Assignment four

# Introduction

In this assignment, you will continue to build on the project by adding new functionality. We will be adding simple game states so that we can have end conditions to the game, win or lose.

At the end of this assignment, you will have a completed simple game that you can either win or lose.

# Getting started

All required setups took place in Assignment 1. Check there if there seems to be something missing.

# Assignment four

## Part 1 (45%)

##### Part 1A

Our player needs health as well, so the game is not over after one misstep. Add a Health component to our player and set its value from the tuning file.

We also want to create a new component to represent an invulnerability period when the player is hit. Otherwise, the player would take damage every frame they are overlapping an enemy. This component should look very similar to the firing cooldown component, with only a cooldown to keep track of. In the Player’s update, decrement the invulnerability the same way you did fire delay.

##### Part 1B

With the new components in place, we can now check for collision between Player and Enemy entities. If this collision happens and the player doesn’t have the Invuln component, decrement the value of its health. Otherwise add an the Invuln component and set its cooldown value to something (I’m using 5 seconds). Print the Player’s current health to the console when damage is taken.

We will handle what happens when the player’s health reaches 0 in another part.

## Part 2 (55%)

##### Part 2a

Create a new tag component to represent a new state that can be added to the GameManager’s entity, GameOver. If this tag is present, then the game is over and all the systems in the GameManager should not run. We only skip these systems as we still want the rendering to continue.

##### Part 2B

Add system that checks all Player entities (in case you ever add multiplayer, for example). If all Player’s health is 0 or less, add the GameOver tag and print “You lose, game over” to the console.

##### Part 2C

Check a view of all entities tagged as Enemy and if that’s empty, then that means no more enemies exist in the game. Add the GameOver tag and print “You win, good job!” to the console.

# Summary

With this assignment done, you should have a simple game that can be won or lost. Congratulations!

Next steps will take place in Project and Portfolio 5. You will you the knowledge gained here to create a larger group-based project game. If you have extra time, it is a good idea to look over this project and clean up any areas you feel could use it. You will be able to use this project as a base on which to build your new game.

There are many things that could be added to this project to strengthen areas you want to gain more experience in. Any extras won’t be part of the submission, but will come in handy in the future project. Consider adding extras if you have the time. Maybe add texturing and updating the models. Create a new level in Blender. Maybe add new enemy types that have different models and behaviors.

# Resources

Documentation for the APIs used in this course are provided here. If you run into issues with any methods in your work, start here.

## ENTT

If you wish to deep dive or learn more about any given aspect of ENTT’s API, check out the ENTT wiki

<https://skypjack.github.io/entt/index.html>

## Gateware

We will be using this API occasionally throughout these assignments for simplicity’s sake. Gateware is a powerful cross-platform API often contributed to by students here at Full Sail just like you. (Designed for 3D Engine builders)

<https://gateware-development.gitlab.io/gcompiler/index.html> (Official Documentation)

*Tip: use the “--->” triple-dash operator on any Gateware proxy to have intellisense show you the actual arguments.*

# FAQ

Since this course is new, there are no questions that have appeared frequently (yet). So please, if you have questions, ask them so we can all grow.