

# Junior Programmer Pathway



## Mission 1: Create with Code 1

### 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:		
	<b>Junior Programmer: Create with Code 1</b>	13 hours and 45 Minutes
	<b>Junior Programmer: Create with Code 2</b>	24 hours and 15 Minutes
	<b>Junior Programmer: Manage scene flow and data</b>	2 hours
	<b>Junior Programmer: Apply object-oriented principles</b>	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

Pathway Description	1
Table of contents	2
Mission 1: Create with code 1	3
Getting started	3
Course introduction	4
Install Unity software	4
Unit 1 - Player Control	5
Unit 1 - Introduction	6
Lesson 1.1 - Start your 3D engines	6
Lesson 1.2 - Pedal to the metal	7
Lesson 1.3 - High speed chase	8
Lesson 1.4 - Step into the driver's seat	9
Challenge 1 - Plane programming	10
Lab 1 - Project Design Document	11
Quiz 1	11
Bonus Features 1 - Share your work	12
Introduction to project management and teamwork	13
Unit 2 - Basic gameplay	14
Unit 2 - Introduction	14
Lesson 2.1 - Player positioning	15
Lesson 2.2 - Food fight	16
Lesson 2.3 - Random animal stampede	17
Lesson 2.4 - Collision decisions	18
Challenge 2 - Play fetch	19
Lab 2 - New project with primitives	20
Quiz 2	21
Bonus features 2 - Share your work	22
Mission 1 checkpoint	22

# Mission 1: Create with Code 01

Part of the [Junior Programmer Pathway](#)

## Mission overview


This first mission in the Junior Programmer pathway will provide you with the core foundation needed to create a wide range of digital experiences in Unity. You'll learn about fundamental programming concepts such as variables, functions and basic logic through two practical projects. You'll also modify a script to customize a simple Unity experience: the beginning of your personal portfolio.



## 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 Live calendar for follow-along practical sessions with Unity established creators.
- When you've finished the mission, you'll get the mission badge for your profile and portfolio.

## Getting started

Lesson link	<a href="#">Getting started</a>	
Length	1 hour 30 minutes	
Lesson objective	<p>In this introductory <b>Unit</b>, you will be introduced to the course, then you will download and install the Unity software.</p> <p>This lesson is part of the <a href="#">Junior Programmer Pathway</a>.</p>	
		

## Course introduction

Lesson link	<a href="#">Course introduction</a>
Length	10 minutes

### Summary


In this lesson, you will be introduced to the course, its goals, what it involves, and how you can get the most out of it.



### Steps


1. Welcome
2. Watch, then do

## Install Unity software


Lesson link	<a href="#">Install Unity software</a>	
Length	25 minutes	
<b>Summary</b> If you do not already have the recommended version of Unity installed on your computer, the first thing you need to do before you get started on the course is install it. In order to do so, if you don't have one already, you will need to create a Unity ID. When you install the software, you will install Unity Hub, which allows you to manage your installations and projects, the Unity engine itself, and Visual Studio, the Integrated Development Environment (or IDE) you will use to code in C#.		
<b>Project outcome</b> Unity Hub, the Unity Editor, and Visual Studio will all be installed on your computer.		
<b>Steps:</b> <ol style="list-style-type: none"><li>1. Before you begin</li><li>2. Download and install Unity Hub</li><li>3. Install a new version of Unity</li><li>4. Sign in or create a new Unity ID</li><li>5. Recap</li></ol>		
<b>New progress</b> <ul style="list-style-type: none"><li>• Installed Unity Hub, the correct Unity Editor version, and Visual Studio</li><li>• Signed into your Unity account</li></ul>		
<b>New concepts</b> <ul style="list-style-type: none"><li>• Unity Hub and its features</li><li>• Editor versions, including LTS releases</li></ul>		

