Unit 4 Lab

Basic Gameplay

Steps:

Step 1: Give objects basic movement

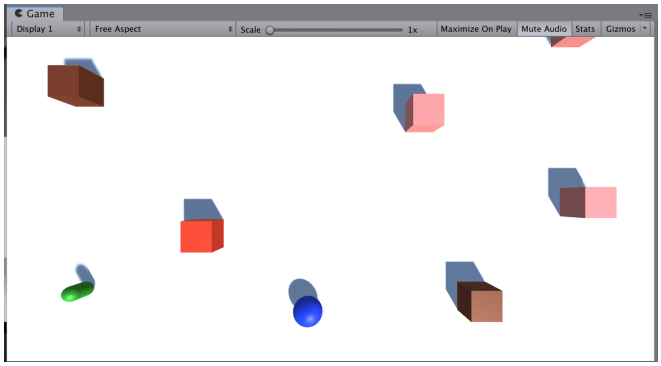
Step 2: Destroy objects off-screen

Step 3: Handle object collisions

Step 4: Make objects into prefabs

Step 5: Make SpawnManager spawn Prefabs

Example of progress by end of lab



Length: 60 minutes

Overview: In this lab, you will work with all of your non-player objects in order to bring your project to life with its basic gameplay. You will give your projectiles, pickups, or enemies their basic movement and collision detection, make them into prefabs, and have them spawned randomly by a spawn manager. By the end of this lab, you should have a glimpse into the core functionality of your game.

Project Outcome: Non-player objects are spawned at appropriate locations in the scene with basic movement. When objects collide with each other, they react as intended, by either bouncing or being destroyed.

Learning Objectives: By the end of this lab, you will be able to:
- More comfortably program basic movement
- More comfortably handle object collisions
- More comfortably spawn object prefabs on timed intervals

Step 1: Give objects basic movement

Before you spawn objects into your scene, they should move the way you want.

1. If relevant, add **Rigidbody** components to your non-player objects
2. Create a **new script**(s) for the objects that will be instantiated during gameplay and attach them to their respective objects (including projectiles or pickups)
3. Program the **basic movement** for your objects and test that they work

- **Tip:** Make sure you uncheck "use gravity" if you don't want them to fall
- **Tip:** In the collider component, check "is trigger" if you don't want actual collisions

By the end of this step, all objects should basically move the way they should in the game.

Step 2: Destroy objects off-screen

To make sure our hierarchy doesn't get too cluttered, let's make sure these objects get destroyed when they leave the screen.

1. Either create a new script or add code to your existing script to make sure objects are destroyed when they leave the screen

- **Tip:** Move your objects in scene view to determine the xyz positions objects should be destroyed

By the end of this step, objects should be removed from the hierarchy when they are no longer in play.

Step 3: Handle object collisions

Now that you have all these moving objects, they're bound to start colliding with each other - we need to program what should happen when everything collides.

1. If relevant, edit the **Rigidbody mass** of your objects
2. If relevant, to change the way your objects collide, create a new **Physics material** for your objects
3. Add **tags** to your objects so you can accurately test for which objects are colliding with which
4. Use ***OnCollisionEnter()*** (for Rigidbody collisions) or ***OnTriggerEnter()*** (for trigger-based collisions) to **Destroy** or **Log** messages to the console what should happen when certain collisions occur

- **Don't worry:** If you collide with a powerup or pickup, the actual functionality does *not* need to be programmed, just the effect
- **Tip:** Should use OnTriggerEnter if objects are being destroyed - but remember that "Is Trigger" must be checked for this to work!

By the end of this step, objects should destroy, bounce, or do nothing based on collisions.

Step 4: Make objects into prefabs

Now that the objects are basically behaving the way they should, if they're going to be instantiated during gameplay, they need to be prefabs

1. In the Assets directory, create a new **Folder** called "Prefabs"
2. **Drag** in each object to create a **new prefab** for it
3. After all objects have been turned into prefabs, **delete** them from the scene
4. **Test** the objects' behavior by dragging them from the Prefabs folder into the scene while the game is running

- **Tip:** When creating new prefabs, you have to drag them one at a time
- **Tip:** Notice that their icons turn blue when they are prefabs

By the end of this step, all objects that will be spawned during gameplay should be prefabs and should no longer be in your scene.

Step 5: Make SpawnManager spawn Prefabs

Now that we have all of our prefabs set up, we can create a spawn manager to spawn them at intervals and, if we want, in random locations.

1. Create an Empty "Spawn Manager" object and attach a new SpawnManager.cs script to it
2. Create individual **GameObject or GameObject array** variables for your prefabs, then **assign** them in the inspector
3. Use the ***Instantiate()***, ***Random.Range()***, and the ***InvokeRepeating()*** methods to spawn objects at intervals (random objects, random locations, or both)
4. Right-click on your *Assets folder* > ***Export Package*** then save a new version in your **Backups** folder

- **Tip:** Name your variables "____Prefab" so you know it requires a prefab value
- **Don't worry:** If it's not perfect yet of if there are some minor bugs - just get the general idea working

By the end of this step, objects should be spawned automatically from the appropriate location.

Lesson Recap

New Progress	• Non-player objects prefabs have basic movement • Objects are destroyed when they leave the screen • Collisions between objects are handled appropriately • Objects are spawned at the appropriate locations on time-based intervals
New Concepts and Skills	• Creating basic gameplay for a project independently