University of York

Department of Computer Science

SEPR - Assessment 3

Implementation Report

Team Craig

Thomas Burroughs

Huw Christianson

Joseph Frankish

Isaac Lowe

Beatrix Vincze

Suleman Zaki

The selection of Geese Lightning's project was influenced by their clear, concise and testable game requirements, as seen in the updated requirements specification. Minimal changes were made to either the layout of the requirement specification or the requirements themselves. No additions were made to the requirements table, however, fit criteria P2.2 for requirement P2 was removed from the specification because it could not be met using the resources available to us. For each new implemented feature either black-box or unit tests have been created to test if it fulfils the requirement not fulfilled by Geese Lightning. To improve traceability a traceability matrix has been created to more easily show which requirements have been fulfilled and tested.

On the selection of Geese Lightning's project, our programmers took time to understand the architecture of the game and discuss whether to adopt their quality standards and coding conventions. The team's programmers agreed that we should adopt Geese Lightning's class architecture e.g. using entities and coding conventions as they were very familiar with ours for Assessment 2. These adoptions required very little effort to implement. Programmers met several times before implementation began to discuss how the team could extend the game using pre-existing methods/classes to maximise the time available for making additions to meet the requirements.

In order to meet the full brief and meet the requirements put in place by Geese Lightning numerous features needed to be added. These include the addition of 3 other locations, a 3rd character type, 2 other powerups, 2 other zombie types, 2 new evil bosses and a 'save and load' feature. Geese Lightning stated in their Implementation report that requirements F5, F7 and F8 had not yet been implemented due to time constraints. As of the end of Assessment 3 all these requirements have now been fully implemented. The additions made by our team have also ensured that each of the requirements partially implemented by Geese Lightning (F1, F3, F4, F6) have now been fully implemented. Furthermore, the team has also made changes to the game's previous functionality, namely: character and object hitboxes, player movement and fixing a game ending crash. These changes have improved the game's previous functionality improving the user experience and ensure the game better fulfils the requirements, specifically- F9, N2, E1. This report details each of the new features and changes to previous features by describing the changes/additions made to the previous team's code and the related requirements which the feature fulfil. For each feature, a suitable explanation and justification has been made in terms of improvement to user experience as well from an architectural viewpoint.

Team Geese Lightning's previous architecture was well represented by their UML diagrams and minimal changes needed to be made. However, some changes were necessary in order to fix or improve the game's previous faulty functionality. Any additions/changes to source code or are architecture etc are reflected within our updated UML diagram and can be seen within our newly created change table.

- Full UML diagram: https://teamcraigzombie.github.io/assets/downloads/UpdatedUML3.pdf
- Change Table: https://teamcraigzombie.github.io/assets/downloads/ChangeTable3.pdf
- Updated Requirements: https://teamcraigzombie.github.io/assets/downloads/UpdatedRequirements3.pdf

Changes to previous functionality

Feature:	**Hitbox adjustments**
Requirement:	F9
Changes/ Additions	Addition of an image layer to preserve the appearance of objects that had tiles removed from the collision layer. Change to update() method in Character class to relocate vertices locations. Addition of wallCollision() method to Character class to better handle collisions.
Explanation/ Justification	Previously both the characters and zombies would snag on the edge of collidable objects including each other. Relocating the vertices in the update() method to be slightly further away from the player helped because the collision points were now ahead of the player so movement would be stopped before part of the player entered the collision tile. Addition of the wallCollision() method was needed for the new collision code which would now stop movement in the specific direction in which the collision occurred, rather than stopping all movement. This code is self-contained and therefore, does not need to be in the update() method and can be in its own method. Having it in its own method also improved readability. These changes have been justified as they allow the player to move between collidable objects with greater ease and ensures characters can move between spaces which the player perceives to be passable,

	preventing frustrations. Furthermore, this change was justified as it allows Zombie AI to be more effective as Zombies have less opportunity to get stuck, which would've reduced the game's difficulty.

Feature:	**Player Movement**
Requirement:	E1
Changes/ Additions	Changes made to keyUp() method in the ZeprInputProcessor class.
Explanation/ Justification	Previously the player's character would be prevented from moving when keys were pressed simultaneously. The games would stop movement on an axis if a key corresponding to the axis was released regardless of if the opposing key was still being pressed. The keyUp() method has been changed so that now when a key is released it first checks to see if the opposing key on that axis is still being pressed or not. If it is then movement is swapped to the other direction. If it isn't then the movement in that axis is stopped. This change prevents any instances where a key is being pressed but the player is not moving. These changes have been justified as they allow the player to move through the map with more fluidity and ease. This helps to prevent players from becoming frustrated when improper movement results in damage from incoming zombies/bosses, which should hopefully ensure players enjoy the game more.

