CS 3160 – Game Programming

Project 1 – Unity Basic Scripting

Learning Objectives

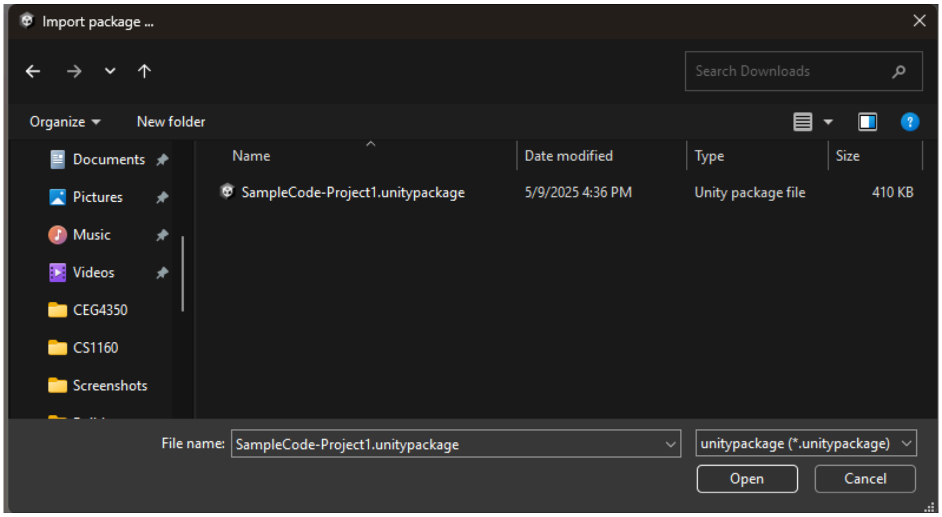
- Learn the basics of Unity scripting, creating prefabs, and instantiating and destroying GameObjects.

Overview

In this project you will become familiar with creating new prefabs, writing new scripts, creating and destroying GameObjects.

Sample Code

To complete this project some sample code is required. This is provided in the form of a Unity Package. Download the “SampleCode-Project1” Unity Package from Pilot. First, create an empty Unity project using the “3D (Built-In Render Pipeline)” template. Then, go to “Assets” > “Import Package” > “Custom Package…” then locate “SampleCode-Project1.unitypackage” downloaded from Pilot and import all files:

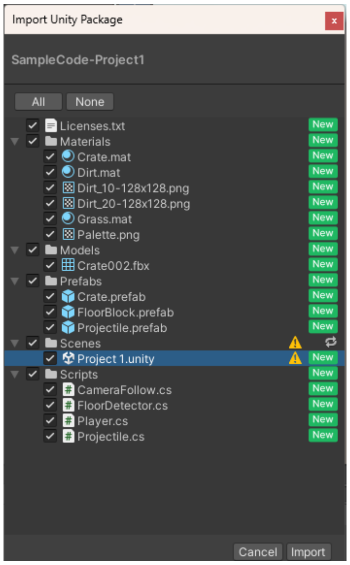


Import package ...

Name	Date modified	Type	Size
SampleCode-Project1.unitypackage	5/9/2025 4:36 PM	Unity package file	410 KB

File name: SampleCode-Project1.unitypackage

unitypackage (*.unitypackage)

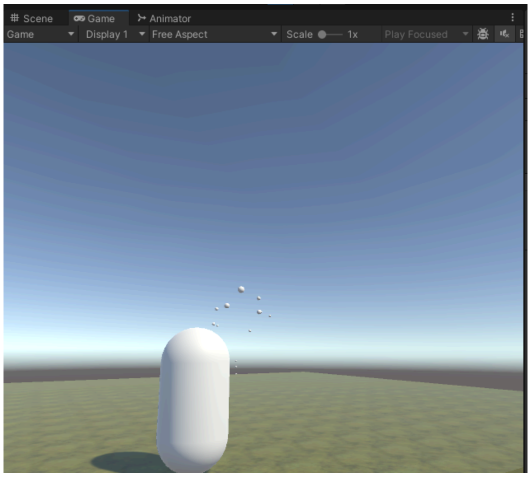


Import Unity Package

SampleCode-Project1

☑ Licenses.txt	New
☑ Materials	New
☑ Crate.mat	New
☑ Dirt.mat	New
☑ Dirt_10-128x128.png	New
☑ Dirt_20-128x128.png	New
☑ Grass.mat	New
☑ Palette.png	New
☑ Models	New
☑ Crate002.fbx	New
☑ Prefabs	New
☑ Crate.prefab	New
☑ FloorBlock.prefab	New
☑ Projectile.prefab	New
☑ Scenes	⚠
☑ Project 1.unity	⚠ New
☑ Scripts	New
☑ CameraFollow.cs	New
☑ FloorDetector.cs	New
☑ Player.cs	New
☑ Projectile.cs	New

