15-466/15-666 Computer Game Programming Fall 2015

Project 2. DUE: Monday, October 26, 11:59PM

In this assignment, you will create a pathfinding algorithm based on your X66 game.

This assignment has **no starter code**. You will create a Unity project from scratch, populate it with objects, and script those objects yourself. You are free to use any assets you wish to in your project, as long as they are able to perform the tasks required by the assignment. If you use assets from the internet, do try to respect the licenses they come with, so you can showcase your assignment without worrying about licensing restrictions. Because there is no starter code, please make your code easy to read and well-annotated.

If you need assistance with setting up a Unity project please drop by office hours or email Pasan (pjulsaks@andrew.cmu.edu).

1. Requirements

You will need to create a scene that has both static and moving obstacles in it. You will need to add the following elements:

• A player controlled character capable of moving around the environment. You may use Unity's default "ThirdPersonCharacterController" class if you wish.

• **At least 10 characters** chasing the player. These characters must: a) Use a form of pathfinding to get a path to the player b) Use some path following behavior to move towards the player.

• For the obstacles, please make sure that

1. The scene should be big enough, the characters should take at least around 20 seconds to reach from one end to another.
2. There is some kind of "maze" in the scene. To put it simply, a setting that PathFinding is needed, i.e. you can't just run your arrival behavior from Project 1 and expect your characters to reach the goal.

• Your game must run at full speed while running pathfinding for your 10 characters. That means no stuttering, framerate drops, etc.

2. Pathfinding Details

You will need to construct a navigation grid of some sort for your game. You will then need to use a graph search algorithm to construct the path for your characters to follow.

Remember that all objects that you want to use during the game need to be a GameObject. Your grid will likely be a game object that is accessed by your characters during the game.

For debugging your grid, you may find it useful to look at Unity's Gizmos interface. This allows you to draw additional information in Unity's viewer to visualize your grid and paths through this grid. This is done by creating a script as normal and overriding the "OnDrawGizmos" function. There is a basic Gizmo included

in the starter code that draws a white box around the object the script is attached to. Look at that and see http://docs.unity3d.com/Documentation/ScriptReference/Gizmos.html for more information.

3. Extra Credit

3D Planning

Create planning algorithm that allows for third dimension. This can be in form of buildings, space travel, etc. In order to get credit for this, the setting would require you to have 3D planning (just a different in height of the terrain does not count). In other words, your setting can allow one agent to be on top of another (like in the building, etc.)

Planning with Time

Incorporate time in the path planning process to account for moving obstacles. Hacks such as extending the bounding box of moving objects does not count for this. Time must be a bona fide dimension in your planning process like X and Y on your grid are.

4D Planning

Combination of above planning algorithms will get you another extra credit. Please note that one main grading criteria is efficiency i.e. your game should be able to run at full speed to get full credit.

Prefer Path

Setup the environment that clearly demonstrates that the planner makes choices trading off costs of running different behaviors such as jumping vs. walking. In order to get credit for this, make sure that in your setting, character has a “choice” (i.e., picking a path that involves jumping or a path that doesn’t), and it will make the choice based on the cost function. Please make it so that some of your characters prefer one choice, and the others prefer another. However, make sure that your characters can still choose the less-preferred path if there’s no other choice.

Collision Avoidance

Modify your setting such that there’s a type of gameObject that your characters do not include in their planning, but will avoid when they are going to run into it. You can do this by creating a tag for your planning algorithm to ignore, but still your character could detect (e.g. via RayCast).

Additional features

As per the first project, you may add any additional interesting features as you see fit, such as particle effects or a UI. Make sure you finish the required components of the project before working on this.

4. Documentation

With your project, please hand in a document that explains the approach you took and how you implemented it, and also any extra features you may have added. Also include details of how to properly run your project and details of options and settings you’ve added in your project.

5. Video

In addition to your write-up please include a short video of your game in action. This is a required part of your handin materials. The video will give you something to show off later, in demo reels for instance, and helps resolve confusion while grading. The video can only help you during grading. Please use a screen recording program to make your video. Videos taken from your phone make it difficult to see what's going on in your game. To capture video from Unity, you can use whatever software you want. For windows, this includes programs such as FRAPS or CamStudio and for Mac this includes Quicktime screen capture. If you're having problems, email the TA or come to office hours.

6. Grading Criteria

Since there are different number of people in each group, the requirement will be a bit different. For 1-person team, in order to get full mark (100%) you need to get 100 points, for 2-person team 120 points and for 3-person team 140 points. To be specific:

1-person team's total score will be equal to the number of points earned

2-person team's total score will be equal to the number of points earned multiplied by 0.833 (=1/1.2)

3-person team's total score will be equal to the number of points earned multiplied by 0.712 (=1/1.4)

Your submission will be graded on these criteria:

- (10%) Realism of Motion. No oscillations, collisions, jitter, etc.
- (60%) Correct Implementation of Path Planning. A working version of A*, or other method. No planning problems with local minima, meaning characters do not get stuck in dead ends or corner cases.
- (20%) Efficiency. Your AI characters should get to their objective as quickly as possible while still running the game at full speed.
- (5%) Documentation
- (5%) Video Included.

Extra Credit

In order to get full point for each category, your scene should be set up (or specify in detail in the report how to run the scene) so that the required behavior can be shown effectively

- (30%) 3D Planning
- (30%) Planning with Time
- (20%) 4D Planning
- (10%) Prefer Path
- (10%) Collision Avoidance
- (10%) Additional Feature