- Visual Studio
- Unity IDs


## Unit 1 - Player Control

<b>Lesson link</b>	<a href="#">Unit 1 - Player control</a>
<b>Length</b>	<b>7 hours 30 minutes</b>
<p><b>Summary</b> In this Unit, you will program a car moving side-to-side on a floating road, trying to avoid (or hit) obstacles in the way. In addition to becoming familiar with the Unity Editor and workflow, you will learn how to create new C# scripts and do some simple programming. By the end of the Unit, you will be able to call basic functions, then declare and tweak new variables to modify the results of those functions.</p> <p><b>Skills</b> Basic application scripting</p> <ul style="list-style-type: none"> <li>• Implement appropriate data types</li> <li>• Implement a code style that is efficient and easy to read</li> <li>• Prototype new concepts</li> </ul> <p>Basic code comprehension</p> <ul style="list-style-type: none"> <li>• Interpret simple code</li> <li>• Improve simple code using the features of an IDE</li> </ul> <p>Basic debugging</p> <ul style="list-style-type: none"> <li>• Diagnose and fix code that compiles, but fails to perform as expected</li> </ul>	
	


## Unit 1 - Introduction

<b>Lesson link</b>	<a href="#">Unit 1 - Introduction</a>
<b>Length</b>	<b>5 minutes</b>
<p>An introductory video for Unit 1, where you will learn to implement player control.</p>	
<p><b>Steps</b></p> <ol style="list-style-type: none"> <li>1. Introduction</li> </ol>	
	

# Lesson 1.1 - Start your 3D engines


Lesson link	<a href="#">Lesson 1.1 - Start your 3D engines</a>	
Length	1 hour 20 minutes	
<div><b>Summary</b> In this lesson, you will create your very first game project in Unity Hub. You will choose and position a vehicle for the player to drive and an obstacle for them to hit or avoid. You will also set up a camera for the player to see through, giving them a perfect view of the scene. Throughout this process, you will learn to navigate the Unity Editor and grow comfortable moving around in 3D space. Lastly, you will customize your own window layout for the Unity Editor.</div> <div><b>Project outcome</b> You will have a vehicle and obstacle positioned on the road and the camera set up perfectly behind the vehicle. You will also have a new custom Unity layout, perfectly optimized for editing.</div> <div><b>Materials</b> <a href="#">Prototype 1 - Starter files</a> (.zip)</div>		
<div><b>Steps</b><ol style="list-style-type: none"><li>1. Make a course folder and new project</li><li>2. Import assets and open Prototype 1</li><li>3. Add your vehicle to the scene</li><li>4. Add an obstacle and reposition it</li><li>5. Locate your camera and run the game</li><li>6. Move the camera behind the vehicle</li><li>7. Customize the interface layout</li><li>8. Lesson recap</li></ol></div>		
<div><b>New functionality</b><ul style="list-style-type: none"><li>• Project set up with assets imported</li><li>• Vehicle positioned at the start of the road</li><li>• Obstacle positioned in front of the vehicle</li><li>• Camera positioned behind vehicle</li></ul></div> <div><b>New concepts &amp; skills</b><ul style="list-style-type: none"><li>• Create a new project</li><li>• Import assets</li><li>• Add objects to the scene</li><li>• Game vs Scene view</li><li>• Project, Hierarchy, Inspector windows</li><li>• Navigate 3D space</li><li>• Move and Rotate tools</li><li>• Customize the layout</li></ul></div>		

