Milestone 2 – Physics Simulation

This is an *INDIVIDUAL* assignment.

Late Policy
See Canvas course site for the late policy.

Use Unity version: <SEE SYLLABUS FOR REQUIRED VERSION NUMBER>

Description

In this assignment, you will be creating a number of physics objects and explore some programmatic control of the physics simulation. You should use the project you submitted for M1 as the foundation. If you aren't happy with your submitted M1, you can use the original github project: https://github.gatech.edu/IMTC/CS4455_M1_Support

Tasks (100 points total)

Make sure the HUD displays your full name.

Implement the following objects in view of the camera at the start of your game (in 16:9/16:10 aspect). Read everything first to have an idea where to put the different objects.

While a few material colors are provided for you, you will need to create some of them!

Basic Rigidbodies with Collision Events
Make three (3) **blue** rigidbody spheres aligned roughly vertically that topple and roll off one another as soon as the game starts. Modify the blue spheres to generate an event (BombBounceEvent already provided) upon collision (e.g. OnCollisionEnter()). This event will be consumed by the AudioEventManager and the bombBounceAudio (glass clink sound) will be played. The crates that are already in the scene have a CrateCollisionReporter script that you might find of interest for completing this task. **(5 points)**

Physics Layers Example
Make three (3) **red** rigidbody spheres aligned roughly vertically that belong to a physics layer that does **not** allow collisions with other members of the same layer. Assigning Layers and tweaks to the Physics Layers Collision Matrix are necessary. The end result should be the spheres collapse into and interpenetrate with one another when the game starts. **(5 points)**

Compound Collider for Complex Geometry
Using the Asset Store, search for "Free Japanese Mask" and import it (looks like a stone mask of a fox and the model is named "Kitsune"). Place it in your scene somewhere and scale size uniformly by 10.0. Next add a rigidbody controller to the Kitsune GameObject. Add a few child GameObjects that each contain a simple collider like capsules, cylinders, and boxes but are otherwise invisible. Make sure these children do **not** have rigidbody components attached. The union of these colliders should approximate the visual dimensions of the mask and collectively provide a good collision shape. Orient the mask such that it falls over and tips on the nose upon start of the game. This is an example of a compound collider physics object. **(10 points)**

Chain with Joint Constraints
Create a **yellow** chain out of at least five (5) rigidbody GameObjects of your choosing and form the chain with ConfigurableJoints. Hang it from up in the air somewhere. Orient the chain such that it is swinging slightly at the start of the game. **(10 points)**

Mecanim-Animated Kinematic Elevator

Add an empty GameObject and name it ElevatorRoot. Create a **blue** child cube GameObject called Elevator with zeroed out transform position and rotation. Scale it to look more like an elevator platform. Add to the Elevator a rigidbody component configured to be kinematic. With the Elevator object selected in the Hierarchy, open the Animation Window. Click the "Create" button within the Animation Window. Save the animation as "Elevator" under Assets/Animations. Confirm that the Elevator GameObject now has an Animator component with AnimatorController of same name assigned.

On the Animation Window, add a property of Transform->Position (click the plus to add it). Add a keyframe midway through your animation so that the Elevator rises and then goes back to the original position. Confirm the animation is set to loop (Inspector view of the Elevator animation selected in the Project View Assets). Test it out. You may need to adjust the speed multiplier in the Animator window view to slow things down. If you don't like the placement of the elevator you can move the "ElevatorRoot" without redoing the animation. You get similar benefits if you want to make a prefab of the elevator and clone it multiple times.

Now add a **red** rigidbody sphere on top of the elevator that gets pushed up and then falls repeatedly as the elevator goes up and down.

(10 points)

Customized Center of Mass

Create a **green** Weeble Wobble or Punching Bozo. These are rounded bottom-heavy toys that cannot be knocked over. Make it out of a capsule collider and a custom center of mass via the Project:Assets/Scripts/Utility/RigidbodyCenterOfMass MonoBehavior with accompanying RigidbodyCenterofMassHandler editor extension. (Be sure and read the script source to understand how it works!) Place it at an angle so it is rocking back and forth at the start of the game. **(10 points)**

Default Friction

Create a **purple** rigidbody box and place it on the ramp named BetterRamp. Confirm that it does not slide down the ramp. **(5 points)**

Custom Friction (Low)

