Junior Programmer Pathway

Mission 2: Create with Code 2

Pathway Description

Designed for anyone interested in learning to code or obtaining an entry-level Unity role, this pathway assumes a basic knowledge of Unity and has no math prerequisites. Junior Programmer prepares you to get Unity Certified so that you can demonstrate your job-readiness to employers.

Key details

- ➔ This Mission will take you approximately 14 hours to complete. Take it at your own pace — you'll receive XP each step of the way.
- ➔ Connect with the Unity community as you learn and check the Learn Live calendar for follow-along practical sessions with established Unity creators.
- ➔ When you've finished the Mission, you'll get the Mission badge for your profile and portfolio.

Skills covered in this course

Basic Code Comprehension
- ➔ Interpret simple code
- ➔ Improve simple code using the features of an IDE

Basic Application Scripting
- ➔ Use common logic structures to control the execution of code.
- ➔ Write code that utilizes the various Unity APIs
- ➔ Implement appropriate data types
- ➔ Write code that integrates into an existing system
- ➔ Implement a code style that is efficient and easy to read
- ➔ Prototype new concepts

Basic Debugging
- ➔ Diagnose and fix code that compiles, but fails to perform as expected
- ➔ Diagnose and fix common compilation errors
- ➔ Diagnose and fix compilation errors related to Unity's Scripting API
- ➔ Diagnose and fix the cause of an exception

Beginner Application scripting
- ➔ Create the scene flow in an application state
- ➔ Implement data persistence across scenes and user sessions
- ➔ Level 1 Version control
- ➔ Maintain a project by correctly implementing version control

- ➔ Implement best practices of version control using Unity Collaborate

Basic Code optimization

- ➔ Maximize code efficiency by correctly executing coding best practices
- ➔ Debug performance issues

Beginner Programming theory

- ➔ Analyze the principal pillars of object-oriented programming
- ➔ Simplify code and make it reusable by correctly implementing the principles of inheritance and polymorphism
- ➔ Make code more secure and usable by correctly implementing the principles of abstraction and encapsulation, including the use of interfaces
- ➔ Write efficient, organized, and comprehensible code by correctly implementing the principles of object-oriented programming

How to use the Pathway		
The Unity Essentials Pathway is broken up into 3 “Missions,” with each Mission containing multiple tutorials and assessments. The following Missions make up the complete Pathway:		
	Junior Programmer: Create with Code 1	13 hours and 45 Minutes
	Junior Programmer: Create with Code 2	24 hours and 15 Minutes
	Junior Programmer: Manage scene flow and data	2 hours
	Junior Programmer: Apply object-oriented principles	1 Hour 45 Minutes
Students are encouraged to complete all the Missions in the correct sequence to ensure the best learning experience.		

Table of contents

Mission 2: Create with code 2

Part of the Junior Programmer Pathway

Mission overview
In this Junior Programmer Mission, you'll expand on your foundational learning and create basic custom interactions with Unity. As you complete four practical projects, you'll explore a range of concepts that support basic functionality, including loops, data types, references, script communication, and UI. To complete the Mission, you'll create a simple prototype application for your portfolio. By the end of this Mission you will have acquired all the skills needed to take the Unity Certified User: Programmer certification exam.

Key details

→ This mission will take you approximately 25 hours to complete. Take it at your own pace — you'll receive XP each step of the way.
→ Remember, you're not alone; connect with the Unity community throughout the mission and check the Learn Live calendar for follow-along practical sessions with established Unity creators.
→ When you've finished the mission, you'll get the mission badge for your profile and portfolio.
→ Consider becoming a Unity Certified User: Programmer.

Skills
Basic application scripting

→ Use common logic structures to control the execution of code.
→ Write code that utilizes the various Unity APIs
→ Implement appropriate data types
→ Write code that integrates into an existing system
→ Implement a code style that is efficient and easy to read
→ Prototype new concepts

Basic debugging

→ Diagnose and fix code that compiles, but fails to perform as expected
→ Diagnose and fix common compilation errors
→ Diagnose and fix compilation errors related to Unity's Scripting API
→ Diagnose and fix the cause of an exception

Unit 3 - Sound and effects

Lesson link	[Unit 3 - Sound and effects]()
Length	**8 hours**

Summary

In this Unit, you will program a fast-paced endless side-scrolling runner game where the player needs to time jumps over oncoming obstacles to avoid crashing. In creating this prototype, you will learn how to add music and sound effects, completely transforming the experience of your projects. You will also learn how to create dynamic endless repeating backgrounds, which are critical for any side-scrolling games. Finally, you will learn to incorporate particle effects like splatters and explosions, which make your games so much more satisfying to play.

Objectives
By the end of this course, you will be able to:
→ Use GameObject.Find and GetComponent to manipulate the components of the current or other game objects
→ Tweak the gravity of your project with Physics.gravity and use ForceModes to apply forces in different ways
→ Utilize new operators like "&&" and bool variables to better control game logic
→ Freeze or constrain the RigidBody component to halt movement on certain axes
→ Use tags to label game objects and call them in the code
→ Use script communication to access the methods and variables of other scripts
→ Manage animation states in the Animator Controller, including adjusting the states' parameters and default state
→ Use SetTrigger, SetBool, and SetInt methods to trigger transitions between animation states
→ Stop and play particle effects to correspond with character animation states
→ Work with Audio Sources and Listeners to play background music
→ Add sound effects to add polish to your project

Skills
Basic application scripting
→ Use common logic structures to control the execution of code.
→ Write code that utilizes the various Unity APIs
→ Implement appropriate data types
→ Prototype new concepts

Basic debugging
→ Diagnose and fix code that compiles, but fails to perform as expected

→ Diagnose and fix common compilation errors
→ Diagnose and fix the cause of an exception

Unit 3 - Introduction

Lesson link	[Unit 3 - Introduction]()
Length	**5 minutes**
An introductory video for Unit 3, where you will learn to implement Animation, Sound, and Particle Effects.	
Steps 1. Introduction	