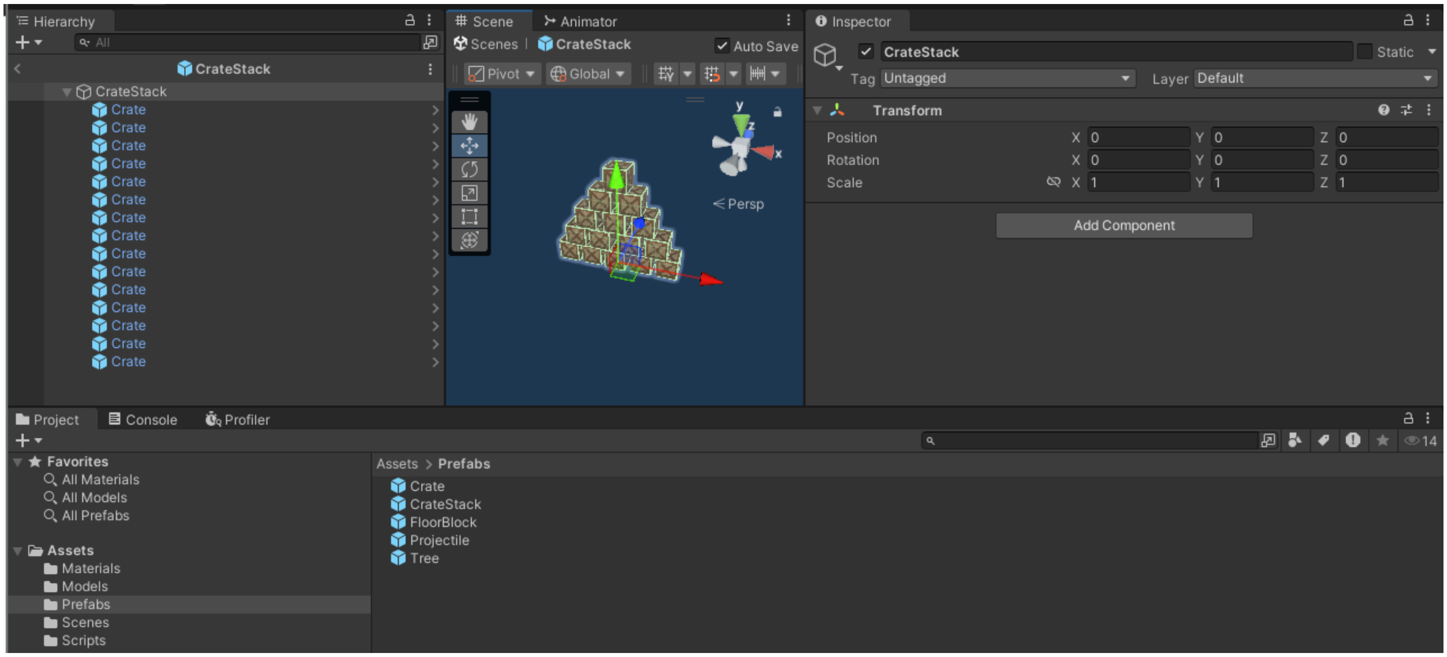
If importing was successful you should be able to load the “Project 1” Scene in the “Scenes” folder and you can test the game and fire projectiles:



NOTE: Importing a Unity Package does not import Layers, this breaks the jumping mechanic. To fix this, create a new Layer called “Floor” and set the “FloorBlock” prefab’s layer to the newly created layer.

