Date: 11/11/24

UniSim

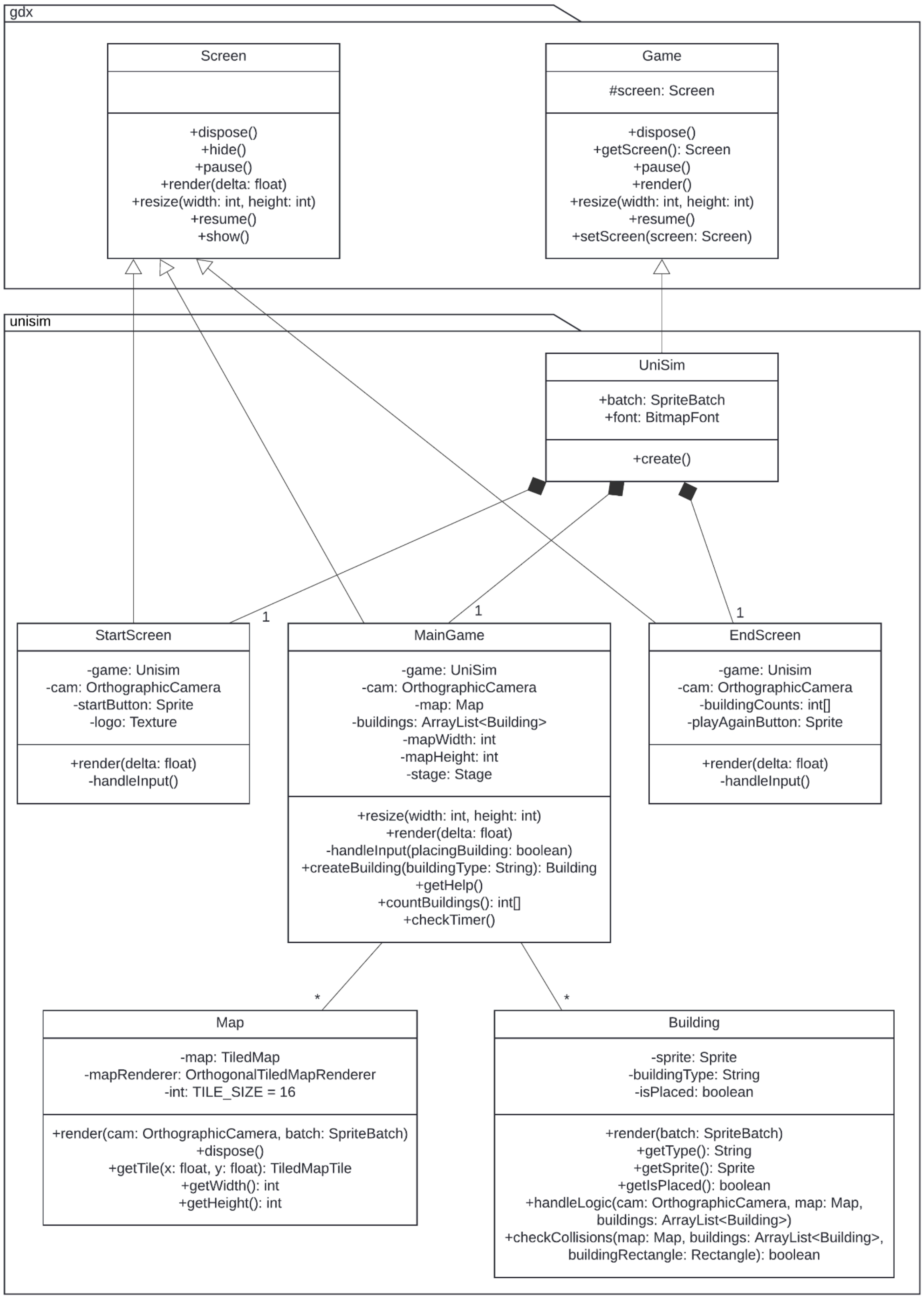
ARCHITECTURE

TEAM6 Game Studios

Work Undertaken by:

