SEPR 2019/20 Assessment 1

Team CheatCodez

Jonathan Grout, George Lesbirel, Cheuk Wang Wu, Lauren Quarshie, Muaz Atif, Lillian Coultas

Architecture

Abstract Representation of Architecture

In this section we will look over our plan to the software structure of Kroy. We will cover an abstract construction of classes using Unified Modeling Language (UML), *Appendix 2.1*. Followed by a description and justification of the way we have designed it this way. Firstly, we will look over an abstract view of how our objects will interact with each other and the environment. In general we will use getters and setters for retrieving variables such as damage and hit points, although these have not all been included in the UML.

Our game will be played from a birds eye view. Players will be able to traverse parts of York and take different entrances and exits to reach other sections. Each region will contain a home base that they need to protect and alien base(s) they need to attack. Enemies will spawn from their bases, there will not necessarily be one enemy base in each region. At a lower level, we aim to store all the regions of York in a Map (e.g. HashMap in Java). This will then be split up into 9 boxes (regions), with the key being the label of the stage. The value of each element will be an Array of the elements (Objects) in the stage. When the user leaves a region, the transition may trigger a minigame.

Our minigame consists of the user pumping water as fast as they can. Depending on: the difficulty they are on, length of playing and how much they pump, they will get different rewards and items. This minigame won't be triggered every time the player makes a transition as we feel this could result in the user having too many bonuses at one time. The amount that the minigame is triggered with depend on how far through the game they are and difficulty. The user is given the option to skip the minigame. *Appendix 2.2* shows the behaviour of our minigame.

Figure 2.1 shows the UML diagram, a larger version can be found in the appendix , *Appendix 2.1.* It is a basic visual representation of how the classes are related to one another and their purpose. This includes basic values that will be stored (such as hitpoints and position) and operations that each class should be able to perform. It shows a visual structure of the program and will be beneficial to speeding up the development of the game by providing an abstract document to refer back to.

