**COMP 4300 Project Specification**

The final project for COMP4300 can be done in groups of up to 4 people, and I highly recommend 4 people for the group. I understand that students do not always enjoy working in groups for school projects, but the reality is this is how you will be working for the rest of your careers, and the ability to work with others in a reasonable manner is factored into the grade for the course.

If there are any problems with your group members not doing their fair share of the work, you must see me immediately so that we can work out a solution while there is still time to do so. My observation over the years is that 99% of the time this happens, it is due to simple miscommunication between group members and can be solved with a few emails. If you wait until the final week before the project is due to come to me with group-related issues, there is probably nothing that can be done to help at that point. Everyone in this class is an adult, and part of the grade for this group project is your ability to work well with others to accomplish a shared goal.

Despite the length and feature scope of this project, which can look pretty scary, most of it has actually been done already in the class assignments. You are allowed to copy any and all code done from your assignments as a starting point for the course project. In this way, almost every line of code that you write for Assignment 3 and 4 are also being written for the project as well.

**Project GitHub URL Submission**

As soon as you have formed a group for the project and created a GitHub repo, please submit your group GitHub repo URL to the Project - GitHub Repo URL assignment folder on D2L. This must be done before the due date of the Project Proposal. All due dates are specified in the folder name in repo root directory.

**Marking Scheme**

* **Project Proposal** (5%)
  + A proposal submitted in order to judge the scope of the project
* **Project Demo** (10%)
  + A short demo gameplay video as a milestone to judge your progress on the project
* **Project Code / Gameplay** (60%)
  + The final submission of your project's code and gameplay that will be graded by prof / TA
* **Project Report Video** (20%)
  + A record video presentation submitted which explains all details of the project
* **YouTube Game Trailer** (5%)
  + A short trailer for your game that you can use in your future portfolio

Important Note: This is a game programming course. The final project game is the most important part of the course being valued at 60% of the total course grade. Also, the course syllabus states that **you must pass the final project to pass the course**. You cannot get 100% in all the assignments, obtain 40% of the course grade, and then just submit a final project worth 10% to get a 50 in this class. You must submit a project which has the vast majority of the required functionality completed, there is no other way to pass this course. Every year, one or two groups take this warning lightly and put off doing the project until the last minute. The result of doing this will be you failing the course. Please take this project as seriously as any other final exam you have in any course.

I am giving you as much time as I can to work on the project: it is due on the final day of exams in order to give me just enough time to mark it before I am required to submit your grades. There will be no extensions of the due date and no allowance for late submissions for the project unless there is a serious medical emergency, and even then we will need to submit a formal request for deferral through the university. I repeat: treat this as if it were a final exam.

**Required Game Features / Gameplay Functionality**

This section details all of the required functionality for your game. You can use it as a checklist to mark your game's progress by putting an [x] inside the square brackets in markup.

**Game Overview**

* Must be implemented using ECS architecture in C++ using only the SFML / ImGui libraries
* You may use any course code already written for assignments as the bases for your project game
* Game Types Allowed: 2D Platformer, Top-Down Shooter, Action RPG, etc
  + If 2D platformer is chosen, it must be significantly different from Assignment 3
  + No assets or levels may be re-used from the class assignments
* Must contain at least 3 pre-built levels, and have a separate 'final boss' battle level
* Must contain a custom menu that allows the player to play the game, edit levels, or select options
* Must contain some sort of in-game menu (item selection, inventory, options, etc)
* Must contain a level editor that allows for loading, editing, and saving of game levels (see relevant section)
* Must contain a ‘game over’ screen indicating when the game has been finished
* All levels, player, and game configuration options must be defined in external text files
* All assets should be gathered or created by the project group members

**Game Scenes**

* Main menu scene that implements the main menu / options functionality
* Overworld map scene that allows for level selection / game progression (Super Mario World etc)
* Main gameplay scene that implements the game physics of the main gameplay mode
* Some sort of item inventory or in-game many scene that is used for a relevant function
* Level editor scene that implements level editor functionality
* A ‘game over’ scene with some sort of animation and game over / credits

**Required Gameplay and Mechanics**

Your game must contain all the following mechanics. You may copy / paste / edit this list for use in your proposal. Your final project submission must contain this list which each finished feature marked as completed.

