

# COMP 4770 – Required Game Features + Group Structure

Your game must have the following functionality, but it should probably have a little bit extra! When in doubt, use Mega Maker as an example. Replicating as much of the functionality of that game as possible is what you should aim for.

## Required Game Features

### 1. Game Overview

- 2D Platforming Game.
- Must contain several pre-built levels, and have a 'Game Over' final boss.
- Must contain 'overworld-like' level selection screen (similar to NES Mario 3)
- Style should be simple, 8-bit feel. But more complex if you can handle it.

### 2. Gameplay and Mechanics

- Gameplay
  - Similar to Mega Maker, must login to server to access the game
  - Objective should be to beat levels
  - Must include game saving of some sort (save / load to DB)
- Mechanics
  - Standard 2D platformer physics. Player is affected by gravity unless some sort of other effect nullifies it or it hits a solid object.
  - Player falls due to gravity until hitting solid ground, and stands on it. Player cannot move through walls or other 'solid' objects
  - Player can move left / right, jump, slide, shoot currently equipped weapon.
  - Objects exist in the game that can be interacted with. In simplest form, something like walking over a health pack restores health, or weapon energy (like mega man).
  - Falling onto spikes (part of level) and into holes should instantly kill player
  - Upon taking damage, player should have several 'invincibility frames' and not take damage during
  - Player and enemy weapons / projectiles should have custom scripts if/when they collide with entities in the game. (ex: entity takes damage from bullet)
  - Movement through the level should be smooth, unless for some cinematic effect.
- Game should have options like difficulty (easy/medium/hard) which multiply enemy HP / damage.
- User should be able to re-map keyboard keys to jump, shoot, etc. Controller support optional.
- Infinite health / ammo cheats should be included (helps with debugging)

### 2. Story, Setting and Character

- Write something for story, but it doesn't have to be Shakespeare.
- The game should have the main player, and several bosses with unique / complex behaviours.

### 3. Levels

- Overall structure of levels. How are they accessed, how are they completed?
- Level Design - Description of level design. This document does not need to include a description of all levels that will be in the game (we are making a maker after all), however, describe characteristics of a level so that a reader can get an idea of how one looks / plays.

#### **4. Interface**

- Game UI should show health, weapon status, etc.
- Player controls the game with keyboard / mouse by default, controller optional
- The game should have sound effects and music

#### **5. Artificial Intelligence**

- Basic scripted enemy AI behavior is expected (move, shoot, follow, etc.)

#### **6. Game Art**

- Game should have \*some\* sprites / art, but not professional. Would prefer 2D 8-bit style.

#### **7. Level Editor**

- Use MegaMaker as an example for your Level Editor

#### **8. Player Account**

- Player should be prompted with login screen, create account / log in to server
- Each player must have their own 'Player Profile Page'
- Player profile contains saved games / high scores / achievements etc

### **GROUP STRUCTURE**

The course will be divided into teams, each team consisting of 4 members (or 5, if odd number of students). Each project will be divided into the following 4 components:

- Main Interface (front-end login screen, game rendering, etc)
- Game Engine (game logic, physics, player movement, etc)
- Level Editor (all level editor logic, saving / loading of levels)
- Backend Server / Database (node.js server, mongodb database, etc)

Groups will assign each of the group members to cover TWO of these 4 components, such that no single component having less than two people assigned to it. For example, group members A, B, C, D might be assigned in the following ways:

- A) Interface + Game Engine
- B) Backend + Level Editor
- C) Interface + Level Editor
- D) Backend + Game Engine

Even though you will be working mostly on the two components you have chosen, you **MUST** have a working knowledge of the entire system, and be familiar with the code of the entire project.