Milestone 2 – Physics Simulation and Game Feel Gardens

This is an *INDIVIDUAL* assignment.

Due: Friday, February 17th, 2017 @ 11:55 PM

Late Policy
See milestone 1 for the late policy (2^(n+1) late policy).

Description
This assignment involves the modification of your controllable character from milestone 1. You will modify your character such that it can interact with a physically simulated world in such a way that it meets some aspects of Game Feel. Additionally, you will create a Game Feel garden for your character to interact within.

General Notes

The HUD must display your full name.

It is highly recommended that you make regular backups of your project. Please consider using git on your local machine, and regularly check-in your project. Alternatively, consider enrolling in Unity's Collaborate beta. It is also suggested that you periodically save your scene as a new file name (e.g. "marioland_v0.scene", "marioland_v1.scene", etc.). Finally, occasionally exit Unity, especially if you are leaving your computer for a while. It's easy to forget to save both the scene and the project regularly and lose changes. Exiting helps to force you to get them saved.

Character Controller Updates

For the first objective, you will update your character controller to support physical interactions. This will be based on extending the work you completed in Milestone 1. Feel free to abandon the special grading keyboard controls and implement more conventional controls, possibly also including gamepad support (a PS4 controller with USB cable is recommended and works with PCs and Macs). However, you must maintain root motion based movement.

Your character will interact with physical objects, most likely through use of a capsule collider and rigid body component. In fact, you may already have this implemented. The collider should provide reasonable collisions with objects in the environment and with minimal interpenetration of 3D models. For instance, your character should be able to bump into rigid body objects in a scene (e.g. push crates or spherical objects around). The specific interactions are up to you to decide upon and implement.

Footstep Sounds and Particle Effects

Your character should make footsteps based on the speed at which he/she walks/runs. This can be accomplished by creating a Mecanim animation curve that exposes a value that specifies whether a foot is touching the ground or not. Then, in script you can watch this value to decide when to play a footstep sound and create a particle effect, such as dust being kicked up. Note that Mecanim animation events are also effective, but with blended animations that present the same event (e.g. footsteps from walking and running animations) you will find that you get more footstep callbacks than desired. In this case, you will need to implement logic to filter out extraneous footstep events.

You will also need to access the character's skeleton to get the world coordinates of the foot for placing the particle effect. Both the footstep sound and particle effect should be dependent on the type of ground surface stepped upon. For instance, sand should make a sound different than mud or a wooden floor. Unique particle effects should be demonstrated for at least two of your ground material types, but feel free to make more than that.

Rag Doll Physics

Your character will utilize a ragdoll mode for a death sequence within your Game Feel garden. For instance, if you character steps on lava, gets hit by a rolling boulder, etc., you can trigger the death. Upon incapacitation, your animated character should become a humanoid ragdoll (no longer controlled by the Mecanim animation system and instead simulate weighted limbs under the influence of gravity and other forces). We recommend use of the Ragdoll Wizard in Unity and perhaps a conveniently placed staircase. A few seconds after death, the character should respawn at the starting point.

Game Feel Garden

You will be creating a Game Feel garden (as a scene in Unity) that leverage the physics interactions of your character controller. Game Feel gardens are environments meant to showcase a game character with realtime control. The user can explore the environment without any particular goal other than to play around with the controls. Each environment provides interaction possibilities unique to that environment. For instance, an icy world may be slippery, a canyon region may have surging wind that affects the player, etc.

You should have two distinct areas in your Game Feel garden, each with a unique theme of interaction. Each area will provide a unique theme/biome in which the player can interact. Additionally, the game feel garden must provide a variety of game objects that are part of the physics simulation and that the player can influence in some way. Furthermore, your garden should provide some reasonable level of polish including textured surfaces, a skybox, lighting effects, and sound effects.

Consider using a level editor plugin for Unity such as ProBuilder Basic along with various prefab objects you find on the asset store.

Rich Sound Feedback via Event-Based Messaging System

You will support rich environmental feedback via audio output. To best support this, you will implement an event-based messaging system to pass all rigid body collision events on to an audio feedback manager. At minimum, the audio feedback manager should play sounds according to the type of object involved in the interaction/collision and the position of the interaction/collision. This includes the character colliding with static objects, such as walls.

Many objects within the environment should also include persistent looping sounds such as a waterfall running, beeps and whirrs from a computer terminal, etc.

Rich and even exaggerated audio feedback is an excellent way to communicate the interactive aspects of a game environment. Additionally, the multimodal nature of visual game objects with corresponding auditory representation is a means of supporting Universal Design principles. Ultimately, well-designed audio feedback can support a game-playing audience with a variety of accessibility needs, such as those with visual impairments.