* Collisions
  + Rectangular bounding box collisions between some entities
  + Some collision between player and level geometry (walls, tiles, etc)
* Movement
  + Your game must contain at least 2 'movement abilitites' with a cooldown or resource cost
  + Our Game's Movement Abilities:
    - Example 1: Double Jump
    - Example 2: Dash (with invulnerability)
    - Example 3: Wall Jump
    - Example 4: Jetpack (with fuel)
* Bullets / Weapons
  + At least 3 unique weapons that are usable by the player
  + Our Game's Weapons are:
    - Example 1: Boomerang (returns when thrown)
    - Example 2: Shotgun (several bullets fired in a spread)
    - Example 3: Sword (melee weapon)
  + Weapons must be swappable during gameplay via hotkey or ui element
  + At least one weapon must consume ammo obtainable during gameplay
* NPCs
  + At least 3 unique non-player characters in the game that act as enemies or allies
  + Must contain basic AI such as path-finding / shooting / patrolling / battling with the player
  + Each NPC has a unique weapon or method for interacting with the player
* Moving Tiles
  + Must include some part of the level which moves, such as platforms / elevators
* Hit Points / Damage
  + Player / enemies in the game should have hit points (life) and take damage / die
  + Invincibility frames must be implemented for all entities that take damage
  + Enemy HP levels must be displayed to the user in some way
* Status Effects
  + Must contain at least 3 separate status effects
  + Status effects are obtainable in some way (item, collision, ability, etc) which alter the gameplay for a limited amount of time
  + Our Game's Status Effects are:
    - Example 1: Speed Potion
    - Example 2: Quad Damage
    - Example 3: Invincibility Star
* Objects / Inventory
  + At least 3 different items should be obtainable during the game via interaction
  + UI should display inventory of these items, allow for their use after obtaining
  + Our Game's Items are:
    - Example 1: Health Pack (used to regain hp)
    - Example 2: Sprint Potion
    - Example 3: Teleport Scroll
* Ray Casting / Visibility
  + Ray casting calculations should be used in some form in the game for lighting AND visibility calculations
  + Our Game's Ray Casting Functionality are (one lighting, one visibility):
    - Example 1: Turret fires when player enters line of sight
    - Example 2: Player torch illuminates areas within line of sight
* Gravity / Acceleration
  + There must be some form of gravity / attractor in the game that applies acceleration to the player
* Camera / World View
  + Your game must use at least 2 different camera views in an interesting way
  + Our Game's Camera Views:
    - Example 1: Standard main camera view of gameplay
    - Example 2: Mini-map of local area that shows player position
* Pathfinding/Steering
  + Some entities in the game must exhibit non-trivial pathfinding and smooth steering behaviour
  + Our Game's Pathfinding/Steering:
    - Example 1: Heat seeking missile smooth steers toward target entity
    - Example 2: Path-finding algorithm used to guide NPC toward player
* Game Progression
  + Overworld map should somehow lock or unlock game progression based on levels completed
  + You must have the ability to save and load your game progress somehow to a file
* Shaders
  + At least 3 shaders must be used to alter entity appearance in a meaningful way
  + Our Game's Shaders:
    - Example 1: Entity turns grayscale when frozen
    - Example 2: Glowing green outline for radiation
    - Example 3: Alternating colors for Mario star-like effect
* Parallax
  + Must incorporate parallax via multiple background layers in some way in your game
  + Our Game's Parallax:
    - Example 1: Background layers used to indicate depth
* User Interface / HUD
  + Must have a user interface / HUD which displays information such as:
    - Player health
    - Weapons and Ammo
    - Game Progression
    - Status Effects / Timers
    - Other UI elements listed here
* Audio
  + Must have background music during gameplay / menu scenes
  + Must have sound effects that occur for events such as getting hit, item pickup, ui selection, etc
* Options
  + Must have an options menu which allows you to change the following settings:
  + Music Volume
  + Sound Effects
  + Game Difficulty - Normal, Easy (deal 2x, take 0.5x damage), Hard (deal 0.5x, take 2x damage)
  + Rebind main gameplay scene keys – for example moving left / right, jump, shoot, etc
* Assets
  + You cannot use any assets that were given out in the course (levels / animations / music)
  + You do not have to create assets from scratch, you can obtain them online
  + If you obtained assets online, please cite the sources of the assets
* Extras
  + 10% of the mark of the project is reserved for extra / new mechanics not specifically listed here.
  + Our Game's Extras:
    - Example 1: Really complicated lighting effect
    - Example 2: Currency system that allows you to buy from shops in-game
    - Example 3: Fancy NPC AI that goes above and beyond

**Level Editor**

* Your game must contain a level editor similar to the one found in the MegaManMaker / Super Mario Maker game
* All of your main gameplay levels must be able to be made from within the editor. Overworld map can be hard-coded
* This level editor should operate on a grid, similar to the one found in Assignment 3 / 4
* The level editor must be made mostly with the ImGui library.

Level Editor Required Functionality:

* Menu that allows you to select an existing level to edit
* Ability to select and place any Texture / Animation defined in the Assets file into the level
* Any parameters of specific entities must be editable via the level editor. For example, this can include:
  + Whether NPCs block vision, movement, or neither
  + The hit points / damage of NPCs in the level
  + The patrol points of moving tiles or NPCs