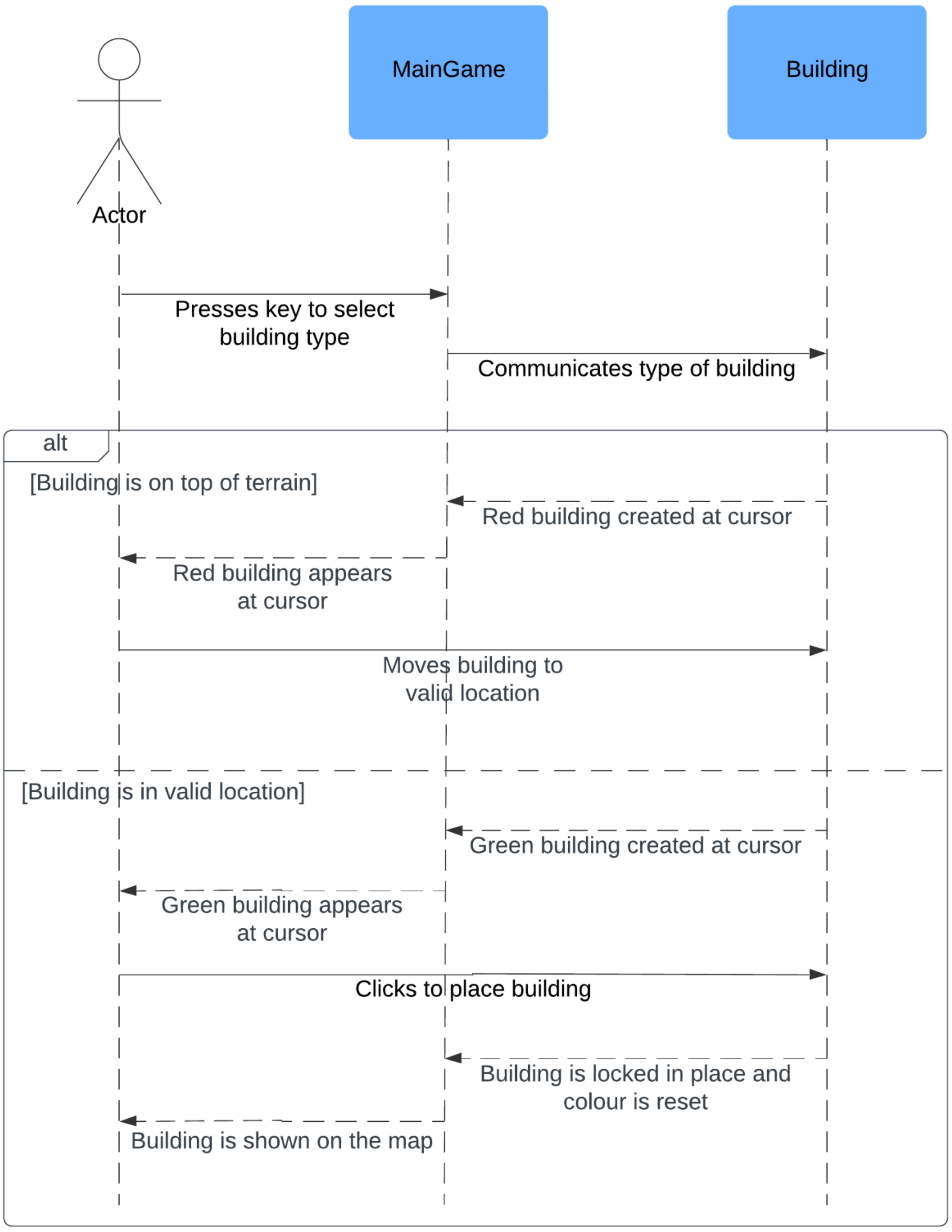
Thomas Koukouris
Adam Khan
Oliver Herron
Sam Jordan
Nathan Hopper
Fergus Irvine

