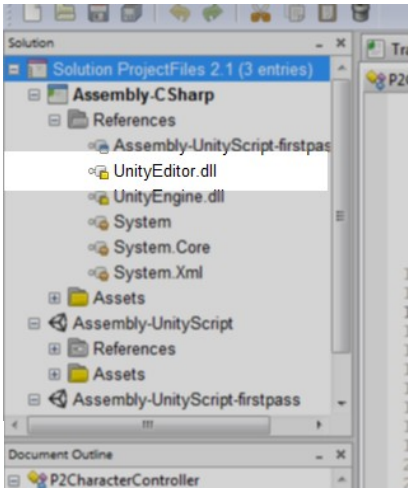
15-466 Computer Game Programming
Fall 2011
Project 2

For this assignment, you will be extending Project 1 to include path planning.
If you would like to update to Unity 3.4.1, you may, however it seems there is a problem that crops up sometimes with 3.4.1 projects where compilation fails with this error: “BCE0021 Namespace 'UnityEditor' not found, maybe you forgot to add an assembly reference.”
It seems counter-intuitive, but you can fix this by removing the “UnityEditor.dll” references in MonoDevelop. You will have to do this every time you re-open Unity or MonoDevelop.



Remove this from all both UnityScript assemblies to fix 3.4.1 compilation bug.

1.) Chasers

We have provided a Unity project you must use as a framework for your code. It includes a couple of simple environments, a 3d character controllable by you (the “player” of the game), 10 computer-controlled characters, and a framework for switching environments and resetting the simulation (essentially the same thing from Project 1). A download link will be emailed and will be available on Max's class website.

Look in “BehaviorScript.cs”. It's much like the one from Project 1, but the only behavior in the switch is now “Chase”. This is all that will be used for grading. Feel free to copy in any code you added for your Project 1 stuff. It may be useful here.

Look at “FrameSetup.cs”. Here you can put some code that is guaranteed to run before any of the “BehaviorScript.cs” files each frame. You might calculate some stuff here that's shared by all of the characters.

If you add anything *outside* of the “BehaviorScript.cs” or “FrameSetup.cs” scripts, please make note of your changes in your submission. It also might not be a bad idea to talk to John (drake@seas.upenn.edu) if you are thinking of changing something outside of those scripts. I do not think you should have to.

We have provided for you, at the top of "BehaviorScript.cs", a variable name “chaseTarget”. This will refer to the player's character. Your agents should attempt to catch this other character.

Available in “FrameSetup.cs” are the variables “trafficControl” and “aabbs”.

“trafficControl” is a reference to the instance of “TrafficControl.cs” in the scene. It manages all of the cars that move around.
trafficControl.carSpeed – is a float and describes the speed of all cars (it is fixed).
trafficControl.randomStartTimePerturbationInSeconds – stores how many seconds, maximum, a

vehicle may delay before appearing in-world. Set it to 0 in the Inspector pane in Unity (find and select the "Traffic" GameObject in the Hierarchy view first) if you would like vehicles to appear in a more consistent fashion.

TrafficControl.minimumCarSpacing – is a float that determines the minimum distance between car centers.

As mentioned, change those values by adjusting them in the Inspector pane for "Traffic"

"aabbs" stores a reference to the GameObject associated with each axis-aligned bounding box (AABB) in the scene. These are all you have to worry about when considering which parts of the world a character cannot occupy. Forget about Project 1 "obstacles" (which were all circular/cylindrical). AABBs are all rectangular. Read their "transform.position" and "transform.lossyScale" values to determine their extents. They are axis-aligned and rectangular so that they might be easy to rasterize to a grid. You can assume the bounds of the part of the world your chaser characters will need to navigate are: X -40 to 40, Z 10 to -70 (Z is up/down... worlds will all be flat, so ignore Z).

You can visualize all AABBs in the scene by selecting "FrameSetup Script" in the Hierarchy view under "Level Configuration". Then, in the Inspector pane, check the "Show AABBMarkers" check box. This will render a purple grid over all AABBs in the scene. I scale them all up by a factor of 1.02, so they stick out a little bit from the AABBs they represent. Uncheck the "Show AABBMarkers" check box to hide them.

