDotNet.Serialization;


public class vec3
{
    public float x = 0, y = 0, z = 0;
}
public class vec3Scale
{
    public float x = 1f, y = 1f, z = 1f;
}
public class Art3d
{
    public vec3 pos = new vec3();
    public vec3 rot = new vec3();
    public vec3Scale scale = new vec3Scale();
    public string model = "";
}

public class Art2d
{
    public vec3 pos = new vec3();
    public vec3 rot = new vec3();
    public vec3Scale scale = new vec3Scale();
    public string texture = "";
    [YamlMember(Alias = "display_type")]
    public string displayType = "";
    public float smoothness = 0.6f;
    public float metallic = 0.1f;
}

public class Data {
    public Dictionary<string, object> dataItems = new Dictionary<string
    , object>();
}


public class modWithData {
    //the mod name
    public string name = "";
    public Dictionary<string, object> data = new Dictionary<string,
    object>();
    //
}

```

```
48
49  public class ObjectClass
50  {
51      enum _dir {
52          east = 3, north = 2, west = 1, south = 0
53      }
54      public float DirToAngle() {
55          if (dir == "north") { return (float)_dir.north * 90; }
56          else if (dir == "east") { return (float)_dir.east * 90; }
57          else if (dir == "south") { return (float)_dir.south * 90; }
58          else{ return (float)_dir.west * 90; }
59      }
60
61      public string mapObject = "";
62      [YamlMember(Alias = "base")]
63      public List<string> _base { get; set; } = new List<string>();
64      public string id = "";
65
66      public string dir = "south";
67
68      public List<string> tags = new List<string>();
69
70
71
72      [YamlMember(Alias = "art3d")]
73      public List<Art3d> art3d { get; set; } = new List<Art3d>();
74      [YamlMember(Alias = "art2d")]
75      public List<Art2d> art2d { get; set; } = new List<Art2d>();
76
77      //can be both a simple string or a class modWithData
78      public List<object> mods = new List<object>();
79
80      public Dictionary<string, object> data = new Dictionary<string,
        object>();
81  }
82
83  public class FileProperties
84  {
85      public string creatorName = "LevelEditor";
86  }
87
88  public class DepthOfField
89  {
90      public bool enabled = false;
91      public float focusDistance = 58;
92      public float focalLength = 0.0f;
93      public float aperture = 1.0f;
```

```
}
public class PostProcessing
{
    public DepthOfField depthOfField = new DepthOfField();
}

public class Music
{
    public bool usePresent = false;
}

public class Sounds
{
    public List<string> fileNames = new List<string>();
}

public class LevelFile
{
    [YamlMember(Alias = "include")]
    public List<string> include { get; set; } = new List<string>();

    [YamlMember(Alias = "file_properties")]
    public FileProperties fileProperties { get; set; } = new
   FileProperties();

    [YamlMember(Alias = "scene_name")]
    public string sceneName { get; set; } = "EmptyWorld";

    [YamlMember(Alias = "post_processing")]
    public PostProcessing postProcessing { get; set; } = new
   PostProcessing();

    [YamlMember(Alias = "grid")]
    public string grid { get; set; } = "  AA,BA,CA\n  AB,BB,CB\n  AC,BC
   ,CC";

    [YamlMember(Alias = "grid_objects")]
    public Dictionary<string, List<object>> gridObjects { get; set; } =
    new Dictionary<string, List<object>>();

    [YamlMember(Alias = "object_definitions")]
    public Dictionary<string, ObjectClass> objectDefinitions { get; set
   ; } = new Dictionary<string, ObjectClass>();

    [YamlMember(Alias = "sounds")]
    public Dictionary<string, Sounds> sounds { get; set; } = new
   Dictionary<string, Sounds>();
```