Lesson 3.1 - Jump force

Lesson link	[Lesson 3.1 - Jump force]()
Length	**1 hour 30 minutes**
Summary The goal of this lesson is to set up the basic gameplay for this prototype. We will start by creating a new project and importing the starter files. Next we will choose a beautiful background and a character for the player to control, and allow that character to jump with a tap of the spacebar. We will also choose an obstacle for the player, and create a spawn manager that throws them in the player's path at timed intervals. **Project outcome** The character, background, and obstacle of your choice will be set up. The player will be able to press spacebar and make the character jump, as obstacles spawn at the edge of the screen and block the player's path.	
Skills Basic application scripting ● Use common logic structures to control the execution of code. ● Write code that utilizes the various Unity APIs ● Implement appropriate data types	

Materials Prototype 3 - Starter files (.zip)
Steps 1. Open prototype & change background 2. Choose and set up a player character 3. Make player jump at start 4. Make player jump if spacebar pressed 5. Tweak the jump force and gravity 6. Prevent player from double-jumping 7. Make an obstacle and move it left 8. Create a spawn manager 9. Spawn obstacles at intervals 10. Lesson recap

Lesson 3.2 - Make the world whiz by

Lesson link	Lesson 3.2 - Make the world whiz by
Length	**1 hour 10 minutes**
Summary We've got the core mechanics of this game figured out: the player can tap the spacebar to jump over incoming obstacles. However, the player appears to be running for the first few seconds, but then the background just disappears! In order to fix this, we need to repeat the background seamlessly to make it look like the world is rushing by! We also need the game to halt when the player collides with an obstacle, stopping the background from repeating and stopping the obstacles from spawning. Lastly, we must destroy any obstacles that get past the player. **Project outcome** The background moves flawlessly at the same time as the obstacles, and the obstacles will despawn when they exit game boundaries. With the power of script communication, the background and spawn manager will halt when the player collides with an obstacle. Colliding with an obstacle will also trigger a game over message in the console log, halting the background and the spawn manager.	
Skills Basic application scripting ● Write code that utilizes the various Unity APIs	
Steps 1. Create a script to repeat background 2. Reset position of background 3. Fix background repeat with collider 4. Add a new game over trigger 5. Stop MoveLeft on gameOver	

6. Stop obstacle spawning on gameOver 7. Destroy obstacles that exit bounds 8. Lesson recap
New functionality ● Background repeats seamlessly ● Background stops when player collides with obstacle ● Obstacle spawning stops when player collides with obstacle ● Obstacles are destroyed off-screen **New concepts and skills** ● Repeat background ● Get Collider width ● Script communication ● Equal to (==) operator ● Tags ● CompareTag()

Lesson 3.3 - Don't just stand there

Lesson link	Lesson 3.3 - Don't just stand there	
Length	**1 hour**	
Summary The game is looking great so far, but the player character is a bit... lifeless. Instead of the character simply sliding across the ground, we're going to give it animations for running, jumping, and even death! We will also tweak the speed of these animations, timing them so they look perfect in the game environment. **Outcome** By the end of this tutorial, you will be able to: ● With the animations from the animator controller, the character will have 3 new animations that occur in 3 different game states. These states include running, jumping, and death, all of which transition smoothly and are timed to suit the game.		
Skills Write code that utilizes the various Unity APIs		
Steps 1. Explore the player's animations 2. Make the player start off at a run 3. Set up a jump animation 4. Adjust the jump animation		

5. Set up a falling animation
6. Keep player from unconscious jumping
7. Lesson recap

New functionality
- The player starts the scene with a fast-paced running animation
- When the player jumps, there is a jumping animation
- When the player crashes, the player falls over

New concepts and skills
- Animation Controllers
- Animation States, Layers, and Transitions
- Animation parameters
- Animation programming
- SetTrigger(), SetBool(), SetInt()
- Not (!) operator

Lesson 3.4 - Particles and sound effects

Lesson link	Lesson 3.4 - Particles and sound effects
Length	**1 hour**

Summary
This game is looking extremely good, but it’s missing something critical: sound effects and particle effects! Sounds and music will breathe life into an otherwise silent game world, and particles will make the player’s actions more dynamic and eye-popping. In this lesson, we will add cool sounds and particles when the character is running, jumping, and crashing.

Project outcome
Music will play as the player runs through the scene, kicking up dirt particles in a spray behind their feet. A springy sound will play as they jump and a boom will play as they crash, bursting in a cloud of smoke particles as they fall over.

Skills
Basic application scripting
- Write code that utilizes the various Unity APIs

Steps
1. Customize an explosion particle
2. Play the particle on collision
3. Add a dirt splatter particle

4. Add music to the camera object
5. Declare variables for Audio Clips
6. Play Audio Clips on jump and crash
7. Lesson recap

New functionality

- Music plays during the game
- Particle effects at the player's feet when they run
- Sound effects and explosion when the player hits an obstacle

New concepts and skills

- Particle systems
- Child object positioning
- Audio clips and Audio sources
- Play and stop sound effects

Overall recap

We've made an incredibly polished game – we have these super cool sound and particle effects. We have upbeat background music. We learned how to utilize animations for our characters, and we did some programming magic to make our background endlessly scroll.

Challenge 3 - Balloons, bombs, & booleans

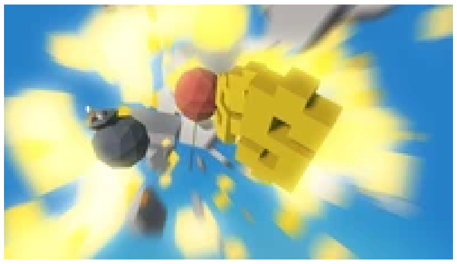
Lesson link	[Challenge 3 - Balloons, bombs, & booleans]()
Length	**1 hour**

Summary

Apply your knowledge of physics, scrolling backgrounds, and special effects to a balloon floating through town, picking up tokens while avoiding explosives. You will have to do a lot of troubleshooting in this project because it is riddled with errors.

Challenge outcome

- The balloon floats upwards as the player holds spacebar
- The background seamlessly repeats, simulating the balloon's movement
- Bombs and money tokens are spawned randomly on a timer
- When you collide with the money, there's a particle and sound effect
- When you collide with the bomb, there's an explosion and the background stops.



