# CS 3540 – MCA4 Game AI

In this assignment, you will build on your FPS game from MCA3, add intelligent NPCs to your game by implementing an FSM, and apply the NavMesh system for NPC navigation.

## Requirements

### NPC FSM (10pts)

1. Implement a functional FSM with **at least** the following states: Patrol, Chase, and Attack.
2. The FSM should be similar to the in-class example. For the Patrol state, the NPC should wander around some points in the game. The Chase state should be active when the player is at a certain distance and within the field of view of the NPC. The NPC should chase the player until it can attack the player (again specified by a certain distance).
3. When the NPC attacks and hits the player, it should cause some damage to the player. The amount of damage should be randomly determined.

### NPC Sight (5pts)

1. Implement a sense of sight for your NPC. This should be integrated into the FSM so that the NPC starts chasing the player when it can see the player based on the sense of sight you implement.

### NPC Navigation (5pts)

1. The movement of the NPC should be based on the NavMesh system. For this purpose, you will need to bake the NavMesh first. Then control the movement of the NPC using the NavMesh system.
2. In your level design, implement a moving platform (dynamic obstacle). When the player is on top of the platform, the player should move with the platform.

### MCA 3 Functionality (all components required; otherwise penalty)

1. The game should build on MCA3 and should have the same functionality (health, projectiles, game states, levels, etc.). You can add more capabilities/mechanics if you like. If you didn’t complete MCA3, you will first need to develop an FPS game meeting the requirements specified in MCA3.

### Overall Game Polish (5pts)

20% percent of your grade will come from how polished your game is. Game polish refers to the look and feel of the game and reflects how much effort you’ve put into making the best game possible given the requirements. For the most part, this is a subjective quality of your game; you’ll know it when you see it. Here is the scale that will be used to assess the polish of your game:

**Excellent:** presents a new game environment not used during in-class demos but is able to use the same mechanics; goes above and beyond the requirements to improve game mechanics; provides aesthetically pleasing look; adds extras when applicable to make the game interesting/innovative; uses realistic behaviors and appropriate graphics instead of simple shapes.

**Satisfactory:** uses the same game theme as the one used during in-class demos; meets the requirements; graphics look good but can be more polished; no extra effort to make the most interesting/innovative/different game; uses simple shapes sometimes to meet the requirements

**Half-baked:** one or more requirements is missing; problematic game mechanics/weird controls/behaviors; graphics/colors/assets don’t look very good; no extras; gets away with simple shapes for the most part; doesn’t convey an effort to produce the best game possible

**Dull:** bare minimums in all aspects; multiple problems with mechanics/graphics/assets. No effort to make the game look and feel good at all.

## Submission Requirements

1. Complete the last page of this document.
2. Publish your game for WebGL.
3. Upload your WebGL build to Unity Play, itch.io, or another website (similar to MCA1). **Make sure it’s playable**.
4. You should also submit a zipped folder of your **Unity Project** (**not** the WebGL build). Please delete redundant assets as they will increase file size. When zipping your Unity Project, include the following folders **only**:

Text

Description automatically generated

Simply, go to your project folder (you can open it while in Unity). Then select these three folders and zip them. This is different from zipping the entire Unity folder (Unity files aren’t included in this method, which will mean smaller files). This folder isn’t expected to be very large (at 20-50 MBs at most). Use the following naming convention for the file: MCA4\_LastnameFirstname

This will give you a compressed file with the zip extension. Because we are using GradeScope for handling submissions, there is a little catch: it automatically unzips zipped folders! To work around this limitation, we ask that you manually change the extension of the zipped file to .rar (instead of .zip). It will still preserve your project files, but the zipped folder won’t be unpacked into your submission on GradeScope. This isn’t ideal, of course. However, there is no way to disable it on GradeScope. Be sure that you change the extension not the file name. If you see the contents of your compressed folder unpacked into your GradeScope submission, that’s a clear indication that the file extension wasn’t properly changed. As CS students, you should have found out how to change the extension of a file on your computer by now.

1. **Attach all your script files to your submission, similar to MCA1. The number of script files will depend on how you structure your game. All must be attached separately outside the zipped folder! Failing to do this may result in a grade of 0.**
2. Complete the last page of this document and submit your High-Concept Document as a PDF. Delete all the prompts in this document when submitting it. Only the High-Concept Document part should be submitted.
3. In summary, your submission should include the zip folder (renamed properly to .rar) for your Unity Project, separate script files, PDF for High-Concept Document, and the link to the playable game (available in the PDF).
4. Submissions will be handled through GradeScope, so you may need to follow along with their custom instructions. Still, you must ensure that you submit all required files.

**Failing to meet submission requirements will result in up to 25% penalty above and beyond other point deductions.**

## High-Concept Document (required)

Before you develop your game, take some time to conceptualize your vision for the game by answering the following questions. You can copy and paste from MCA3, if nothing has changed.

### Concept

What is the core concept of your game? What is the theme?

### Goal and Description

What is the goal of the player? What is the game challenge (i.e., collect all gems before the time is up?)

### Core Mechanics

What are the core game mechanics implemented in your game?

### Controls and UI

How do we control the player? Are there other actions other than moving the player around the world?

What are the UI elements implemented in your game? What do they do/mean (e.g., the centered textbox keeps track of time)?

### Levels

Provide a description of what is different in each level of your game here.

### **Link to your Game**

Provide the link to your playable game here (in case GradeScope doesn’t show the comments field).