```
135
136     //generic feild so i dont know whats in there
137     //not sure on this feild so change when needed
138     [YamlMember(Alias = "global_data")]
139     public Dictionary<string, object> globalData { get; set; } = new
        Dictionary<string, object>();
140 }
```

Listing 15: classes for yaml loading

objectDefinitions

objectDefinitions maps from the `gridObjects` (e.g. B, Pt) to one (and only one) LevelObject.

LevelObject

An object definition that is resolved into a MapObject.
Can either:

1. Be a fully-described "Custom" object (e.g. with a custom glTF model added as an Art3D),
2. Reference a MapObject prefab built into the base game,
3. Either **1** or **2**, and additionally reference one or more other LevelObjects to use as a base

Required Fields:

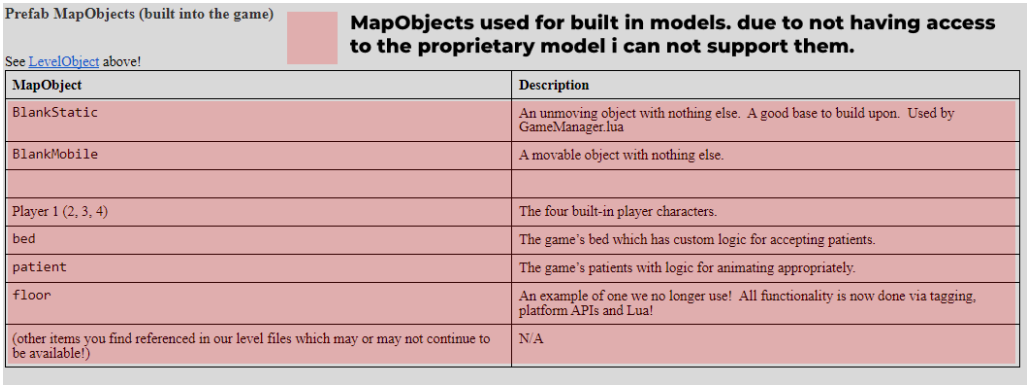
only "custom"

Field	Type	Description
mapObject	string	mapObject should be used for **1** and **2** (see above). e.g: 'mapObject: Custom' for a fully-described object 'mapObject: Player 1' to reference a player built into the game
OR		
base	List of string	base should be used for **3**. The object will inherit the properties of any listed base objects. E.g. 'base: [Obj1, Obj2]', where Obj1 and Obj2 are keys of other LevelObjects within objectDefinitions.
OR		
id	string	id is lazily resolved as either a mapObject or a base.

Optional Fields:

Field	Type	Description
dir	Direction	A direction the object faces Default value: South
mods	List of Mods	A list of mods (which can optionally include `data` for just this mod). Default value: Empty
data	Key: string, Value: object	Data that will be provided to **all** mods on this MapObject. Default value: Empty
art3d	List of Art3D	Describes custom 3D art (i.e. glTF/.glb files). Default value: null
art2d	List of Art2D	Describes custom 2D art. Default value: null
textMeshes	List of TextMesh	Describes custom text mesh(es). Default value: null
tileScale	Vector3Optional	Scales the object so that the size of its overall bounding box (which covers all 3D models) is a multiple of the size of a single grid tile in any given dimension. Default value: null (not applied)
centreToTile	boolean	When true, the object will be positioned so that the centre of its overall bounding box is at the centre of the tile it's placed on (on the xz plane). Default value: false
tags	List of string	Tags applied to the object that can be used by both C# and mod code to alter behaviours. There are some system-special tags (see below). Default value: Empty

Figure 16: support for object definitions



Prefab MapObjects (built into the game)

MapObjects used for built in models. due to not having access to the proprietary model i can not support them.

See LevelObject above!

MapObject	Description
`BlankStatic`	An unmoving object with nothing else. A good base to build upon. Used by GameManager.lua
`BlankMobile`	A movable object with nothing else.
Player 1 (2, 3, 4)	The four built-in player characters.
`bed`	The game's bed which has custom logic for accepting patients.
`patient`	The game's patients with logic for animating appropriately.
`floor`	An example of one we no longer use! All functionality is now done via tagging, platform APIs and Lua!
(other items you find referenced in our level files which may or may not continue to be available!)	N/A

Figure 17: unsupported MapObject values