Skills

Basic application scripting
- Write code that utilizes the various Unity APIs

Basic debugging
- Diagnose and fix code that compiles, but fails to perform as expected
- Diagnose and fix common compilation errors
- Diagnose and fix the cause of an exception

Materials
Challenge 3 - Starter files (.zip)

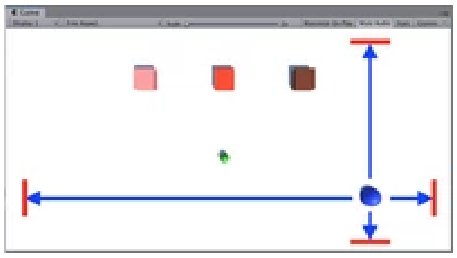
Steps
1. Overview
2. Warning
3. The player can't control the balloon
4. The background only moves when the game is over
5. No objects are being spawned
6. Fireworks appear to the side of the balloon
7. The background is not repeating properly
8. Bonus: The balloon can float way too high
9. Bonus: The balloon can drop below the ground

Lab 3 - Player control

Lesson link	Lab 3 - Player Control
Length	**1 hour 10 minutes**

Summary
In this lesson, you program the player's basic movement, including the code that limits that movement. Since there are a lot of different ways a player can move, depending on the type of project you're working on, you will not be given step-by-step instructions on how to do it. In order to do this, you will need to do research, reference other code, and problem-solve when things go wrong.

Project outcome
The player will be able to move around based on user input, but not be able to move where they shouldn't.



Skills
Basic application scripting
- Prototype new concepts

Steps:
1. Create PlayerController and plan your code
2. Basic movement from user input

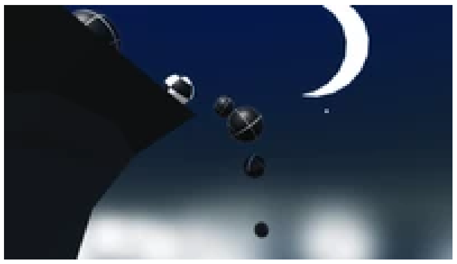
3. Constrain the player's movement 4. Code Cleanup and Export Backup 5. Lesson recap
New progress ● Player can move based on user input ● Player movement is constrained to suit the requirements of the game **New concepts & skills** ● Program in C# independently ● Troubleshoot issues independently

Quiz 3

Lesson link	Quiz 3	
Length	**15 minutes**	
Summary This quiz will assess your knowledge of the skills and concepts learned in Unit 3. **Quiz objective** This quiz will assess your ability to: ● Tweak the gravity of your project with Physics.gravity and use ForceModes to apply forces in different ways ● Utilize new operators like "&&" and bool variables to better control game logic ● Freeze or constrain the RigidBody component to halt movement on certain axes ● Use tags to label GameObjects and call them in the code ● Use script communication to access the methods and variables of other scripts ● Manage animation states in the Animator Controller, including adjusting the states' parameters and default state ● Use SetTrigger, SetBool, and SetInt methods to trigger transitions between animation states ● Stop and play particle effects to correspond with character animation states ● Work with Audio Sources and Listeners to play background music ● Add sound effects to add polish to your project		
Questions 1. You are trying to STOP spawning enemies when the player has died and have created the two scripts below to do that. However, there is an error on the underlined code, "isAlive" in the EnemySpawner script. What is causing that error? 2. Match the following animation methods with its set of parameters.		

3. Given the animation controller / state machine below, which code will make the character transition from the "Idle" state to the "Walk" state? 4. Which of these is the correct way to get a reference to an AudioSource component on a GameObject? 5. When you run a project with the code below, you get the following error: "NullReferenceException: Object reference not set to an instance of an object." What is most likely the problem? 6. Which of the following conditions properly tests that the game is NOT over and the player IS on the ground? 7. By default, what will be the first state used by this Animation Controller? 8. Which of the following variable declarations observes Unity's standard naming conventions (especially as it relates to capitalization)? 9. Which of the following is most likely the condition for the transition between "Run" and "Walk" shown below? 10. Which of the following do you think makes the most sense for a simple

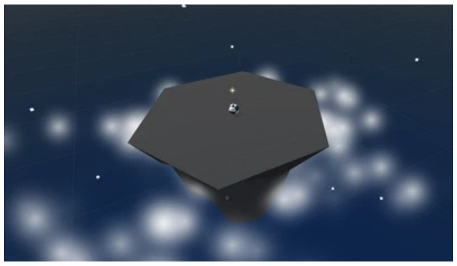
Unit 4 - Gameplay mechanics

Lesson link	Unit 4 - Gameplay mechanics	
Length	**7 hour 40 minutes**	
Summary In this Unit, you will program an arcade-style Sumo battle with the objective of knocking increasingly difficult waves of enemies off of a floating island, using power ups to help defeat them. In creating this prototype, you will learn how to implement new gameplay mechanics into your projects, which are new rules or systems that make the game more interesting to play. On one hand, you will learn to program a powerup, which gives the player a temporary advantage. On the other hand, you will learn to program increasingly difficult enemy waves, which make survival more challenging for the player. A good balance of powerups and increasing difficulty make for a much more interesting gameplay experience.		
Skills Basic application scripting ● Use common logic structures to control the execution of code. ● Write code that utilizes the various Unity APIs ● Write code that integrates into an existing system ● Implement a code style that is efficient and easy to read ● Prototype new concepts Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected ● Diagnose and fix common compilation errors ● Diagnose and fix the cause of an exception		

Unit 4 - Introduction

Lesson link	Unit 4 - Introduction
Length	**5 minutes**
Summary An introductory video for Unit 4, where you will learn to implement new Gameplay Mechanics. Watch the video here.	
Skills Basic application scripting ● Use common logic structures to control the execution of code. ● Write code that utilizes the various Unity APIs ● Write code that integrates into an existing system ● Implement a code style that is efficient and easy to read ● Prototype new concepts Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected ● Diagnose and fix common compilation errors ● Diagnose and fix the cause of an exception	

Lesson 4.1 - Watch where you're going

Lesson link	Lesson 4.1 - Watch where you're going
Length	**1 hour**
Summary First things first, we will create a new prototype and download the starter files! You'll notice a beautiful island, sky, and particle effect… all of which can be customized! Next you will allow the player to rotate the camera around the island in a perfect radius, providing a glorious view of the scene. The player will be represented by a sphere, wrapped in a detailed texture of your choice. Finally you will add force to the player, allowing them to move forwards or backwards in the direction of the camera. **Project outcome** The camera will evenly rotate around a focal point in the center of the island, provided a horizontal input from the player. The player	

will control a textured sphere, and move them forwards or backwards in the direction of the aforementioned focal point.	