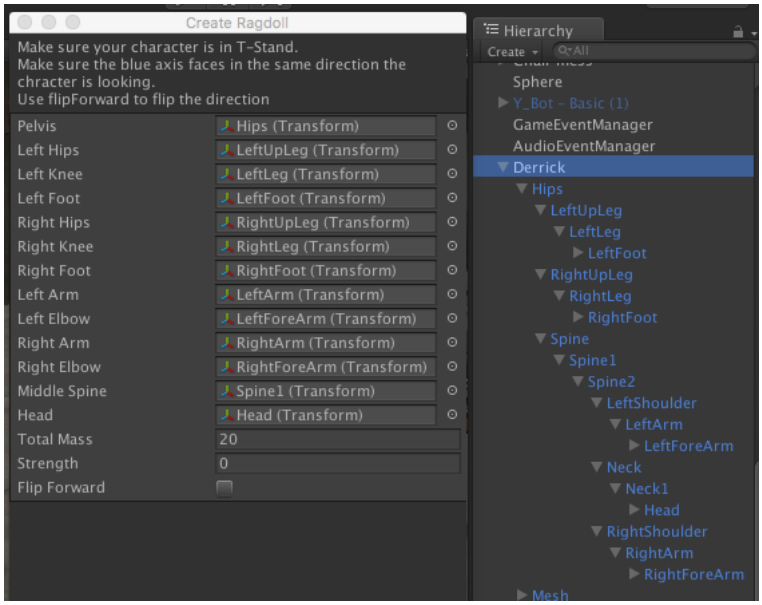
Next to the box implemented from the previous task, create a **green** rigidbody box. Modify it to have a new PhysicsMaterial that has 0.1 dynamic and static friction and 'Friction Combine' set to 'minimum'. Make sure this box slides down the ramp when the game is played. **(5 points)**

Bouncy Rigidbody

Create an **orange** rigidbody sphere. Assign a new PhysicsMaterial that has a 0.9 bounciness and 'Bounce Combine' set to 'maximum'. Place it a bit above the ground. You should see the ball bouncing when the game plays. **(5 points)**

Ragdoll

Place the existing 'Y_Bot' asset in the scene right above the nearest hurdle.
(Project:Assets/ModelsAndAnimations/YBot with Animations/Y_Bot/Y_Bot)
Run the Ragdoll Wizard via Edit Menu:GameObject->3D Object->Ragdoll…
Assign the bones as detailed in the image below. Make sure Y_Bot is aligned such that he flops over the hurdle as he falls when the game starts. **(15 points)**



Scripting Forces

Make a **black** click beetle or jumping bean out of a capsule. You will need to write a MonoBehavior script that intermittently applies a force to the rigidbody in FixedUpdate(). An impulse or instant velocity change for the force type will give a good jump effect. Liberal use of the Random functions will make the behavior more realistic (e.g. bounded but random time between jumps, jump angle and force magnitude, etc.). Also, jumps should only occur when the GameObject is on the ground. **(20 points)**

Congratulations, you have now experimented with many of the rigidbody physics simulation capabilities of Unity and are ready to apply these features to your game project!

Task Checklist

Modify the scene to include all of the following within view of the camera at start of the game:

1.) Your name appears in the HUD
2.) three (3) **blue** rigidbody spheres with collision sounds
3.) three (3) **red** rigidbody spheres that don't collide with one another
4.) Asset Store model "Free Japanese Mask" with custom compound collider
5.) **yellow** jointed chain made of at least five (5) rigidbody GameObjects
6.) **blue** kinematic rigidbody elevator using Mecanim animation with **red** rigidbody sphere going for a ride
7.) **green** Weeble Wobble/Punching Bozo that tilts but can't be knocked over
8.) a **purple** cube with rigidbody box that does **not** slide down the ramp
9.) **green** cube with rigidbody box that does slide down the ramp
10.) **orange** rigidbody sphere that bounces
11.) Y_Bot ragdoll that collapses over the hurdle
12.) **black** click beetle or jumping bean that jumps intermittently
13.) submit project according to submission requirements (Make sure you pass audit)

Submission:

You will submit a Zip/7Zip of your project via Canvas. If the file is too big for Canvas, then submit a link to a private cloud hosting (such as GT's Box license). **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. Refer to Assignment Packaging and Submission on the Canvas Syllabus for further details.**

The submissions should follow these guidelines:

a) Your name should appear on the HUD of your game when it is running.
b) Follow the *Assignment Packaging and Submission* steps including:
    i. ZIP file: <*lastName_firstInitial*>_m<*milestone number*>.*zip*
    ii. Complete Unity Project
    iii. Builds
    iv. Readme file should be in the top-level directory: <*lastName_firstInitial*>_m<*milestone number*>_*readme.txt* and should follow base requirements from *Assignment Packaging and Submission*
    v. Size reduction

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). Do not alter or remove your submission from cloud hosting until your grade has been returned.

Resources:

Scaffolding project for this assignment:
https://github.gatech.edu/IMTC/CS4455_M1_Support

Collision Action Matrix – This is very useful for understanding Unity's collision rules
https://docs.unity3d.com/Manual/CollidersOverview.html

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

Also, be aware that the tutorial's EventManager has been improved and can be found in the M1 project resource.