Feature:	**Game ending crash/error**
Requirement:	N2
Changes/ Additions	Removal of renderer.dispose() call from the dispose() method in the Level class.
Explanation/ Justification	At the end of the Halifax level a spontaneous crash would occur, preventing the player from completing the level and hence the game at that point. The crash would create an error dump file. Upon analysing this file it was realised that the crash was related to the renderer attribute of the Level class when the dispose() method was called. Therefore, the problem was with the renderer.dispose() call. The removal of this call prevented any attempts to access restricted memory and thus cause access violations; stopping the crash from occurring. It was also deemed that the dispose call was unnecessary as any pointers to the renderer would be lost anyway when the screen was switched therefore removing it from memory. This change is justified as it allows the character to complete the game, satisfying requirement N2. The crashing of the game would have led to player frustration and loss of interest due to the loss in progress. Players can now set the goal of completing the game knowing crashes will not prevent them from reaching their goal, hopefully providing more motivation and enjoyment.

Newly added featured

Feature:	**3 additional locations**
Requirement:	F1
Changes/ Additions	Additional classes for each new level extending the Level Class
Explanation/ Justification	A CS Building Level, Greg's Place Level and Library Level have been added to the game with menu buttons added with descriptions to add to the storyline. Each new level required a new class which extended the Level class as can been seen in the architecture, preventing the need to re-write the previous group's methods. Within each class the 'waves' array was adjusted to ensure that each level had zombie waves one greater than the level before, helping to achieve requirement **F2.** Furthermore, each level had a newly created Tiled map to help distinguish between the levels and provide the player with new challenges. The levels were added to the Switch Case in the Zepr class in the changeScreen method. This Switch Case was used to switch between screens when the 'progress' was updated by complete() method in each levels class allowing the player to progress through the game. These changes have been justified as they ensure requirement F1 is fulfilled and allows for two evil bosses to be implemented one which needed to be implemented on the 6th level (Library Level). Having more levels makes the game feel more complete so provides more of a storyline, hopefully engaging the player more with the game.

Feature:	**2 additional power-ups**
Requirement:	F6

Changes/ Additions	Additional classes added for each new power up extending the PowerUp class.
Explanation/ Justification	A 'Nuke' power up was added which ends the current wave instigating the next wave to begin. Upon walking over this power up it is activated, and its delay and timer are started. Once the delay finishes, all the alive zombies in the current wave are killed. This delay makes the nuke function a little more realistically like a bomb. Once the timer ends then the cooldown period is over so the power up deactivates and the next wave begins. The timer and delay are needed so that the nuke does not kill zombies in any wave other than the current wave. An 'Insta Kill' power up was added which allows the character to kill each zombie on one hit. When the player walks over this power up a timer starts, and it activates giving the player the ability to kill zombies in one hit. Once the timer ends the power up deactivates. Having new classes which inherit from the PowerUp class keeps in line with the architectural structure already present. These changes have been justified as they ensure requirement F6 is fulfilled. Adding more power ups to the game adds greater variation to gameplay creating more excitement and opportunities for players to make critical decisions, deepening player's interests. Nuke and Insta Kill powerups were chosen as many players will be familiar with their functionality from other games, so little confusion should arise from their use.

Feature:	**Additional character**
Requirement	F3
Changes/ Additions	Changes made to respawn() method in Player class which handles the properties of the additional character.
Explanation/ Justification	An Engineer character type has been added that has an increased attacking range so can attack Zombies from a further distance. The player can select this character type from the SelectLevelScreen as with the other character types. To handle the new player type, some new code was added to the method already present in the Player class which already implemented the other player types. This new code assigns values to the relevant attributes of the Player class to give this new character its unique stats. This change has been justified as it ensures requirement F3 has been fulfilled. it also provides greater variation to gameplay, like many new features, making the game more engaging for players. Having numerous character types allows the player to add the replayability value of the game so players can experience the game with new challenges.