Skills
Basic application scripting
- Write code that utilizes the various Unity APIs
- Write code that integrates into an existing system
- Implement a code style that is efficient and easy to read

Skills
Basic application scripting
- Write code that utilizes the various Unity APIs

Materials
Prototype 4 - Starter files (.zip)

Steps:
1. Create project and open scene
2. Set up the player and add a texture
3. Create a focal point for the camera
4. Rotate the focal point by user input
5. Add forward force to the player
6. Move in direction of focal point
7. Lesson recap

New functionality
- Camera rotates around the island based on horizontal input
- Player rolls in direction of camera based on vertical input

New concepts and skills
- Texture Wraps
- Camera as child object
- Global vs Local coordinates
- Get direction of other object

Lesson 4.2 - Follow the player

Lesson link	Lesson 4.2 - Follow the player
Length	**1 hour**

Summary The player can roll around to its heart's content... but it has no purpose. In this lesson, we fill that purpose by creating an enemy to challenge the player! First we will give the enemy a texture of your choice, then give it the ability to bounce the player away, potentially knocking them off the cliff. Lastly, we will let the enemy chase the player around the island and spawn in random positions. **Project outcome** A textured and spherical enemy will spawn on the island at start, in a random location determined by a custom function. It will chase the player around the island, bouncing them off the edge if they get too close.	
Skills Basic application scripting ● Write code that utilizes the various Unity APIs ● Write code that integrates into an existing system ● Implement a code style that is efficient and easy to read	
Steps: 1. Create project and open scene 2. Set up the player and add a texture 3. Create a focal point for the camera 4. Rotate the focal point by user input 5. Add forward force to the player 6. Move in direction of focal point 7. Lesson recap	
New functionality ● Camera rotates around the island based on horizontal input ● Player rolls in direction of camera based on vertical input **New concepts and skills** ● Texture Wraps ● Camera as child object ● Global vs Local coordinates ● Get direction of other object	

Lesson 4.3 - PowerUp and CountDown

Lesson link	[Lesson 4.3 - PowerUp and CountDown]()
Length	**1 hour**

Summary
The enemy chases the player around the island, but the player needs a better way to defend themselves, especially if we add more enemies. In this lesson, we're going to create a powerup that gives the player a temporary strength boost, shoving away enemies that come into contact! The powerup will spawn in a random position on the island, and highlight the player with an indicator when it is picked up. The powerup indicator and the powerup itself will be represented by stylish game assets of your choice.

Project outcome
A powerup will spawn in a random position on the map, eagerly awaiting the player. Once the player collides with this powerup, the powerup will disappear and the player will be highlighted by an indicator. The powerup will last for 5 seconds after pickup, granting the player super strength that blasts away enemie

Skills
Basic application scripting
- Write code that utilizes the various Unity APIs

Steps
1. Choose and prepare a powerup
2. Destroy powerup on collision
3. Test for enemy and powerup
4. Apply extra knockback with powerup
5. Create Countdown Routine for powerup
6. Add a powerup indicator
7. Lesson recap

New functionality
- When the player collects a powerup, a visual indicator appears
- When the player collides with an enemy while they have the powerup, the enemy goes flying
- After a certain amount of time, the powerup ability and indicator disappear

New concepts and skills
- Debug concatenation
- Local component variables
- IEnumerators and WaitForSeconds()
- Coroutines
- SetActive(true/false)

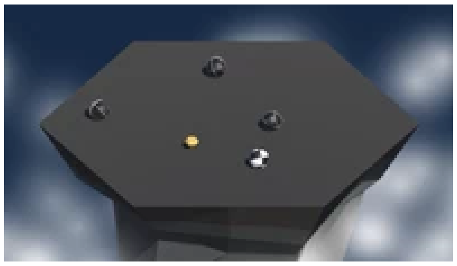
Lesson 4.4 - For-loops for waves

Lesson link	Lesson 4.4 - For-loops for waves

Length	**1 hour 10 minutes**

Summary
We have all the makings of a great game; A player that rolls around and rotates the camera, a powerup that grants super strength, and an enemy that chases the player until the bitter end. In this lesson we will wrap things up by putting these pieces together!

First we will enhance the enemy Spawn Manager, allowing it to spawn multiple enemies and increase their number every time a wave is defeated. Lastly we will spawn the powerup with every wave, giving the player a chance to fight back against the ever-increasing horde of enemies.



Project outcome
The Spawn Manager will operate in waves, spawning multiple enemies and a new powerup with each iteration. Every time the enemies drop to zero, a new wave is spawned and the enemy count increases.

Skills
Basic application scripting
- Use common logic structures to control the execution of code.
- Write code that utilizes the various Unity APIs
- Write code that integrates into an existing system

Steps
- Write a for-loop to spawn 3 enemies
- Give the for-loop a parameter
- Destroy enemies if they fall off
- Increase enemyCount with waves
- Spawn powerups with new waves
- Lesson recap

New functionality
- Enemies spawn in waves
- The number of enemies spawned increases after every wave is defeated
- A new power up spawns with every wave

New concepts and skills
- For-loops
- Increment (++) operator
- Custom methods with parameters
- FindObjectsOfType

Challenge 4 - Soccer scripting

Lesson link	Challenge 4 - Soccer scripting

Length	**1 hour**

Challenge overview
Use the skills you learned in the Sumo Battle prototype in a completely different context: the soccer field. Just like in the prototype, you will control a ball by rotating the camera around it and applying a forward force, but instead of knocking them off the edge, your goal is to knock them into the opposing net while they try to get into your net. Just like in the Sumo Battle, after every round a new wave will spawn with more enemy balls, putting your defense to the test. However, almost nothing in this project is functioning! It's your job to get it working correctly.