STRUCTURAL DIAGRAM



This is a UML class diagram that details the classes used in the game and their relationships. Some of the relevant libGDX classes are also shown here. Each class is represented by a box with three compartments. The top compartment contains the name of the class, the centre compartment contains the attributes, and the bottom compartment contains the methods. The symbols in front of the methods and attributes indicate the visibility, a ‘+’ sign indicates that it is public, whilst a ‘-’ sign indicates that it is private. The types are listed after the colons, showing the types of each attribute, each method parameter, and the return type of each method. Lines between classes indicate that those classes are associated. The multiplicity is indicated by the numbers, they represent how many instances of this class are associated with another class. Integers represent a fixed number or range, but asterisks represent any number of instances. Diamonds represent that the class is composed of the associated class, meaning that if the class is deleted, the classes it is composed of would also be deleted. Arrows represent that the class inherits from the associated class it is pointing to. Finally, the boxes around the classes indicate what package they are a part of, not all gdx classes are shown here.

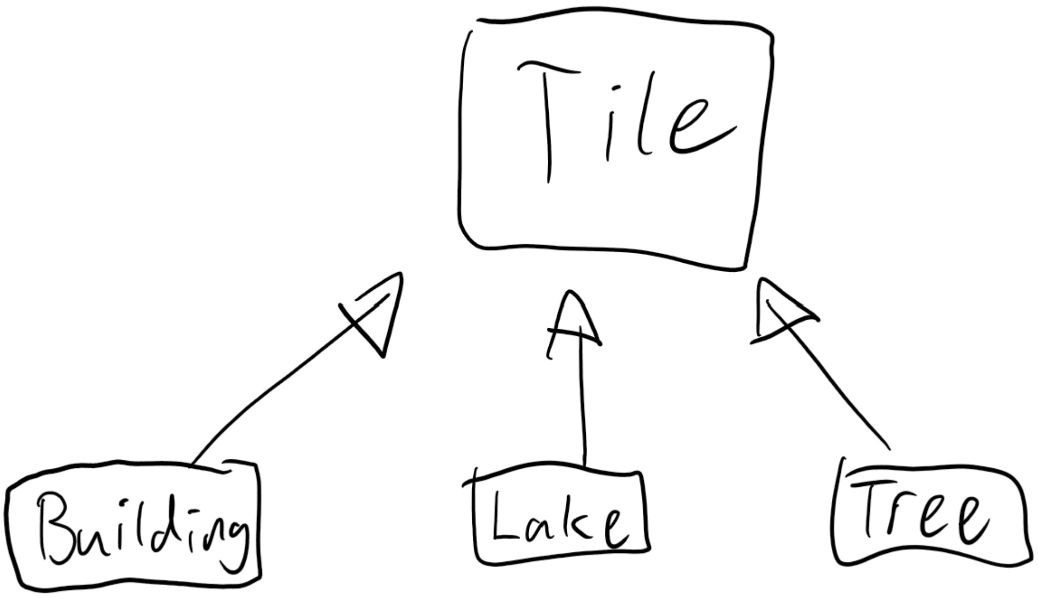
BEHAVIOURAL DIAGRAM


MainGame
Building
Actor
Presses key to select building type
Communicates type of building
alt
[Building is on top of terrain]
Red building created at cursor
Red building appears at cursor
Moves building to valid location
[Building is in valid location]
Green building created at cursor
Green building appears at cursor
Clicks to place building
Building is locked in place and colour is reset
Building is shown on the map

This is a sequence diagram that shows the actions that the system and the player take when placing a building in the game, the actions are listed chronologically, with the action at the top being the first action taken and the action at the bottom being the last. The box labelled ‘alt’ shows two alternative scenarios that could play out when attempting to place the building.

