Programming Assignment 2: Elvis vs. Grannies

Handed out: Tue, Apr 11. Due: Wed, Apr 26, 11:59pm. Late policy: up to 6 hours late: 5% of the total; up to 24 hours late: 10%, and then 20% for each additional 24 hours. The submission procedure will be the same as in the first programming assignment (see below).

Overview: The objective of this assignment is to learn more about Unity through the design of a primitive game that involves controlling the movement of characters through the use of Unity animator controllers and navigation meshes. The game consists of the following required elements:

Scene: The scene consists of Elvis’s groovy floating mansion in the sky, which consists of a number of shiny golden boxes with a few pieces of purple velour furniture (see Figs. 1 and 2).

Elvis: The player character is a humanoid character, Elvis, who navigates about the scene using a click-to-move approach. The player clicks the mouse over desired point on the ground platform, and Elvis moves to this point. (This is done through a combination of Unity’s navigation mesh and raycasting. Note that I explicitly want you to implement this form of navigation, as opposed to WASD or arrow-based inputs.)

You can use any humanoid model you want in place of Elvis, but your model should support at least two animations, an idle animation and a moving animation. Our player model is an Elvis impersonator called “Big Vegas” and was downloaded from the Mixamo Store, a repository that contains a number of character models and animations. I also downloaded from there two simple animations, an in-place jogging animation and an idle animation. (These will be made available to you if you want to use them. See the Projects section of the class web page.)
Grannies: Elvis is being chased by a number of non-player characters, called grannies. The grannies are also humanoid characters. They spawn at regular intervals. (In our implementation, there are three grannies. The first one spawns 3 seconds into the game and the other two spawn at 5-second intervals after this. You are allowed to modify these values, but make the number small enough that the grader can easily win your game.) The grannies start in an idle state. Whenever Elvis comes within a given distance threshold of a granny (we used 5 units), the granny starts chasing him. (If Elvis manages to escape beyond this distance, the granny returns to the idle state.)

If any granny catches up and collides with Elvis, then the game is lost. While each granny moves fast enough to make the game challenging (in our implementation the granny and Elvis have the same maximum speeds), grannies turn very slowly. The player wins by navigating near the edge of the platform and then stepping to the side at the last second, sending the slow-turning granny over the edge of the platform.

The granny character should support at least three animations: an idle animation, a moving animation, and a falling animation. Even though the grannies should not be able to the obstacles, you probably do not want your grannies using the navigation mesh, since they must have the ability to walk over the edge off. Their movement can be handled either explicitly by scripts or through the Unity physics engine. Raycasting can be use to determine whether grannny has stepped off the platform.

Our granny model is called “Sporty Granny” and was also downloaded from from the Mixamo store along with her animations.

Camera Control: The camera should move with Elvis. In our implementation, we chose to have the camera facing Elvis diagonally, since this way you can see his face. (The Holistic3d tutorial cited below describes a simple method for setting up this sort of camera placement within the Unity editor.) You may place your camera wherever you like, but it should move with the player character.

If implemented correctly, the above elements are worth 80% of the grade. For full credit, the following additional elements should also be implemented.

Entry/Exit Screens: (5%) Rather than simply terminating the game, an informative message should be displayed to the player indicating whether the game is won or lost. There should also be an introduction scene that explains the game’s objectives and inputs. (The introduction screen is not in our basic implementation.)

Pause/Rerestart/Quit: (10%) It should be possible to pause the action (and all the animations), restart, or quit your game at any time. (Pause is not supported in our basic implementation.)

Skybox: (5%) Elvis’s floating mansion looks rather unimpressive under the default Unity skybox. I downloaded a prettier skybox, by MentalDreamAssets from the Unity Asset Store (see Fig. 2). You can use ours, which will be made available, our obtain your own.

As always, you are welcome to experiment with your own models, animations, and game semantics. The main elements (navigation meshes, animations and blending, click-to-move) must be present in some form. Do not worry too much about minor flaws in motion and
collision detection. Our main interest is that the animations are smooth, natural, and occur promptly when requested, and that the game is easy to play and test.

**Online Resources and Tutorials:** There are a number of online resources that were helpful in designing our implementation:

- [https://unity3d.com/learn/tutorials/topics/navigation/navmesh-agent](https://unity3d.com/learn/tutorials/topics/navigation/navmesh-agent) The Unity video tutorial on NavMeshes. Also check out the videos under the section “Live Sessions on Navigation”
- [https://docs.unity3d.com/Manual/Navigation.html](https://docs.unity3d.com/Manual/Navigation.html) The Navigation section from the Unity manual. Check out the subsection on “Navigation How-Tos” especially “Moving an Agent to a Position Clicked by the Mouse” and “Coupling Animation and Navigation.” (Note that with the simple animations that I used, I had some difficulty getting the method described in “Coupling Animation and Navigation” to work
- [https://www.youtube.com/watch?v=...rest of link omitted](https://www.youtube.com/watch?v=...rest of link omitted) A tutorial by Holistic3d entitled “Mechanim & Mixamo in Unity 5”. This explains how to create and download the Mixamo models and animations that we used in our implementation.

**Our Implementation:** We have implemented the programming assignment. The files can be found on the [CMSC 425 Projects page](http://example.com). We will also create a file containing a lot of technical information about our implementation, which you may use if you like.

**Final Submission:** We will use the same procedure that we did for the first programming assignment (see the Submission Instructions from the Piazza post on Feb 5, 2017). Send an email to Alejandro with a link to a repository that contains a zip file with your submission. Also, remember to remove your `Library` and `Temp` directories, in order to keep the size small.

**Programming Style:** We will be reading your code to see that you implemented everything in a reasonable manner. Although programming style is not an explicit part of your grade, we reserve the right to deduct points for programs that are so poorly documented or organized that the TA cannot figure out how your program is working.
Optional Elements for Extra Credit: You may add additional features to your game for the purposes of extra-credit points. (See the syllabus regarding extra-credit points.) Please explain any additional features are in your Readme.txt file. The number of points of extra-credit credit will be left to the discretion of the TA.

External Resources: If you make use of any external resources in your program (or things that you developed prior to this class), even if you modified them, you must credit them in your Readme.txt file. Failing to do so will be considered an act of plagiarism. If you are unsure, check with me.