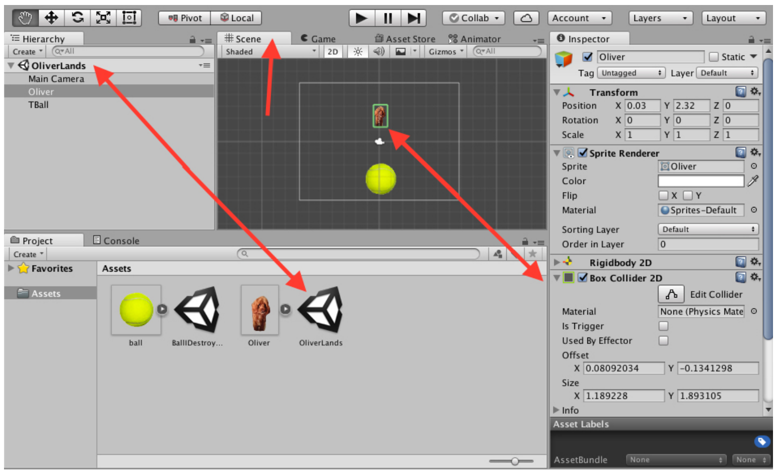
02 Phun with Physics
Code Community, 2017

Instructions: Follow the steps on your own or with a partner. Ask a tutor to help you if you get stuck. Or show a tutor how things work.

Project Goal: A simple collision video.
Unity Knowledge: *physics, scene, game object, component, MonoBehavior, Script,*
Integrated Programming Environment (IDE): panels: scene, hierarchy, inspector, project
Coding Knowledge Using classes such as Rigidbody 2D

Explore:

1. This project has two scene assets, OliverLands and BallDestroy. Find them in the project panel in the Assets folder. Click on Oliver in the Hierarchy window. Check out Oliver in the Inspector. Oliver has two new components: Rigid Body 2D and Box Collider 2D. The Rigid Body 2D object is closed. The Box Collider 2D object is open. In the Scene window the Box Collider is marked in green. Open the Rigid Body 2D object in the Inspector and notice that Oliver is a *dynamic* rigid body!



Oliver — Static — Tag: Untagged — Layer: Default
Transform
Position X 0.03 Y 2.32 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Sprite Renderer
Sprite: Oliver
Color
Flip: X Y
Material: Sprites-Default
Sorting Layer: Default
Order in Layer: 0
Rigidbody 2D
Box Collider 2D
Material: None (Physics Mate
Is Trigger
Used By Effector
Offset X 0.08092034 Y -0.1341298
Size X 1.189228 Y 1.893105
AssetBundle: None None

2. Click on theTBall object. Notice that it is a Kinematic Rigid Body – it doesn't actually do physics, but physics objects (dynamic) can interact with it. Notice that it has an Edge Collider 2D.

3. Your turn: Play with physics:
   a. Change the mass of Oliver.
   b. Change the TBall to a dynamic object. Then freeze its constraints. Change its mass. See what the other variables do.
   c. Change the TBall Collider to a sphere.
   d. Move Oliver to one side a bit.
   e. Put a 'ground' object below the ball that is Kinematic so that the ball can only roll sideways.
   f. Add more balls to the picture (how can you duplicate an object?)
   g. Make gravity move 'up' rather than down – you do this on the object.
   h. Make gravity move 'sideways' along the x axis. (Edit project settings, 2D).
4. Of course you want stuff to happen other than physics when things collide. This requires code. Switch the scene to BallDestroyed. Notice that the ball is dynamic. Run the project. Make sure you stop the run, don't just pause it. Now add the script OliverCollision as a component to the game object Oliver. (There are a number of different ways to do this. The easiest is to drag the script over to the Inspector for Oliver. Only compiled scripts (those that don't have syntax errors) can become components. Run the scene again, then study the script.
5. Your turn: Play with collisions:
   a. Try using different colliders other than box or edge.
   b. Add another object to the scene make things collide.
   c. Add a border around your scene so that objects stop when they get to the border. (This one is a challenge. If you can't figure it out, what till next time!)
   d. Make a game out of it: Challenge a friend to set up the variables in object inspectors to avoid being destroyed.

© RiverSound Solutions, 2017

Resource Information:

The Unity Physics Engine

Physics requires that an object in the hierarchy have a Rigid Body component. In order to interact with other objects it should also have a Collider. 2D projects have rigid body and collider classes that differ from 3D; they are more efficient. The most important point for 2D physics is that all collisions take place on the plane where z = 0. In order to make things happen when two bodies collide, you need to have scripts (MonoBehaviour components) that can be called by the event handler when the collision is entered, while it is happening (stay) and when it is exited. The Unity physics tutorials are pretty good. You could spend weeks elaborating on them. Please try to stick to the 2D Physics Tutorials and Assignments though, because 3D adds a whole dimension to this. (No pun intended…)

Object Oriented Programming

The theme emerging in these tutorials is that objects are everything. Objects like MonoBehaviors can be extended so that your script *inherits* all the existing data and methods of the parent. Objects are also connected through has-a links that allow objects to interact through code and data sharing. These ideas will be critical to understanding how to build up a project in Unity. Creating rules for movement, making user interfaces (UIs) and controlling input devices like keyboards, mice (and fingers on mobile devices) all require 'wiring up' the relationships between objects.

© RiverSound Solutions, 2017