Feature:	**2 evil bosses**
Requirement	F7
Changes/ Additions	Addition of new classes for each boss which extends the Zombie class;
Explanation/ Justification	At the end of the Courtyard level a 'Big Chef' boss spawns after defeating the final wave of zombies. This boss has increased health and therefore takes more damage to defeat. The level cannot be completed until the boss is defeated. he final level spawns a 'Librarian' boss. The two bosses were implemented by changing the classes constructors to assign different values to its attributes, such as assigning a drastically increased health value. New classes were required as any other approach would require changes to many other classes. Adding new classes also keeps in line with the approach used to add new zombie types. These changes are justified as they ensure that requirement F7 is fulfilled. Adding bosses to the ends of levels provides a new form of challenge for the player and provides a useful marker of progression, elevating the player's sense of achievement after completing the level. This, in turn, improves how enjoyable the game is.

Feature:	**Mini-game**
Requirement	F5
Changes/ Additions	Addition of new MiniGameLevel class which extends Level class. Addition of a Goose class which extends Character class. Changes to touchDown() and touchUp() methods in ZeprInputProcessor class. Addition of new boolean attribute in Player class.
Explanation/ Justification	A mini game has been added to the game with a corresponding button added to SelectLevelScreen. In this mini game the player must shoot geese flying across the screen in order to score points. If the player misses a shot, then they lose a bullet. Once they lose 5 bullets the game ends and their final score is displayed. Extending the Level class for the new MiniGameLevel class was useful as it prevented having to re write code. However, some methods such as the render() method had to be overridden as they required a

	different implementation. The class also required many new methods such as shoot() and isGooseMissed() as they implemented completely new functionality not present in any other part of the game as is the nature of mini games. A new Goose character class was needed as it had different values for its attributes but extending Character meant much more code did not have to be rewritten. In order to support the functionality of the methods in the MiniGameLevel class, some access levels for some attributes in the Level class had to be changed so that MiniGameLevel could access them. Changes were required in the input processor and a new boolean attribute was required in the Player class so that player inputs could be received and used in the mini game. This change is justified as it ensures requirement F5 is fulfilled. A mini-game provides an alternate form of enjoyment for the player should they lose interest in the main game. This is because the mini-game can be played with less intensity or concentration. By including a mini game, we can ensure that players do not tire of the game's feature as quickly or 'burnout'.

Feature:	**2 additional zombie types**
Requirement	F4
Changes/ Additions	Addition of new classes for each new zombie type which extend the Zombie class. Changes to the spwanZombies() method in the Level class.
Explanation/ Justification	A Medic Zombie (increased health) and Sporty Zombie (increased speed) type have been added to the game. The number of each Zombie type, including the generic zombie, is randomly generated by the spawnZombies() method. Depending on which number is generated one of the types of zombies is spawned by the method and added to the aliveZombies array. This change to the spawnZombies() method is needed in order to randomly spawn the different types of zombies. New classes were used to implement the different zombie types. The constructors would assign the appropriate values to the corresponding attributes to make each zombie different. New classes were used as any other approach would require drastic changes to many other classes. This change is justified as it ensures requirement F4 is fulfilled. The numerous Zombie types adds a level of unpredictability to each level creating excitement and forces the player to change player styles adding further challenge.

Feature:	**Save and load game**
Requirement	F8
Changes/ Additions	Changes to show() method in SelectLevelScreen class. Addition of new attribute to SelectLevelScreen class.
Explanation/ Justification	To implement a save/load feature the buttons on the level selection screen were given listeners so that when they were pressed they would perform the desired action. A new attribute was added which contains the file handler for the save file. When the save button is pressed the file handler writes the current progress to the text file. When the load button is pressed the progress is set to what the file handler reads and the level select screen is reloaded in order to update it with the saved progress.

Features not fully implemented

All features required by Assessment 3 in the full product brief have been implemented as well as requirements stated by Geese Lightning's Requirement Specification, apart from F2 and F3 which have only been partially implemented.

Although, requirement F2 (The game must get progressively more difficult) has been met by ensuring more zombies are spawned in later waves and stages, we have not implemented fit criterion F2.2 which states that 'More difficult zombie types are spawned at later waves and stages. We decided not to do so as we believe that the increase in the size of waves provided a sufficient increase in difficulty and challenge as the game progressed. The inclusion of more difficult zombie types could make the game increasingly frustrating, leading to the loss of player motivation/interest.

Similarly, requirement F3 (There must be three different player types the user can choose to play as with different abilities) has been met by having three different character types with varying stats e.g speed, max health. Due to time constraints, we were unable to implement special abilities (F3.2). Despite this, we still believe that each character type is distinguishable from each other and provides the user with enough choice, challenge and enjoyment.