Challenge outcome
- Enemies move towards your net, but you can hit them to deflect them away
- Powerups apply a temporary strength boost, then disappear after 5 seconds
- When there are no more enemy balls, a new wave spawns with 1 more enemy

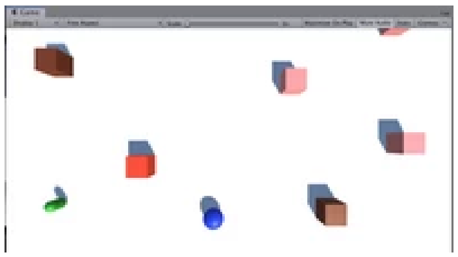
Skills
Basic application scripting
- Write code that utilizes the various Unity APIs
- Basic debugging
- Diagnose and fix code that compiles, but fails to perform as expected
- Diagnose and fix common compilation errors
- Diagnose and fix the cause of an exception

Materials
Challenge 4 - Starter files (.zip)

Summary
- Warning
- Hitting an enemy sends it back towards you
- A new wave spawns when the player gets a powerup
- The powerup never goes away
- 2 enemies are spawned in every wave
- The enemy balls are not moving anywhere
- Bonus: The player needs a turbo boost
- Bonus: The enemies never get more difficult

Lab 4 - Basic gameplay

Lesson link	Lab 4 - Basic gameplay
Length	**1 hour 10 minutes**

Summary 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.	
Skills Basic application scripting ● Prototype new concepts	
Steps 1. Give objects basic movement 2. Destroy objects off-screen 3. Handle object collisions 4. Make objects into prefabs 5. Make SpawnManager spawn Prefabs 6. Recap	
New functionality ● 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 & skills** ● Creating basic gameplay for a project independently	

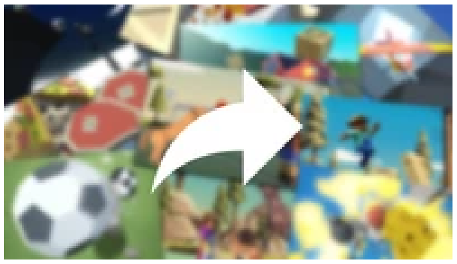
Quiz 4

Lesson link	[Quiz 4 - Unity Learn]()
Length	**15 minutes**

Summary This quiz will assess your knowledge of the skills and concepts learned in Unit 4. **Quiz objective** This quiz will assess your ability to: ● Apply Physics Materials to make game objects more or less bouncy ● Calculate new vectors to steer objects in custom directions ● Write more advanced custom functions and variables to make your code clean and professional ● Write informative debug messages with Concatenation ● Use IEnumerator and Coroutines to repeat and delay functions ● Use for loops to efficiently and dynamically run code multiple times ● Use SetActive to make game objects appear and disappear from the scene ● Use FindObjectsOfType to track the current number of objects in the scene	
Questions 1. You're trying to write some code that creates a random age between 1 and 100 and prints that age, but there is an error. What would fix the error? 2. The following message was displayed in the console: "Monica has 20 dollars". Which of the line options in the PrintNames function produced it? 3. The code below produces "error CS0029: Cannot implicitly convert type 'float' to 'UnityEngine.Vector3'". Which of the following would remove the error? 4. Which of the following follows Unity's naming conventions (especially as it relates to capitalization)? 5. You are trying to assign the powerup variable in the Inspector, but it is not showing up in the Player Controller component. What is the problem? 6. Your game has just started and you see the error, "UnassignedReferenceException: The variable playerIndicator of PlayerController has not been assigned." What is likely the solution to the problem? 7. You are trying to create a new method that takes a number and multiplies it by two. Which method would do that? 8. Which comment best describes the code below? 9. The code below produces the error, "error CS0029: Cannot implicitly convert type 'UnityEngine.GameObject' to 'UnityEngine.Rigidbody'". What could be done to fix this issue? 10. Which of the following statements about functions/methods are correct:	

Bonus features 4 - Share your work

Lesson link	Bonus features 4 - Share your work
Length	**1 hour**

Summary In this tutorial, you can go way above and beyond what you learned in this Unit and share what you've made with your fellow creators. There are four bonus features presented in this tutorial marked as Easy, Medium, Hard, and Expert. You can attempt any number of these, put your own spin on them, and then share your work! This tutorial is entirely optional, but highly recommended for anyone wishing to take their skills to a new level.	

Unit 5 - User Interface

Lesson link	[Unit 5 - User Interface]()
Length	**8 hours 10 minutes**
In this Unit, you will program a game to test the player's reflexes, where the goal is to click and destroy objects randomly tossed in the air before they can fall off the screen. In creating this prototype, you will learn how to implement a User Interface – or UI - into your projects. You will add a title screen with a difficulty select menu that will control how challenging the gameplay is, you will add a score display that will track how many points the player has earned, and you will add a Game Over screen, which will allow the player to restart and try again. In learning these skills, you will be able to create a fully "playable" experience that the user can enjoy from start to finish without having to restart the application to try it again.	
Skills Basic application scripting ● Use common logic structures to control the execution of code. ● Write code that utilizes the various Unity APIs ● Implement appropriate data types ● Write code that integrates into an existing system ● Implement a code style that is efficient and easy to read ● Prototype new concepts Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected	

Unit 5 - Introduction

Lesson link	[Unit 5 - Introduction]()
Length	**5 minutes**

An introductory video for Unit 5, where you will learn to implement a User Interface.	

Lesson 5.1 - Clicky mouse

Lesson link	Lesson 5.1 - Clicky mouse
Length	**1 hour**

Summary It's time for the final unit! We will start off by creating a new project and importing the starter files, then switching the game's view to 2D. Next we will make a list of target objects for the player to click on: Three "good" objects and one "bad". The targets will launch spinning into the air after spawning at a random position at the bottom of the map. Lastly, we will allow the player to destroy them with a click! **Project outcome** A list of three good target objects and one bad target object will spawn in a random position at the bottom of the screen, thrusting themselves into the air with random force and torque. These targets will be destroyed when the player clicks on them or they fall out of bounds.	