These diagrams were created using LucidChart, a free to use web-based diagramming application. It comes with many UML diagram templates built-in, and has a simple drag-and-drop interface that makes creating or editing diagrams quick and easy. Using the templates, shapes such as the three compartment boxes and diamond-ended arrows were able to be added directly from the menu, eliminating the need to create the shapes from simpler ones. Group collaboration features were also helpful, as they allowed anyone in the group to view changes made to the diagrams immediately.

ARCHITECTURE DESIGN



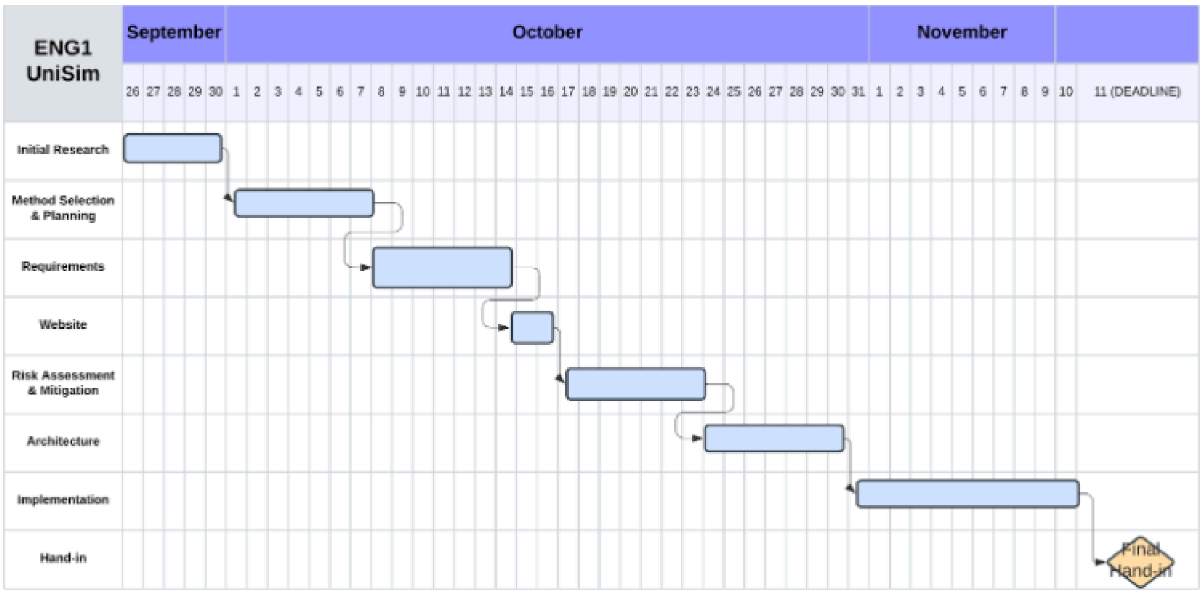
Initially, the game was going to be based on tiles, with each tile being able to contain one element. Each tile would have had a sprite and coordinates. A tile could contain a building, a tree, water, or any other structure we decide to add in the future. Although, after some discussion, we later scrapped this idea as we decided that this architecture would make it more difficult to place larger structures, like bigger buildings, that may cover more than one tile.

We ended up settling on a simpler architecture, using libGDX's in-built classes to our advantage. This architecture allows us to use maps created in Tiled, an open source map editor, in our game. This makes it easier for new maps to be added in the future or for existing maps to be edited. There are 3 different screens, a title screen, a view of the main game, and an ending screen. This meets the requirement UR_START_GAME, where the user can start a game by pressing a button, rather than having the timer start as soon as the game is opened. Buildings can be placed around the map, and the user can select different building types by pressing different keys. The architecture allows for more building types to be added in the future if needed. This meets the UR_BUILDING, UR_BUILDINGS_LEARNING, UR_BUILDINGS_LIVING, UR_BUILDINGS_RECREATION and UR_BUILDINGS_FOOD requirements. To meet the UR_BUILDINGS_PREVIEW requirement, buildings show a preview on the map before the user places them, to ensure that the user places them in their desired location, and to check that the user does not try to place buildings in invalid locations, such as on top of a tree or road.