Event-based Messaging Example:
https://unity3d.com/learn/tutorials/topics/scripting/events-creating-simple-messaging-system

Grading Criteria

Your submission should satisfy the following requirements.

Character Control **(50 points total, breakdown below)**

Basic physics interaction **(10 points)**
Footstep sounds and particle effects dependent upon ground surface **(20 points)**
Ragdoll effect upon death of character, with delayed respawn **(20 points)**

Game Feel Garden **(30 points total, breakdown below)**

The game feel garden represents a unique environment visually, auditorily, and interactively (via physical simulation). There should be two distinct themes within the scene.

The Game Feel garden must exhibit the following environmental behaviors:

- The scene should have at least five **unique** physically simulated object types (geometry nodes with rigid bodies and collider shapes) in it that have dynamic physical properties controlled by the physics engine. Examples include: crates, boulders, weighted companion cubes, happy fun ball, etc. You don't need to be the author of the models, and you can acquire prefab objects from the Asset Store. Lastly, make prefabs (if they aren't already) of your five object types and replicate judiciously for a more interactive environment (e.g. a pile of rocks). **(5 pts)**
- The scene should have at least one unique compound object type consisting of joints somewhere in it. For the compound object, you should be the author of the joints/constraints which form the compound object. (e.g. door, suspension bridge, turnstile, chain, etc.). **(5 pts)**
- The scene must have variable height terrain (e.g. ramps, stairs, platforms, mesh terrain, etc.) that your character can move up and down on. **(5 pts)**
- The scene must have at least two ground materials impacting footstep sounds and particle effects of the character. **(5 pts)**
- Game Feel: The scene must have two uniquely identifiable themes. It should also have a level of polish that improves: the game feel of the character, and sensation of interacting with a spatial simulation. This should include graphic-textured surfaces, skybox, lighting effects, etc. **(10 pts)**

Rich Sound Feedback via Event-Based Messaging System **(20 points)**

- All your rigid body objects should make unique sounds when interacted with. This can be satisfied with audio feedback upon collisions with the player, other rigid bodies, and static colliders, such as the floor/walls.
- You must use an event-based messaging system based on Unity's Event support.
- At minimum, your events must communicate object/material type and position. Therefore, your events should be parameterized. For instance, a boulder bouncing down a mountain will make stone sounds, and each sound will appear to come from the point of each collision.

- You should also have at least 2 object types that give off continuous audio feedback (e.g. broken robot, a crate with an angry wolverine inside, etc.). Note that you don't specifically need to use the event mechanism for continuous audio.

Extra Credit

Do one of the options below for **UP TO** 5 points. Do both 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.

Real-Time Foley Effects Engine a la Trespasser

Expand upon the "Rich Sound Feedback via Event-Based Messaging System" requirement to include features originally seen in Trespasser's game engine:

- Attenuation (volume level) based upon collision force
- Variation of event sounds by randomly selecting from a sound bank of similar sounds (e.g. 3 different recordings of stone hitting stone)
- Further variation via slight changes to pitch, perhaps determined by collision force, direction, etc.
- Selection of sound bank according to material interactions (e.g. stone-on-stone sound bank versus metal-on-stone sound bank)

Carry Rigid Body Object

Modify your character to animate picking up, carrying, and dropping a small rigid body object. (You can turn off physics for the object while carrying.)

Resources:

Event System Tutorial - https://unity3d.com/learn/tutorials/topics/scripting/events-creating-simple-messaging-system

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 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>_mX.zip (X is milestone #)
c) A /build/ directory should contain a build of your game. Please make sure you preserve the data directory that accompanies the EXE (if submitting a Windows build)
d) Readme file should be in the top-level directory: < lastName_firstInitial >_mX_readme.txt and should contain the following
    i. Full name, email, and TSquare account name

ii. Detail which platform your executable build targets (Windows, OSX, GlaDOS, etc.) Also let us know if you implemented game controller support, and which one you used.
iii. Specify which requirements you have completed, which are incomplete, and which are buggy (be specific)
iv. Detail any and all resources that were acquired outside of class and what they are 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”). This also includes other students that helped you or that you helped.
v. 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
vi. Detail exact steps grader should take to demonstrate that your game meets assignment requirements. (E.g. “First, walk towards the pile of blocks and bump into them to knock them down. This should demonstrate actor movement via physically simulated forces and interactivity with environment…”)
vii. 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: (**up to 20 points deducted** 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).