CrateStack Prefab

First, create a stack of crates by duplicating the provided “Crate” prefab in the “Prefabs” folder. You can organize them however you like. It is recommended that you create this stack of crates near the origin seen here. Lastly, place this “CrateStack” prefab inside of the “Prefabs” folder:



CrateStackSpawner Script

We want to create a way to respawn or regenerate our stack of crates because once the player knocks over the crates, there is nothing fun left to do. Create a new script called “CrateStackSpawner” in the “Scripts” folder. CrateStackSpawner must have 4 methods:

- Start()
  - Calls the CreateNewCrates() method to create a new stack of crates at the beginning of the game
- Update()
  - Check if the player types “R” then:
    - Call DeleteOldCrates() method to remove the old previous crates
    - Call CreateNewCrates() method to create a new stack of crates
- CreateNewCrates()
  - Check if a prefab is not null then:
    - Instantiate a new prefab, hold onto the return value for destroying later
    - Set the position of the instantiated prefab to some starting position
- DeleteOldCrates()
  - Check if the instantiated prefab is not null then:
    - Destroy the instantiated prefab

Lastly, add the CrateStackSpawner script to the “Environment” GameObject in the Scene and provide the CrateStack prefab to the script component so it can spawn in your CrateStack prefab.

CrateHealth Script
Create a new script called “CrateHealth” in the “Scripts” folder. This will be a way we can give health points to our crates so we can destroy them. Create two fields (variables):

- public float initialHealth = 100;
    - Holds the initial health of the crate that is used inside of the Start() method
- public float currentHealth;
    - Holds the current health of the crate
    - Set currentHealth = initialHealth inside of the Start() method

Write two methods in CrateHealth:

- Start()
    - Sets currentHealth to initialHealth to give the crate some starting health points
- TakeDamage(float damage)
    - Uses the parameter “damage” to subtract damage from currentHealth and overwrite currentHealth
    - Checks if currentHealth is less than or equal to 0 then:
        - Destroys the gameobject itself
        - Hint: Use Destroy(gameobject) to have a gameobject destroy itself

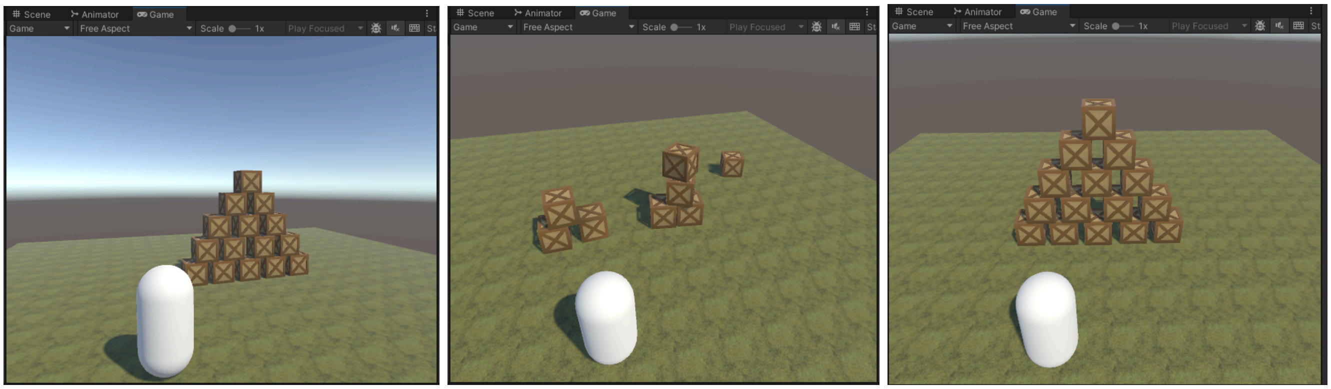
Lastly, add the CrateHealth script to the Crate prefab in the “Prefabs” folder. Now check your CrateStack prefab. If done correctly, all of the crates in your crate stack should now have the CrateHealth script attached.