Skills
Basic application scripting
- Use common logic structures to control the execution of code.
- Write code that utilizes the various Unity APIs
- Implement appropriate data types
- Write code that integrates into an existing system
- Implement a code style that is efficient and easy to read

Materials
Prototype 5 - Starter files (.zip)

Steps:
1. Create project and switch to 2D view
2. Create good and bad targets
3. Toss objects randomly in the air
4. Replace messy code with new methods
5. Create object list in Game Manager
6. Create a coroutine to spawn objects

7. Destroy target with click and sensor
8. Lesson recap

New functionality
- Random objects are tossed into the air on intervals
- Objects are given random speed, position, and torque
- If you click on an object, it is destroyed

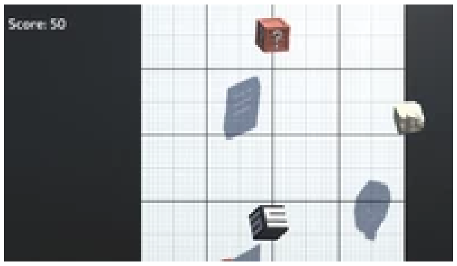
New concepts and skills
- 2D View
- AddTorque
- Game Manager
- Lists
- While Loops
- Mouse Events

Lesson 5.2 - Keeping score

Lesson link	[Lesson 5.2 - Keeping score]()
Length	**1 hour 10 minutes**

Summary
Objects fly into the scene and the player can click to destroy them, but nothing happens. In this lesson, we will display a score in the user interface that tracks and displays the player's points. We will give each target object a different point value, adding or subtracting points on click. Lastly, we will add cool explosions when each target is destroyed.

Project outcome
A "Score:" section will display in the UI, starting at zero. When the player clicks a target, the score will update and particles will explode as the target is destroyed. Each "good" target adds a different point value to the score, while the "bad" target subtracts from the score.

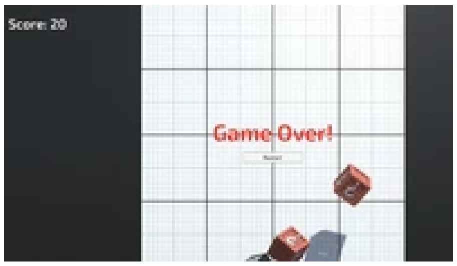


Skills
Basic application scripting
- Write code that utilizes the various Unity APIs

Steps
1. Add Score Text position it on screen
2. Edit the Score Text's properties
3. Initialize Score Text and variable
4. Create a new UpdateScore method

5. Add score when targets are destroyed 6. Assign a point value to each target 7. Add a Particle explosion 8. Lesson recap
New functionality ● Random objects are tossed into the air on intervals ● Objects are given random speed, position, and torque ● If you click on an object, it is destroyed **New concepts and skills** ● 2D View ● AddTorque ● Game Manager ● Lists ● While Loops ● Mouse Events

Lesson 5.3 - Game Over

Lesson link	Lesson 5.3 - Game Over	
Length	**1 hour 10 minutes**	
Summary We added a great score counter to the game, but there are plenty of other game-changing UI elements that we could add. In this lesson, we will create some "Game Over" text that displays when a "good" target object drops below the sensor. During game over, targets will cease to spawn and the score will be reset. Lastly, we will add a "Restart Game" button that allows the player to restart the game after they have lost. **Project outcome** When a "good" target drops below the sensor at the bottom of the screen, the targets will stop spawning and a "Game Over" message will display across the screen. Just underneath the "Game Over" message will be a "Reset Game" button that reboots the game and resets the score, so the player can enjoy it all over again.		
Skills Basic application scripting ● Write code that utilizes the various Unity APIs		
Steps 1. Create a Game Over text object		

2. Make GameOver text appear
3. Create GameOver function
4. Stop spawning and score on GameOver
5. Add a Restart button
6. Make the restart button work
7. Show restart button on game over
8. Lesson recap

New functionality
- A functional Game Over screen with a Restart button
- When the Restart button is clicked, the game resets

New concepts and skills
- Game states
- Buttons
- On Click events
- Scene management Library
- UI Library
- Booleans to control game states

Lesson 5.4 - What's the difficulty?

Lesson link	[Lesson 5.4 - What's the difficulty?]()
Length	**1 hour**

Summary
It's time for the final lesson! To finish our game, we will add a menu and title screen of sorts. You will create your own title, and style the text to make it look nice. You will create three new buttons that set the difficulty of the game. The higher the difficulty, the faster the targets spawn!

Project outcome
Starting the game will open to a beautiful menu, with the title displayed prominently and three difficulty buttons resting at the bottom of the screen. Each difficulty will affect the spawn rate of the targets, increasing the skill required to stop "good" targets from falling.

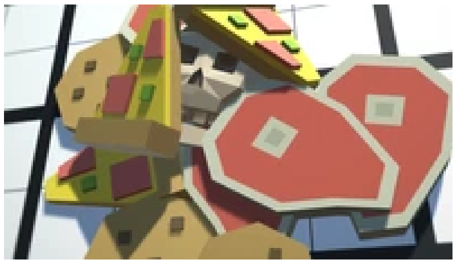
Skills
Basic application scripting
- Write code that utilizes the various Unity APIs
- Write code that integrates into an existing system

Steps

1. Create title text and menu buttons
2. Add a DifficultyButton script

3. Call SetDifficulty on button click 4. Make your buttons start the game 5. Deactivate title screen on StartGame 6. Use a parameter to change difficulty 7. Lesson recap
New functionality ● Title screen that lets the user start the game ● Difficulty selection that affects spawn rate **New concepts and skills** ● AddListener() ● Passing parameters between scripts ● Divide/Assign (/=) operator ● Grouping child objects

Challenge 5 - Whack-a-food

Lesson link	Challenge 5 - Whack-a-Food	
Length	**1 hour**	
Summary Put your User Interface skills to the test with this whack-a-mole-like challenge in which you have to get all the food that pops up on a grid while avoiding the skulls. You will have to debug buttons, mouse clicks, score tracking, restart sequences, and difficulty setting to get to the bottom of this one. **Challenge outcome** All of the buttons look nice with their text properly aligned. When you select a difficulty, the spawn rate changes accordingly. When you click a food, it is destroyed and the score is updated in the top-left. When you lose the game, a restart button appears that lets you play again.		
Skills Basic application scripting ● Write code that utilizes the various Unity APIs Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected		
Materials Challenge 5 - Starter files (.zip)		
Overview ● Warning ● The difficulty buttons look messy ● The food is being destroyed too soon		

- The score is being replaced by the word "score"
- When you lose, there's no way to restart
- The difficulty buttons don't change the difficulty
- Bonus: The game can go on forever

Lab 5 - Swap out your assets

Lesson link	Lab 5 - Swap out your assets
Length	1 hour 30 minutes
Summary In this lab, you will finally replace those boring primitive objects with beautiful dynamic ones. You will either use assets from the provided course library or browse the asset store for completely new ones to give your game exactly the look and feel that you want. Then, you will go through the process of actually swapping in those new assets in the place of your placeholder primitives. By the end of this lab, your project will be looking a lot better. Project outcome All primitive objects are replaced by actual 3D models, retaining the same basic gameplay functionality.	
Skills Basic application scripting ● Prototype new concepts	
Materials Create with code - Course library (.zip)	
Steps 1. Import and browse the asset library 2. Replace player with new asset 3. Browse the Asset Store 4. Replace all non-player primitives 5. Replace the background texture 6. Recap	

Quiz 5

Lesson link	Lab 5 - Swap out your assets
Length	**1 hour 30 minutes**

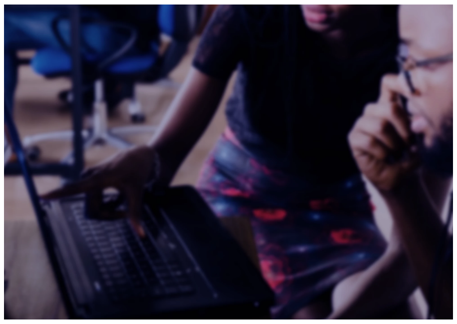
Quiz Objective This quiz will assess your ability to: ● Create a Game Manager object that controls game states ● Detect where the user has clicked their mouse in order to create a click-based program ● Use the Canvas to create UI elements like a Title, Buttons, or score display ● Lock elements and objects into place on the UI with Anchors ● Use variables and script communication to update elements in the UI ● Make UI elements appear and disappear with .SetActive ● Use script communication and Game states to implement working “Game Over” screen ● Restart the game using a UI button and SceneManagement class ● Add listeners to detect when a UI Button has been clicked and trigger functionality ● Set difficulty of gameplay from title screen by passing parameters between scripts	
Questions 1. Which of the following follows Unity naming conventions (especially as they relate to capitalization)? 2. If there is a boolean in script A that you want to access in script B, which of the following are true? 3. Which code to fill in the blank will result in the object being destroyed? 4. You run your game and get the following error message in the console, “NullReferenceException: Object reference not set to an instance of an object”. Given the image and code below, what would resolve the problem? 5. Read the Unity documentation below about the OnMouseDrag event and the code beneath it. What will the value of the “counter” variable be if the user clicked and held down the mouse over an object with a collider for 10 seconds? 6. Based on the code below, what will be displayed in the console when the button is clicked? 7. You have declared a new Button variable as “private Button start;”, but there’s an error under the word “Button” that says “error CS0246: The type or namespace name 'Button' could not be found (are you missing a using directive or an assembly reference?)” What is likely causing that error? 8. Look at the documentation and code below. Which of the following lines would NOT produce an error? 9. If you wanted a button to display the message “Hello!” when a button was clicked, what code would you use to fill in the blank? 10. Which of the following is the correct way to declare a new List of game objects?	

Bonus features 5 - Share your work

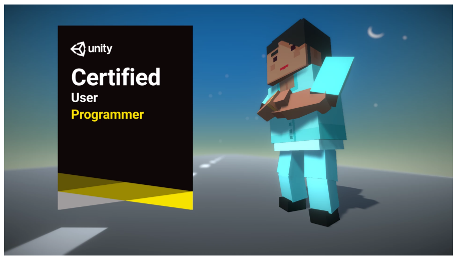
Lesson link	[Bonus features 5 - Share your work]()

Length	**1 hour**
Summary In this tutorial, you can go way above and beyond what you learned in this Unit and share what you've made with your fellow creators. There are four bonus features presented in this tutorial marked as Easy, Medium, Hard, and Expert. You can attempt any number of these, put your own spin on them, and then share your work! This tutorial is entirely optional, but highly recommended for anyone wishing to take their skills to a new level.	

Introduction to user feedback and testing

Lesson link	[Introduction to user feedback and testing]()
Length	**20 minutes**
In this tutorial, you'll explore: ● The purpose of user feedback ● How user testing can be integrated into your design and development process ● Tips for running feedback sessions and getting actionable feedback from your target users	
Skills Communication & collaboration ● Participating in listening, constructive feedback cycles, and peer review Basic design process ● Coordinate a user feedback and testing session	
Steps 1. Overview 2. What is user testing and feedback? 3. Why conduct structured user testing? 4. The phases of user testing 5. Planning for user testing 6. Preparing yourself to run the session 7. Facilitating the session 8. After the session 9. Exercise: Mini product evaluation 10. Summary and next steps	

Next steps

Lesson link	[Next steps]()	
Length	**2 hours 15 minutes**	
Summary In this final Unit, you will prepare to continue developing your Unity and C# skills independently, including learning how to complete your personal project and how to prepare for the Unity Certified User: Programmer exam, which you can take through your school program.		
Skills Basic application scripting ● Implement appropriate data types ● Implement a code style that is efficient and easy to read ● Prototype new concepts Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected ● Diagnose and fix compilation errors related to Unity's Scripting API		
Steps 11. Overview 12. What is user testing and feedback? 13. Why conduct structured user testing? 14. The phases of user testing 15. Planning for user testing 16. Preparing yourself to run the session 17. Facilitating the session 18. After the session 19. Exercise: Mini product evaluation 20. Summary and next steps		

Next steps: Introduction

Lesson link	[Next steps - Introduction]()
Length	**5 minutes**

Summary
A video introducing the final section of the course, where you will learn to prepare for the Unity Certified User Exam and to complete and share your own Personal Project.

