Milestone 1 – Character Animation

This is an *INDIVIDUAL* assignment.

***** SPECIAL NOTE *****
Please remove all unnecessary assets from the projects you are submitting.

Due Date

September 9, 2016 @ 11:55 PM

Late Policy

Starting with Milestone 1, we will use a 2^(*n*+1) late policy point penalty, where *n* is the number of days late. For example, 1 day late is a reduction of 2^(1+1) or minus 4 points. Assignments are considered late if submitted any amount of time beyond the due date/time. Each 24 hours late beyond the due date determines the number of days late. More examples: 2 days late is minus 8, 3 days late is minus 16, 4 days late is minus 32, 5 days late is minus 64 (no assignments accepted any later). Note that this late policy does not apply to assignments that involve in-class presentation, and is subject to change for any specific assignment.

Description

For this assignment you will be creating a 3D humanoid character that can walk and run around an environment.

You will be obtaining (or creating) a rigged 3D human character model of your choosing, applying a variety of locomotion animations with root motion, and combining these assets with a Mecanim animation controller (Unity's state-based animation blending system). This character will transition smoothly between walking and running according to user input. You will also configure an appropriate chase camera (or write your own). Lastly, you will find an appropriate demonstration level for your character to move around in (or create your own).

Mechanim is Unity3d's animation system. It combines Blend Trees and Finite State Machines (FSMs) to support creation of fluid animations for characters. Mechanim can also be leveraged to work with non-humanoid objects in the world and even the GUI system.

Requirements

For this assignment you will need to complete the following requirements:

Obtain (or create) a rigged 3D character model that you feel might be useful in the future (e.g. upcoming assignments). Even a character that will not be player-controlled later could be useful for a non-player character (NPC). Note that you cannot use any of the models from the Unity tutorials or any 1st or 3rd party source that provides *a pre-made Unity mechanim controller*. When in doubt, contact your TA!
10 pts

Utilize a chase camera. Your chase camera can come from the Unity standard assets, tutorials, etc. Or write your own. In either case, make sure you configure it so that it works appropriately with your character. Please document your source in your submission readme file.
5 pts

Utilize an appropriate level. Your level you use should at minimum provide a variety of terrain to traverse, including obstacles that block the character's path, or require climbing to navigate. Specifically, your level should have hills or ramps of various angles.
5 pts

Create a Mechanim Animation Controller for the model and animations that:

- Contains at least 2 floating point input parameters defined for the controller (e.g. forward [0.0,1.0] or [-1.0,1.0], left/right [-1.0,1.0]). These parameters represent continuous analog-style input values. (**5 pts)**
- You must also have animations with root motion for the following (at minimum):
    - Walk forward
    - Run forward
    - Run left forward (maximum turn rate)
    - Run right forward (maximum turn rate)
    - Walk left forward (maximum turn rate)
    - Walk right forward (maximum turn rate)
    - Idle (standing in place)

  (10 pts)
- From the default/idle animation state, the character animation transitions to (and possibly through) various animations using the values of the mecanim parameters discussed above. You must implement a combination of animation states and blend trees to support the locomotion speeds from walking to full run. Also, you must support various turn angles from no turn (straight ahead) up to full turn. **(30 pts)**
- Position translation of the avatar must occur via root motion (animation of relative position). **(5 pts)**

Implement a keyboard input testing script that sets the parameters you use in the animation controller based on keyboard inputs from the player. For demonstration purposes, use the keyboard numerals to set different global speeds and use keys W,A,S,D,Q,E to move. This will create a standardized pseudo-analog way for the graders to assess your character control. As is, this approach is a pretty cumbersome way to control a character. It is only in place to make grading easier.

You may want to additionally support video game controllers such as the PS4 controller, but that isn't required for this assignment.

<u>Test control details:</u>
"1" sets slowest speed (walking speed). "2" through "9" set progressively higher speeds (blend between walking and running at appropriate ratio). "0" is 100% full speed (full run).

