Game Development and Multimedia Technologies Summative Assignment – Covid-19 Fighter in the UK

This coursework requires you to develop and implement a game fitting the "Covid-19 Fighter in the UK" theme. You are expected to apply a good range of knowledge and skills learnt from this module to complete your work. The implementation can be done by PyGame or Unity. You should choose a suitable game genre and design a story matching the required theme. You should also design game machinations to control game progression and game object abilities. Finally, you should demonstrate an effective use of multimedia content. Note that the focus of this coursework is game development with implementation rather than realistic graphics modeling.

The coursework contributes 100% of the module assessment. Submission date is the 28th January 2021 (2pm). The marking criteria is as follows. The level of achievement that you can achieve under each marking criteria is shown in the table at the end of this page.

- Complete game specification form and submit a 1-minute video demonstration (8%)
- Game design matching the "Covid-19 Fighter in the UK" theme (5%)
- Core development and implementation (30%)
- Game mechanics with machinations diagrams (30%)
- Good use of game engine (12%)
- Demonstrate creativity (15%)

The **game specification form** provides a detailed breakdown of what technical aspects you are expected to include for each marking criteria above. All aspects under each criteria are equally weighed within the criteria. You are required to fill in and return the form, providing brief descriptions of how your game meets the criteria. **No mark will be given to your coursework if you do not fill in the form.** You may refer lecture slides for the definition of the terms using in the form and the methods for implementing your game to meet each of the criteria.

Your submission should include the game specification form, your implementation with all source codes and resource files, a readme file showing instructions of how to run your game and what external resources you have adopted, and a 1-minute video showcasing reputable features of your game. You should compress all files into a single zip file and upload it to DUO for submission.

The level of achievement of each criteria is determined as follows. This is developed according to the marking and classification conventions published in pp.15-16 of the university core regulations.
(https://www.dur.ac.uk/resources/university.calendar/volumeii/2020.2021/coreregsug.pdf)

Level of achievement:	Range of Marks
No submission	0%
Inadequate or incomplete submission	0 - 40%
Satisfactory to Good (in terms of correctness and completeness)	40 - 60%

Very Good to Excellent (in terms of completeness and robustness)	60 – 80%
Outstanding to Perfect (in terms of completeness, robustness and complexity)	80 - 100%

COMP3567 Game Specification Form Student ID: 000745975

Marking Criteria	**Describe how your game matches the criteria (Description of each item is limited to 50 words)**
Game design matching the 'Covid-19 Fighter in the UK' theme (5%)	
Justification of the choice of game type:	I decided on a 2D Tower Defence style game. This is because I felt it would be straightforward to balance a strategy's game flow, progression and enjoyment across strictly defined levels. It allowed me to design a simple game appealing to all skill levels.
Game story:	The player is the last hope for a local village, where an increasing amount of virus particles are heading to and killing the population. Earn money by killing coronavirus particles and spend it wisely on lockdowns, vaccines and towers to survive three waves and save the town.
Core development and implementation (30%)	
Game scene (visual representation [2D, 2.5D or 3D], internal data structure):	The game is 2D, built with a tile-map. The tile-map is generated using a script accessing a text file storing values representing particular tile sprites. Each tile is attached to a tilescript, e.g. to assist pathfinding. The village structure is randomly generated, for a different scene each gameplay.
Game flow and how it is designed (e.g., navigation, screen scrolling, levels):	There are three levels/waves which increase in difficulty. Players place towers before each wave and watch the gameflow and implement vaccines/lockdowns during the wave. Coronavirus particles travel across the screen during each wave. The game can be paused and volume adjusted by clicking escape.
Game interaction (e.g., action detection and response generation):	The user interface is composed of buttons which use event triggers such as pointer enter and pointer exit to display tooltips. On onClick() events, towers can be selected and placed. Elements such as a vaccine 'health bar' appear for a limited time after clicking a button.

Game object (e.g., use of sprite, 3D objects, animation, multimedia):	Sprites include village buildings, tile sprites to create and divide the map, tower sprites and their pellets which can be fired. Animations show these exploding on enemy contact. Sprites are used for the virus which pulse and cripple upon death. There is background music and sounds for each coronavirus death.
Game mechanics development and implementation (30%)	
Main game rules / logics to control game progression, difficulty and end game conditions:	The game ends once all three waves are completed, or the village health is depleted. Difficulty increases through more enemies with increased health. Each tower upgrade costs increasingly more. To balance challenge and ability, each tower upgrade provides greater killing potential, through increased damage and special attacks.*
Control of game object abilities:	Towers have a projectile speed, damage and special attacks which can be improved upon upgrades. Coronavirus particles can pathfind around towers to the village. Their health is depleted by tower pellets. Stone towers can slow down particles, whereas fire can damage particles for an extended period of time.*
Good use of game engine (12%)	
Justification of the choice of game engine (pyGame, Unity) in terms of suitability of matching the theme and the expected target audience (game player):	Unity is a more powerful engine, which means greater potential for further complex advancements. There is also more flexibility and choice in terms of tile-based games, necessary for my game. The game would ideally be focussed towards mobile users, and Unity works well with mobile apps.
Types of user input supported (keyboard, mouse, joystick, etc.):	The input supported consists of keyboard and mouse. The mouse is used to click buttons on the user interface, and place towers. It is also used to navigate the pause and end game menus. The escape key is used to pause the game.
Types of game object interaction supported (e.g., event triggering, collision detection):	Event triggering is used when hovering over buttons, displaying a description tooltip, e.g. OnPointerEnter. OnClick functions are necessary to execute button functionality. Collision detection and rigid bodies are necessary, e.g. for detecting enemies in a tower's range. Necessary functionality is executed in turn.
Other game engine features used (e.g., asset, incorporation of external libraries):	None of the sprites used are my own - they are imported from asset libraries. For instance, the tile-pack. The grid component was key for implementing the tile map. I also made use of rigid bodies and collision detection, e.g. for

	recognising pellets hitting enemies and enemies reaching the village.
Demonstrate creativity (15%)	
Effective use of multimedia content:	I sourced an open sourced audio clip which I edited so it loops. This plays as background music. There is also a ‘splat’ sound effect used when an enemy dies. The volume for these can be adjusted in the options menu.
Advanced interaction implemented (e.g., game physics, object tracking, steering behaviour):	I implemented an A* path finding algorithm. Enemies spawn from a range of points, and have to navigate around towers to the village entry point. Although a 2D game, when elements are closer to the user, they will appear ‘on top’ of further elements.

***Note:** Your work must be done by yourself and comply with the university rules about plagiarism and collusion. (https://www.dur.ac.uk/learningandteaching.handbook/6/2/4/)

* Machination Diagram: