

C# Final assignment – VAMPIRE

Hand out date: -September 8-

Due date: -October 10 -

This assignment is the final assignment of the C# course. The objective is to create a vampire survivors-like game, it does not have to be line by line the same game, but the core gameplay loop of- Killing enemies, levelling up, upgrading character, Repeat- should be there.

Vampire survivors is a so called “rogue like”. Core to this is replay ability and the brutality of the game. You lose very quickly and very easily, but you can start playing again very quickly. In vampire survivors there’s even a time limit of 30 minutes before you finish the game. Since you are making a roguelike there does not have to be a win condition or a win state.

Core to vampire survivors is the massive hordes of enemies approaching you. And the player being able to kill them very fast. There are plenty of vampire survivor like games out there, so if you need inspiration, it is heavily recommended to try one of the existing games in this genre.

Remember, **you do not have to reinvent the wheel!** This is a programming assignment where you will implement a lot of features, please feel free to take a lot of inspiration from other games. It is easier to backwards engineer features from other games than designing them by yourself!

Create a vampire survivors like game with the following features –

G Requirements ->

- Player
 - Player Combat
 - Player can take damage & Die
 - Player can deal damage with attacks (combat mechanic is up to you)
 - Player Upgrades
 - Player can “level up” after killing a certain amount of enemies
 - Player can choose an upgrade after levelling up, minimum of 3 upgrades to choose from. And a minimum of 3 unique upgrades.
 - Player movement
 - Player can move around using WASD & Arrow keys.
 - Player movement is normalized
 - Player UI
 - Experience or indicator for level up is displayed in UI
 - Player health is displayed in UI
 - Camera
 - Camera follows the player in a top-down perspective
 - Player is always at the centre of the screen
- Enemies
 - Enemies spawn over time
 - Enemies move towards the player and attack
 - Minimum of 2 varieties of enemy ex,
 - Enemy that goes directly toward the player
 - Enemy that orbits the player and slowly moves closer

- Enemy that only runs at the player if the player is not looking in that direction
 - Enemy boss with a lot of health spawns and prohibits smaller enemies from spawning.
- Architecture
 - States
 - The game has a game over menu
 -
 - The game has a pause state
 - During this state none of the enemies move, and the player cannot move. You are not allowed to do `Time.timeScale = 0`; unless you have a reasonable explanation and explain why in the hand in.
 - The game has a upgrade state.
 - During this state the game is also paused and will only unpause once the player has chosen an upgrade.
 - The game has a main menu
 - A high score is displayed in the main menu. Example of high score is how many total enemies the player has killed
 - High score is persistent between sessions (High score does not reset when you close and open the game)
 - The game has a settings menu with all the following features
 - change game volume
 - change between Fullscreen / windowed
 - change VSync.
 - Progression
 - Enemies become more difficult as the game progresses example
 - More enemies spawn
 - Enemies have higher HP
 - Different enemy types spawn
- Juice
 - Main menu has a main menu song playing in background
 - While playing there is a game song playing in the background
 - Sound effect when the player levels up
 - Sound effect when the player dies
 - Sound effect when enemies die
 - Sound effect when the player attacks
 - Sound effect when the enemies attack
 - Particle system or death animation when the enemies are destroyed.

VG Requirements ->

- Architecture
 - Player high score is saved to Json with a user system
 - Player can input a username, and the high score will save to that username
 - There is a leaderboard in the main menu displaying the top high scores in order.
 - Enemies are spawned and destroyed using an object pool
 - Enemies are created using the factory and or builder pattern
 - Changing the states in the game is done using a generic State machine (not finite)
 - Use the singleton pattern.
 - Game runs at 60fps

- Demonstrate adaptability.
 - In the final hand in demonstrate how you do the following, adding a new enemy or upgrade to the game should not require restructuring of the entire game.
 - Creating a new enemy should have a smooth workflow
 - Creating a new upgrade should have a smooth workflow
- Juice
 - Smooth transition between states ex-
 - Animated
 - Fade in-out
 - Damage numbers appear on enemies when attacking them.
 - Feedback on gaining experience ex-
 - Small sound effect
 - UI shakes
 - Colour change on xp bar
 - One of the following –
 - Gibs (flying limbs when killing enemies)
 - Camera shake

Instructions for hand in –

Before hand in, remember to go through each feature to make sure they all work

Submit the following on omniway.

G

- A video with a max length of **15 minutes** showcasing all features. The video should also include your voiced commentary on how you have implemented each feature. It is important to evaluate each feature from the list of requirements. This video can either be sent with a google drive link or uploaded to YouTube as unlisted, you then submit a txt file with a link to the video.
- A document (min 1 page, max 5 page). This document should include –
 - 3 of your most challenging features to implement
 - What was the feature?
 - Why was it challenging?
 - How did you solve it?
 - Would you do it differently next time and how?
 - Structure of your work
 - In what order did you do your features?
 - Did you plan before you created features, or did you just iterate over time?
 - What tools/resources did you use?

VG

In addition to the G and VG requirements, and the instructions for hand in listed above.

Showcase workflow of creating a **new enemy** and a **new upgrade** for the game. This new enemy and upgrade should then be showcased as it is implemented into the already existing game. (The new enemy does not require animations or art, placeholder is fine).