Projectile Script
We must modify the Projectile script to allow projectiles to damage the crates. To do this add a new method to the Projectile script:

- private void OnCollisionEnter(Collision other)
    - Get the CrateHealth component from the “other.gameobject” which is the GameObject that the projectile has collided with.
    - If the CrateHealth component was found then:
        - Call the TakeDamage() method on the CrateHealth component
        - Pass in however much damage you like as an argument to the TakeDamage() method
    - Note: This is a standard method inside of Unity that will execute whenever the projectile touches another collider

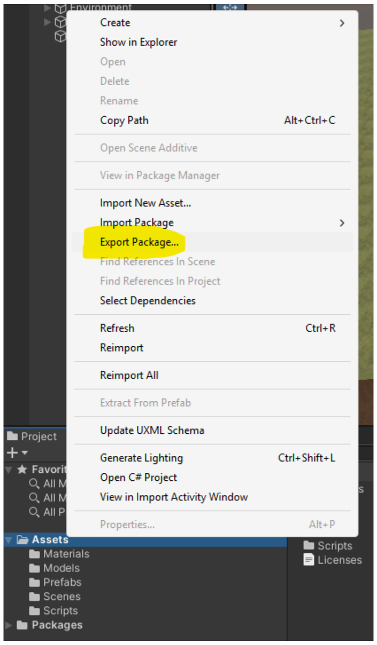
Results

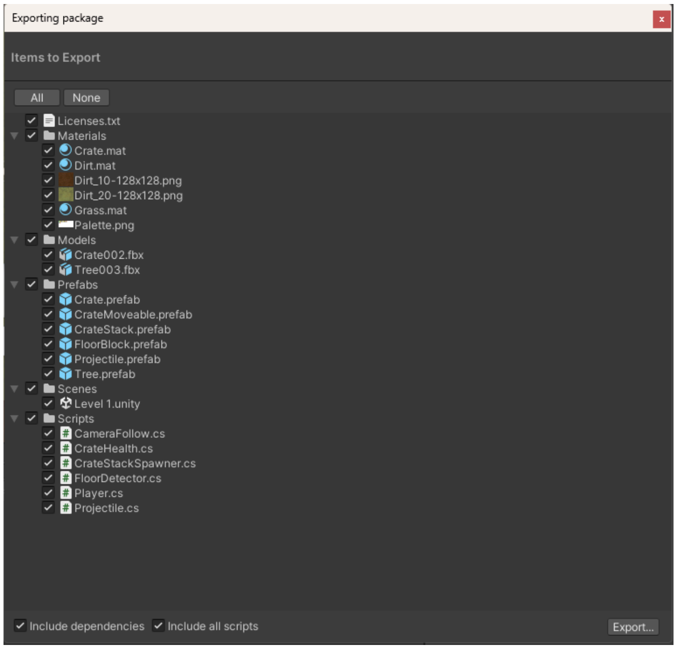
The resulting game should spawn a stack of crates at the start, allow you to destroy them with projectiles, then respawn them by pressing “R”. This is shown below:



How to Submit

To submit your code, you must pack your project into a Unity Package. To create a Unity Package first right click on the “Assets” folder inside of the Unity Editor, select all items and hit Export. Lastly, name your file “YourLastName-Project1” where “YourLastName” is your actual last name, and submit this file to the Pilot dropbox:





- ☑ Licenses.txt
- ☑ Materials
  - ☑ Crate.mat
  - ☑ Dirt.mat
  - ☑ Dirt_10-128x128.png
  - ☑ Dirt_20-128x128.png
  - ☑ Grass.mat
  - ☑ Palette.png
- ☑ Models
  - ☑ Crate002.fbx
  - ☑ Tree003.fbx
- ☑ Prefabs
  - ☑ Crate.prefab
  - ☑ CrateMoveable.prefab
  - ☑ CrateStack.prefab
  - ☑ FloorBlock.prefab
  - ☑ Projectile.prefab
  - ☑ Tree.prefab
- ☑ Scenes
  - ☑ Level 1.unity
- ☑ Scripts
  - ☑ CameraFollow.cs
  - ☑ CrateHealth.cs
  - ☑ CrateStackSpawner.cs
  - ☑ FloorDetector.cs
  - ☑ Player.cs
  - ☑ Projectile.cs

☑ Include dependencies ☑ Include all scripts

Grading

This lab is worth 5.00 points, distributed as follows:

Task	Points
There are no build or runtime errors	1.00
Created the CrateStack prefab	1.00
Wrote the CrateStackSpawner script with all 4 methods	1.00
Modified the Projectile script with additional method	1.00
Wrote the CrateHealth script with all 2 methods	1.00
Total	5.00