**Game Specification Form Student ID: ffgt86 Level 3**

|  |  |
| --- | --- |
| **Marking Criteria** | **Describe how your game matches the criteria** |
| **Game design (10%)** | |
| Game Goals: | Collect 11 Q-bits and deliver them to a portal |
| Game Type: | 2D top-down shooter |
| **Core development (30%)** | |
| Game scene (visual representation [2D, 2.5D or 3D], internal data structure): | 2D. The game is run from main, which loads the tilemap and assets. It first presents a menu, which can also be accessed by pressing ESC while playing.  Each sprite inherits from a Sprite class. Within this class are several subclasses, for the player, for zombies, for weapons, for items, and for effects. |
| Game flow / game progression (e.g., navigation, screen scrolling, levels): | Screen scrolls from top-down. Navigation is used WASD or the arrow keys; the mouse was found to be too unreliable with PyGame. One contiguous level is used, representing the Science Site. |
| Game interaction (e.g., action detection and response generation): | Zombies detect the player and pursue them in a detection radius. Zombies additionally detect nearby zombies and move away to prevent stacking. The game detects keyboard inputs to move the player, fire, reload, and pick up weapons. |
| Game object (e.g., use of sprite, 3D objects, animation, multimedia): | See below. The game is 2D. Some sprites (such as items) are animated. The game incorporates music and sound effects, controlled from the menu. |
| **Game mechanics (30%)** | |
| Game rules / logics: | 11 Q-bits must be collected, and delivered to a portal which spawns zombies, to win. Weapons have magazine capacities, though not ammunition. If a player or zombie takes too much damage, they are killed.  Health packs cannot be picked up if the player has full health. Weapons must be reloaded. |
| Game challenges: | It is difficult to complete this game, as there are so many zombies. |
| **Good use of game engine (15%)** | |
| Choice (pyGame, Unity): | PyGame |
| User input (keyboard, mouse, joystick): | User input is exclusively via keyboard. Mouse was found to be too unreliable with PyGame. |
| Game object interaction (e.g., event triggering, collision detection): | Collisions are detected between almost all sprites: between the player and zombies, the portal, items, weapons, and walls; and between the zombies and walls and the player. |
| Incorporate multimedia content: | The game incorporates music, sound effects, and animations. |
| Other features used (e.g., asset, incorporation of external libraries): | External libraries used include pytweening (for animations), and pytmx (for using tmx map files). Assets are mostly from https://www.kenney.nl/assets. |
| **Demonstrate creativity (15%)** | |
| Game economy (e.g., support to game type, game feedback, game difficulty): | Difficulty can be toggled and increases as the player collects Q-bits. It affects zombie spawn rates, speeds, and damage. |
| Advanced Interaction (e.g., game physics, object tracking, steering behaviour): | Little to report here. Grenades were considered but not implemented. |