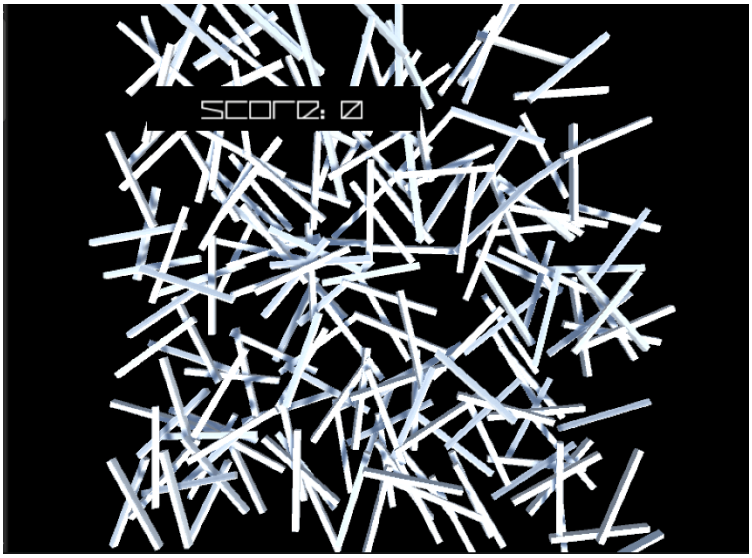
CODE NINJAS®

Bronze Belt Ninja Guide

Activity 05 Prove Yourself: Don't Touch the Chopsticks

Prove Yourself: Don't Touch the Chopsticks



A cube is a 3-dimensional shape with 6 sides made up of squares.

A square is a 2-dimensional shape that has 4 sides of equal length.

The shapes you can see in the above image are called rectangular prisms. 2 of the sides are still squares, but the length of the shape has been changed so that 4 of the sides are rectangles.

To test your understanding of how to use the Unity interface and 3D shapes, go into your complete Don't Touch the Cubes game, and do the following:

1. **Resize the cubes to become rectangular prisms as in the image above.**

2. **Adjust the rotation and movement speed of the obstacles.**

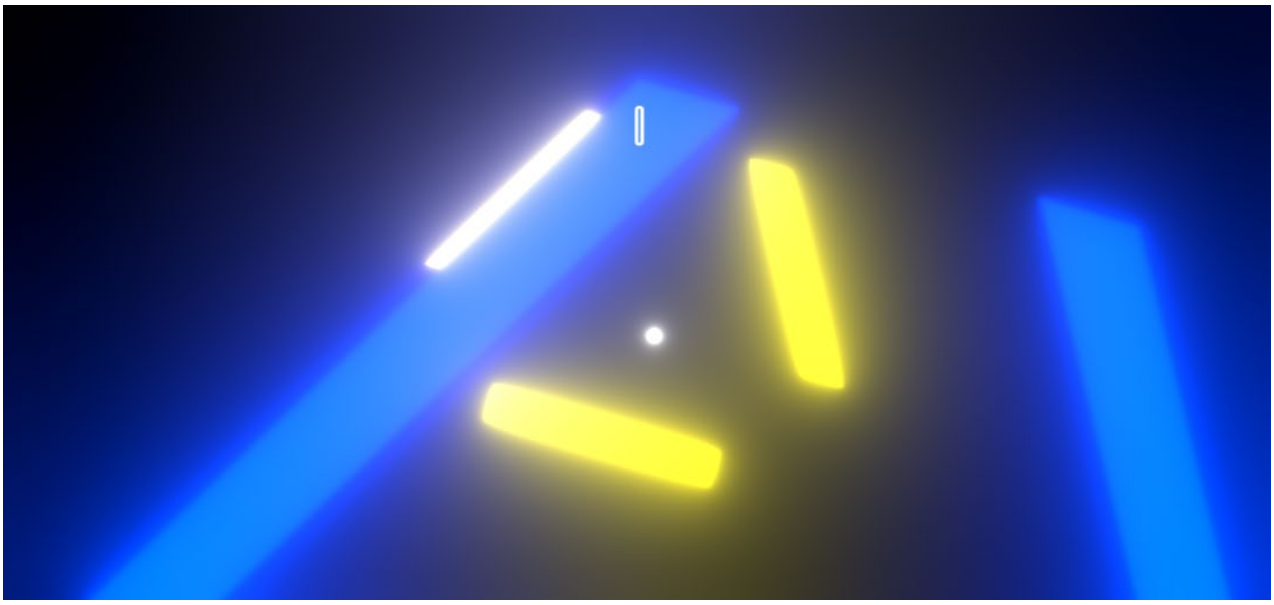
3. **Try adjusting the Time to Destruction variable if the objects in your game disappear before you can reach them.**