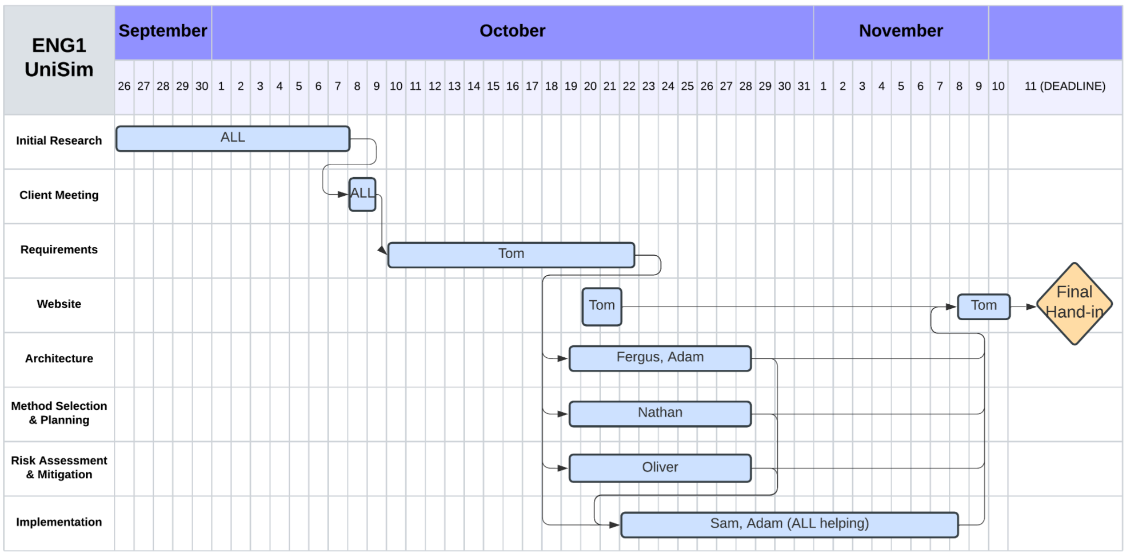
As per the briefing the game had to be a University Simulator, therefore that was the central focus to deliver.

Resource Type	Name	Reason	Other options
File Sharing	Google Drive	It is easy to use and files can be well organised. Our group is also already familiar with it.	
Coding Language	Java	We are required to use Java for this project.	
Game Engine	LibGDX	Least complex and likely to be used by other groups. It also has good documentation and is performance friendly.	LWJGL Or Slick2D
Code Sharing	Github	Widely used and useful to know how to use it. A lot of support/guides are available	
Website Hosting	Github	Easy to use to create a simple static website which is all that is needed for our project. Easy to provide links to deliverables as well as a description of our project	https://webstudio.is/
Map Design	Tiled		
UML Diagram	Lucidchart		
Java Documentation	OpenJDK	Extremely useful when coding in Java	
Game Assets	Kenny.nl	A large collection of free 2D game assets, very helpful as we are not required to spend time creating our own assets	

This was the initial resources list we had and would build off of, consisting of the resources we would utilise for the project's overall development.
Overall we had set out the ideal projection for the project to be developed within, and had the tools necessary to properly begin. As the project developed more techniques, and greater resources were used which will be outlined through the architectural diagrams.
The initial GANTT chart looked like this:



However the eventually revised version which was more detailed and updated to more accurately encapsulated the workflow:



This includes the lifetime of the project, as well as clear roles and responsibilities upheld by members of the team throughout.
The parameters of the Project were reflected in the initially forecasted Classes, Menu, MainGame and EndGame. Although we finished with these, we added additional classes like Map, Assets (Prototype for the Menu class), and multiple prototypes for the Menu due to some issues with the initial implementation in LibGDX and JFrame - they were then used as reference. We had to educate ourselves on the usage of GitHub, in which we factored our code to not interfere with one another but still remain open to the group for critique and potential updates.