Direction controls are defined as:
W-forward
S-backward (not required except for extra credit)
A-hard left turn
D-hard right turn
Q-light left turn
E-light right turn

If the player presses W (forward) then the character should walk/run forward based on the current speed setting (set by numerals). If the player presses W+A then the character walk/run forward and turn left at the maximum turn rate. W+Q will result in the character walking/running forward while turning slightly left. You don't need to support turning in place (e.g. turning while **not** walking/running forward) unless you are attempting the extra credit.
10 pts

The overall aesthetics of your character control should meet the following expectations:

- Demonstrate smooth transitions between all animations and animations states
- Character's feet should appear to land with every step across all blend ratios
- No visible moon walking or feet sliding along the ground
- No teleporting, glitching, etc., that breaks the illusion of a human moving naturally in space

20 pts

Extra Credit Opportunities:

Do one of the options below for **UP TO** 5 points. Do two out of three of the options below for **UP TO** 10 points. Be sure your readme clearly documents that you have completed extra credit and want the grader to assess. Actual credit awarded is determined by the grader, specifically determining if the spirit of the extra credit task is met and the interaction and aesthetic requirements are met.

1.) Implement jump that works from arbitrary heights as well as falling from arbitrary heights (e.g. walk off ledge without pressing jump) both with simulated gravity acceleration. Hint: You will need to abandon the use of root motion while in these animation states in order to work with differing heights. Please use spacebar for jump. Also, make sure your map provides the appropriate test environment.

2.) Add blended root motion based animation character control for backwards walking and backwards running. Also, add left/right turn-in-place animations such that the player can press and release the turn buttons to pick any orientation desired without the character walking away from the point he/she is standing.

3.) Add an environment-specific animation such as pulling up and over an overhead ledge or pressing a button on a wall, etc. Please email instructor(s)/TA(s) to confirm your choice is appropriate.

Tips:

- Pay careful attention to any tutorial instructions you follow. Often if you miss one little step, you'll have a lot of trouble (e.g. make sure to select the humanoid option in Unity when using Mixamo characters and animations).

Resources:

- Setting up the model in Unity
- Scripting and Mecanim with animations
- Mixamo's Animation Overview
- Mixamo's Advanced Anim Tutorial
- Character Animation – (old Unity 4 tutorial, but still useful) https://www.youtube.com/watch?v=Xx21y9eJq1U
- Unity Animations From Blender Rigify - http://docs.unity3d.com/Manual/BlenderAndRigify.html
- More Blender: http://zakjr.com/blog/blender-to-unity-workflow-part-1
- Adobe Mixamo: https://www.mixamo.com
- Blender - https://www.blender.org/
- Autodesk Education Free Software - http://www.autodesk.com/education/free-software/all

Tips:

From Amy: (some of her links are above too)

A few pointers on using Mixamo with Mecanim:

- Most/all of the pre-made models are humanoid type so they should be able to use the same animations. There's an option to download packages but you can pick individual animations from the library and add them to your Unity project, then apply them to the model. Just make sure the model and anims all the same type: Generic or Humanoid
- For multiple animations on one character, under the Animation object's Rig tab, be sure to set the Source to the character's avatar you initially set. All the animations for a character can share the same avatar which determines the bones and mesh, so it'll save your file size if you just have all the animations look at one model avatar instead of each animation having and making its own instance of a model.

There's plenty more tutorials besides these…so have a look and see what works for you!

Submission:

You should submit a 7ZIP/ZIP file of your Unity project directory via t-square. **Please clean the project directory to remove unused assets, intermediate and final build files, etc., to minimize the file size and make it easier for the TA to understand.**

The submissions should follow these guidelines:

a) Your name should appear on the HUD of your game when it is running.
b) ZIP file name: <lastName_firstInitial>_m1.zip
c) A /build/ directory should contain either a Windows or OSX build of your game.
d) Readme file should be in the top level directory: < lastName_firstInitial >_m1_readme.txt and should contain the following
    i. Full name, email, and prism account name
    ii. Detail which requirements you have completed, which are incomplete, and which are buggy (be specific)
    iii. Detail any and all resources that were acquired outside of class and what it is being used for(e.g. Asset Bundles downloaded from the Asset Store for double sided cutout shaders, or this file was found on the internet has link http://example.com/test and does the orbit camera tracking).
    iv. Detail any special install instructions the grader will need to be aware of for building and running your code, including specifying whether your developed and tested on Windows or OSX
    v. Detail exact steps grader should take to demonstrate that your game meets assignment requirements.
    vi. Which scene file is the main file that should be opened first in Unity
e) Complete Unity project (any file you acquired externally should be attributed with the appropriate source information)

Submission total: (**<u>up to 20 points deducted</u>** by grader if submission doesn't meet submission format requirements)

Be sure to save a copy of the Unity project in the state that you submitted, in case we have any problems with grading (such as forgetting to submit a file we need).