4. **Adjust the difficulty variable by making it lower to make the game easier and higher for a more difficult game.**

Prefabs

When you're making a game, the last thing that you want to do is manually make the dozens (or even hundreds) of objects that the player must deal with one at a time. It is much easier just to set it up one time and then let the computer take care of the rest.

The Prefab in Unity is just that. Any combination of **GameObjects** and components can be set aside for when you need them again. A prefab could be a combination of models to quickly build a scene, or it can be an assortment of characters that the player must deal with. The bonus of using a prefab is that any changes that you make to the original get applied to all copies.

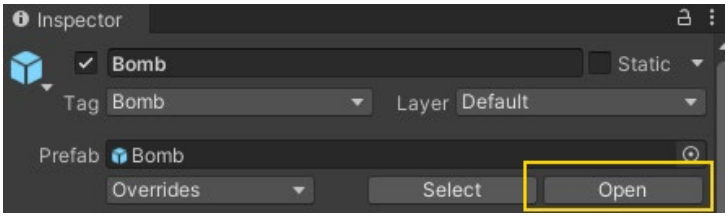


Making a Big Deal out of Many, Little Deals

Any time that you need to reuse a **GameObject** in your scene, you should make it into a **Prefab**. Making a prefab is as simple as dragging a **GameObject** from the Hierarchy into your **Prefabs** folder. Both the original object and the Prefab are now marked with blue cubes to indicate that the object in the scene is an instance of the prefab from your folder.

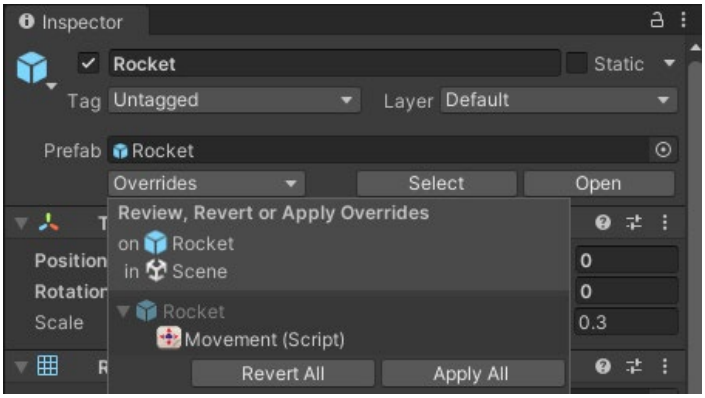
Opening a Prefab

To edit an asset in the **Prefabs** folder, select it and click **Open Prefab**. This opens Prefab Mode where you can make changes to the original prefab that then get applied to all instances of that prefab in your scene.



Overrides

Any changes that you make to a prefab in your scene only affect that instance of the prefab. If, for some reason, you want those changes to be saved as part of the original prefab (and applied to all the prefabs in a scene), you must click on **Override** in the **Inspector** panel and select **Apply All**. Any unapplied overrides in a prefab show up in bold text in the **Inspector** panel.



Instantiating

Prefabs can be taken out of the Prefabs folder and added to your game through code at run time. To do so, you will need to define that variable in your code (usually through a public GameObject variable) and use the Instantiate command to create new instances. Instantiate requires three things: the variable for the prefab, the position where to put it in the scene, and the desired rotation.

```
Instantiate(bombPrefab, new Vector3(randomX, spawnY, 0), bombPrefab.transform.rotation);
```

Prefabs and Variables

When instantiating a prefab from the **Prefabs** folder, care must be taken with what is attached to it as a public variable. Prefab scripts have no problem with public variables that are self-contained, such as a number for a speed setting or a boolean value. However, if you have a variable that requires linking to another GameObject in the scene, the prefab will forget that information when the Prefab is instantiated (the exception is if that object is another prefab in the same folder). In those cases, it's best to use a private variable and define that variable using *GameObject.Find("ObjectName")* in the Start function of the script.