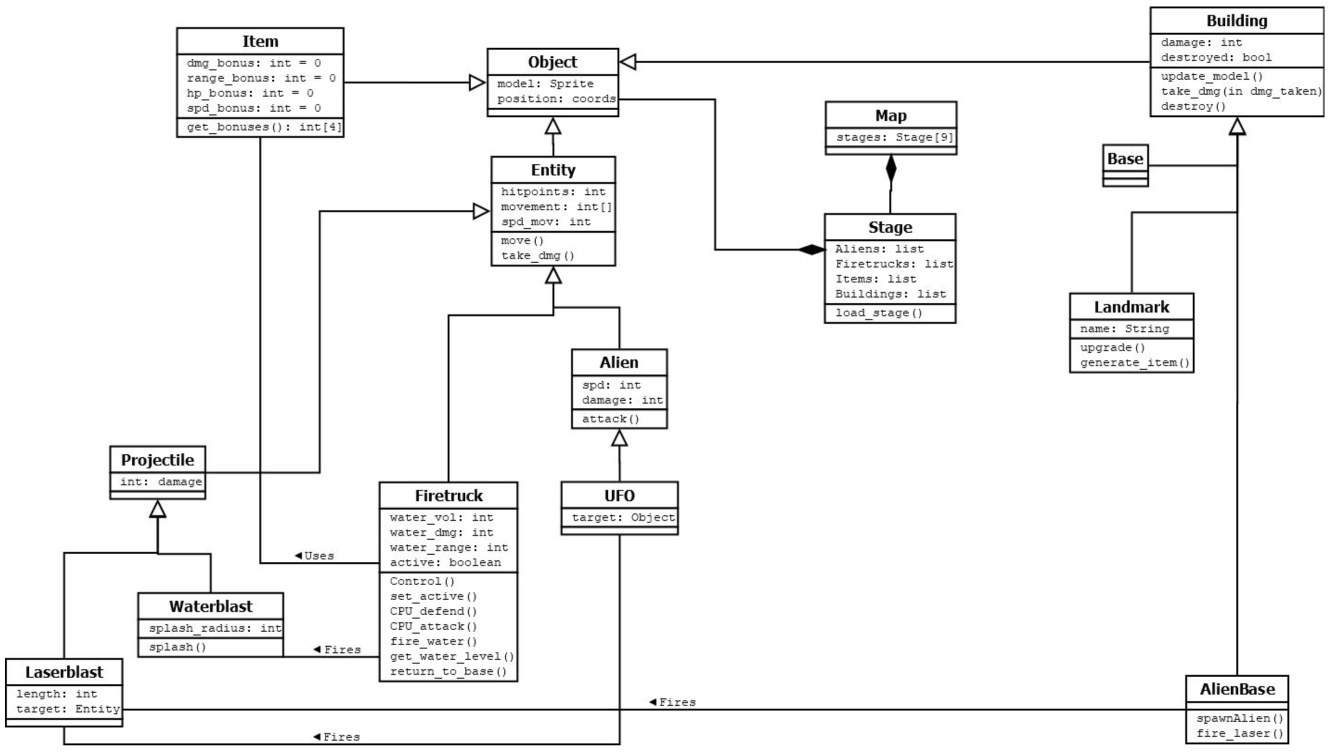

Figure 2.1 - The main game UML

Justification of Architecture

Software Component	Description/Justification	UML Explanation
Object	- Model is used to store the sprite models for each object - Position holds the position of the object in the game which will be mapped onto the libgdx coordinate system - The object class is the base class for most other classes in the game	Composition relationship with Stage as if the Stage is destroyed so are all Objects in that Stage
Entity	- Movement would store something like a transformation matrix for the movement of the entity - applied when move operation is next called - Spd_mov is the movement speed which would correspond to how many times the movement vector is applied - Hitpoints store the amount of health an entity has. - Move and take_dmg are functions that will be part of the interaction with the game. The entity class inherits from the Object	

	class. Entities are movable objects that can be interacted with e.g. Aliens can be attacked, they must be able to move and give damage.	
Firetruck	- Water volume for storing "ammo" - Water damage, how much 1 hit does - Speed - isActive - CPU_attack and CPU_defend are used to run the attack and defend operations on the fire trucks that aren't controlled by the player - Inherits from Entity	CPU_attack and CPU_defend was used to show that attack and defend operations are also going to be implemented to be handled by the CPU Composition relationship with Stage as if the Stage is destroyed so are all Firetrucks in that Stage Association with Waterblast because Waterblast is fired by Firetruck
UFO	- For targeted attack after x amount of time at random target Inherits from Alien	UFO will only attack so has no listed operations as it inherits from Alien Association with Laserblast because Laserblast is fired by UFO
Alien	- Master alien class - HP - Attack points - Set and get - Inherits from Entity	Attack operation to show that aliens mainly attack enemies
Projectiles	- Damage dealt - maxDistance - (speed inherited)	
Waterblast	- Splash radius on waterblast deals fractional amount of damage to surrounding aliens	
Laserblast	- Length of laserblast which corresponds to amount of damage	
Building	- HP Basic class for structure objects that don't move	
Base	Base for firetrucks to return to and refill water and repair damage	
Landmark	- Name Can be upgraded to increase health of	

	Landmark and improve its item generation	
AlienBase	- HP	spawnAlien is the only operation of the AlienBase as the only purpose of the AlienBase is to be a primary target for the player and to be the origin of Alien Entities It has an association with laserblasts as it is able to fire laserblasts at fire trucks randomly
Item	Items can improve up to 4 different stats on a Firetruck. Each item can have its bonuses applied to any firetruck. It is generated by Landmarks and can be given by the minigame too	
Stage	Stores all the objects that are in each stage including their positions	
Map	World in which the game is played Collection of all the stages in an array	Things in stages are handled in child class Stage and object instances