Skills
Basic application scripting
- Implement appropriate data types
- Implement a code style that is efficient and easy to read
- Prototype new concepts

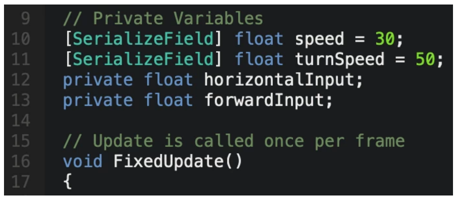
Basic debugging
- Diagnose and fix code that compiles, but fails to perform as expected
- Diagnose and fix compilation errors related to Unity's Scripting API

Lesson 6.1 - Project optimization

Lesson link	[Lesson 6.1 - Project optimization]
Length	**30 minutes**

Summary
In this lesson, you will learn about a variety of different techniques to optimize your projects and make them more performant. You may not notice a huge difference in these small prototype projects, but when you're exporting a larger project, especially one for mobile or web, every bit of performance improvement is critical.

Project outcome
Several of your prototype projects will have improved optimization, serving as examples for you to implement in your personal projects.



```
// Private Variables
[SerializeField] float speed = 30;
[SerializeField] float turnSpeed = 50;
private float horizontalInput;
private float forwardInput;

// Update is called once per frame
void FixedUpdate()
{
```

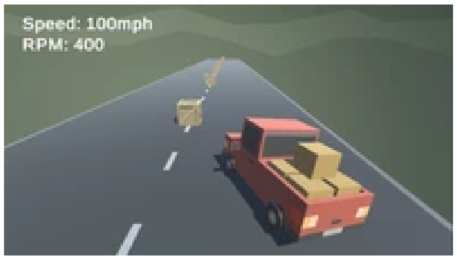
Skills
Basic application scripting
- Implement appropriate data types
- Implement a code style that is efficient and easy to read

Materials
Object pooling (.zip)

Steps:
1. Variable attributes
2. Unity Event Functions
3. Object Pooling
4. Lesson recap

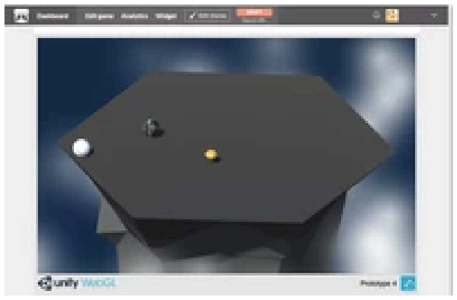
New Concepts and Skills 1. Optimization 2. Serialized Fields 3. readonly / const / static / protected 4. Event Functions 5. FixedUpdate() vs. Update() vs. LateUpdate() 6. Awake() vs. Start() 7. Object Pooling

Lesson 6.2 - Research and troubleshooting

Lesson link	Lesson 6.2 - Research and troubleshooting
Length	**1 hour**
Summary In this lesson, you will attempt to add a speedometer and RPM display for your vehicle in Prototype 1. In doing so, you will learn the process of doing online research when trying to implement new features and troubleshoot bugs in your projects. As you will find out, adding a new feature is very rarely as simple as it initially seems – you inevitably run into unexpected complications and errors that usually require a little online research. In this lesson, you will learn how to do that so that you can do it with your own projects. **Project outcome** By the end of this lesson, the vehicle will behave with more realistic physics, and there will be a speedometer and Revolution per Minute (RPM) display.	
Skills Basic application scripting ● Prototype new concepts Basic debugging ● Diagnose and fix code that compiles, but fails to perform as expected ● Diagnose and fix compilation errors related to Unity's Scripting API	
Steps ● Make the vehicle use forces ● Prevent car from flipping over ● Add a speedometer display ● Add an RPM display ● Prevent driving in mid-air	

• Lesson recap
New concepts and skills • Searching on Unity Answers, Forum, Scripting API • Troubleshooting to resolve bugs • AddRelativeForce, Center of Mass, RoundToInt • Modulus/Remainder (%) operator • Looping through lists • Custom methods with bool return

Lesson 6.3 - Sharing your projects

Lesson link	[Lesson 6.3 - Sharing your projects]()
Length	**30 minutes**
Summary In this lesson, you will learn how to build your projects so that they're playable outside of the Unity interface. First, you will install the necessary export modules to be able to publish your projects. After that, you will build your project as a standalone app to be played on Mac or PC computers. Finally, you will export your project for WebGL and even upload it to a game sharing site so that you can send it to your friends and family. **Project outcome** Your project will be exported and playable as a standalone app on Mac/PC or for embedding online.	
Steps • Install export Modules • Build your game for Mac or Windows • Build your game for WebGL • Lesson recap	
New concepts and skills • Installing export modules • Building for Mac/PC • Building for WebGL/HTML	

ECS Survival Guide

Lesson link	ECS Survival Guide
Length	**10 minutes**

Summary
In this tutorial, you will learn at a very high level about the Entity Component System (ECS) and how it is different from the typical object-oriented or component-based system you are probably familiar with.

If you are preparing to take the Unity Certified User: Programmer exam, or just want to know what ECS is at a very high level, this will be a very helpful tutorial.

Steps:
- Overview
- What is ECS and what does it look like?
- Recap

Career research and preparation

Lesson link	Career research and preparation
Length	**25 minutes**

Overview
In this tutorial, you'll research career areas, job titles, and the necessary certifications and qualifications to help you identify career goals. This will help you to determine the focus and goals for your portfolio.

In this tutorial, you'll begin to:
- Explore the career choices available for you to apply the skills you're developing in a professional role
- Identify the key skills and qualifications required to be hired in these roles

Skills
Basic job preparation
- Prepare yourself for the job search

Materials Video transcript - Career research and preparation (.zip)
Steps: 1. Overview 2. Research your career options 3. Certification 4. What do employers look for? 5. Exercise: Create a career research document

Introduction to portfolios

Lesson link	Introduction to portfolios
Length	**25 minutes**
Summary In this tutorial, you'll: ● Review the goals and uses of a portfolio ● Plan your own portfolio using a flowchart ● Begin to select and organize content	
Skills Basic job preparation ● Create a portfolio, enabling you to pursue a job in real-time development	
Steps 1. Overview 2. Review portfolio types and tools 3. What goes in a portfolio?	

Mission 2 checkpoint

Quiz: Create with code 2