2.) Documentation

Explain in a few short paragraphs what approaches you took for path planning and path following. Pseudocode is not good enough; please use English. Please put this in a separate text file in your submission, rather than just in comments in your code.

Also, if you changed anything outside of "BehaviorScript.cs" or "FrameSetup.cs", please let us know so that we do not inadvertently break your project.

3.) Video

Use a screen-recording program like CamStudio or Fraps (my favorite; works the best) to record a video of your game in action. Show yourself playing the game while your chaser agents pursue. It should be at least a minute long. **Unlike Project 1, the video is <u>required</u>**.

4.) Submission

Submit your entire Unity project and video by compressing it into a .zip or .rar archive. .tar.gz/.tgz is fine too.

It will be too large to email it to John, so we are working on setting up something else. Look for an email in the next few days about submission for Project 2.

The deadline will be sent in an email.

Grading Criteria

Your submission will be graded on these criteria:

- realism of the motion (e.g., no instantaneous velocity switching, no oscillations, no collisions, no slowdowns due to computationally-intensive operations)
- no path planning problems with "local minima". That is, if the target is on the other side of a dead end, your chasers don't just continue running into the dead end. Instead, they should turn around and find a way around.
- Efficiency of catching the target. If you get to a road with moving vehicles, your chaser agents should not merely wait for a clear path; they should be daring and cross when they can.
- concise/clear documentation
- inclusion of a video demonstrating your project

Appendix: The Provided Unity Project

We will try to explain in some more detail how the project is initially laid out and try to point out some tips and limitations here. If you have any doubts, please email John (drake@seas.upenn.edu).

Scenes

It's a lot like Project 1. If you're confused by something, please just ask.

Virtual Character

The chasers are set up just like in Project 1. BehaviorScript has changed, but not much. There's just the new "chaseTarget" and "frameSetupScript" references. I didn't make the player character a child of another GameObject like I did with the P1 characters. You can directly read the chaseTarget's position with "chaseTarget.transform.position".

The "chaseTarget" player character is controlled by keyboard input when you run the project. It's like the frog in "Frogger." Hold your [Shift] key to run.

Creating a Custom Level/Environment

Copy one of the existing scenes which are not "Initial Scene" and give it a new name. Leave the Behavior object settings on the chasers at "Chase".

Remove the "Stuff that won't get destroyed as levels are loaded" tree from your new level.
Edit the stuff in "Environment". Anything that is a "wall" or "obstacle" in your world should be given the "AABBObstacle" tag and should have rotation settings of 0,0,0. If you must rotate some geometry, do NOT give it the AABBObstacle tag and instead create another, invisible object with 0,0,0 rotation to serve as the AABB for that rotated geometry. Turn on AABB markers as described above to see how your bounding boxes look and where they are placed. Giving them 0 rotations simplifies the work you have to do when rasterizing AABBs onto a grid (should you choose to do that).

If you want to change the number of traffic lanes, copy or delete existing traffic lanes. The direction of traffic travel is determined by the positions of the Start and End objects which are children of each traffic lane. I would suggest only making traffic travel the same way it goes now (along the X axis). You can change the number of cars which can occupy a single lane by copying or deleting the “P2 Car” objects which are children of a lane's “Vehicles” object.

If you try to run the game, it will not automatically recognize your new level. To get it to show up in the level load list that appears on the game's GUI when the “Show Load Buttons” button is clicked, open your new custom scene and then click on “File->Build Settings”. On the window which pops up, click the “Add Current” button or drag your scene from the Project pane onto the list. You can reorder the levels by dragging them around in the list. Keep “Initial Scene” at the top of the list.

Miscellaneous Notes

To get FrameSetup and TrafficControl to run before BehaviorScript, I used the script execution order feature in Unity. Since I haven't shown this to any of you yet, I'll describe where it can be found here: Click on one of the script assets in the Project view. Look in the Inspector view for the button “Execution Order...” and click it. Here you can see and drag/drop to reorder scripts. Any script not explicitly in the list gets run at “Default Time” in arbitrary order. You can drag scripts to happen after “Default Time” too. You don't need to mess with any of this for this assignment, but I wanted to point it out since it is very useful for exactly what I wanted FrameSetup to do: happen before other things, guaranteed, every frame. Without this custom ordering, Unity picks an arbitrary order to run all the scripts.