## Lesson 1.2 - Pedal to the metal


Lesson link	<a href="#">Lesson 1.2 - Pedal to the metal</a>	
Length	1 hour 10 minutes	
<div><b>Summary</b> In this lesson you will make your driving simulator come alive. First you will write your very first lines of code in C#, changing the vehicle's position and allowing it to move forward. Next you will add physics components to your objects, allowing them to collide with one another. Lastly, you will learn how to duplicate objects in the hierarchy and position them along the road.</div> <div><b>Project outcome</b> You will have a moving vehicle with its own C# script and a road full of objects, all of which may collide with each other using physics components.</div> <div><b>Skills</b> Basic code comprehension<ul style="list-style-type: none"><li>• Interpret simple code</li><li>• Improve simple code using the features of an IDE</li></ul>Basic application scripting<ul style="list-style-type: none"><li>• Implement a code style that is efficient and easy to read</li></ul></div>		
<div><b>Steps</b><ol style="list-style-type: none"><li>1. Create and apply your first script</li><li>2. Add a comment in the Update() method</li><li>3. Give the vehicle a forward motion</li><li>4. Use a Vector3 to move forward</li><li>5. Customize the vehicle's speed</li><li>6. Add Rigidbody components to objects</li><li>7. Duplicate and position the obstacles</li><li>8. Lesson recap</li></ol></div> <div><b>New functionality</b><ul style="list-style-type: none"><li>• Vehicle moves down the road at a constant speed</li><li>• When the vehicle collides with obstacles, they fly into the air</li></ul><b>New concepts &amp; skills</b><ul style="list-style-type: none"><li>• C# scripts</li><li>• Start vs Update</li><li>• Comments</li><li>• Methods</li><li>• Pass parameters</li><li>• Time.deltaTime</li><li>• Multiply (*) operator</li><li>• Components</li><li>• Collider and Rigidbody</li></ul></div>		




## Lesson 1.3 - High speed chase

Lesson link	<a href="#">Lesson 1.3 - High speed chase</a>	
Length	50 minutes	
<b>Summary</b> Keep your eyes on the road! In this lesson you will code a new C# script for your camera, which will allow it to follow the vehicle down the road and give the player a proper view of the scene. In order to do this, you'll have to use a very important concept in programming: variables.  <b>Project outcome</b> The camera will follow the vehicle down the road through the scene, allowing the player to see where it's going.  <b>Skills</b> Basic code comprehension <ul style="list-style-type: none"><li>• Interpret simple code</li><li>• Improve simple code using the features of an IDE</li></ul> Basic application scripting <ul style="list-style-type: none"><li>• Implement appropriate data types</li><li>• Implement a code style that is efficient and easy to read</li></ul>		
<b>Steps</b> <ol style="list-style-type: none"><li>1. Add a speed variable for your vehicle</li><li>2. Create a new script for the camera</li><li>3. Add an offset to the camera position</li><li>4. Make the offset into a Vector3 variable</li><li>5. Smooth the Camera with LateUpdate</li><li>6. Edit the playmode tint color</li><li>7. Lesson recap</li></ol> <b>New functionality</b> <ul style="list-style-type: none"><li>• Camera follows the vehicle down the road at a set offset distance</li></ul> <b>New concepts &amp; skills</b> <ul style="list-style-type: none"><li>• Variables</li><li>• Data types</li><li>• Access Modifiers</li><li>• Declare and initialize variables</li><li>• Next lesson</li></ul>		

## Lesson 1.4 - Step into the driver's seat


Lesson link	<a href="#">Lesson 1.4 - Step into the driver's seat</a>	
Length	50 minutes	
<b>Summary</b> In this lesson, we need to hit the road and gain control of the vehicle. In order to do so, we need to detect when the player is pressing the arrow keys, then accelerate and turn the vehicle based on that input. Using new methods, Vectors, and variables, you will allow the vehicle to move forwards or backwards and turn left to right.		
<b>Project outcome</b> When the player presses the up/down arrows, the vehicle will move forward and backward. When the player presses the left/right arrows, the vehicle will turn.		
<b>Skills</b> Basic code comprehension <ul style="list-style-type: none"><li>• Interpret simple code</li><li>• Improve simple code using the features of an IDE</li></ul>		
<b>Steps</b> <ol style="list-style-type: none"><li>1. Allow the vehicle to move left/right</li><li>2. Base left/right movement on input</li><li>3. Take control of the vehicle speed</li><li>4. Make vehicle rotate instead of slide</li><li>5. Clean your code and hierarchy</li><li>6. Lesson recap</li></ol>		
<b>New functionality</b> <ul style="list-style-type: none"><li>• When the player presses the up/down arrows, the vehicle will move forward and backward</li><li>• When the player presses the left/right arrows, the vehicle turns</li></ul>		
<b>New concepts &amp; skills</b> <ul style="list-style-type: none"><li>• Empty objects</li><li>• Get user input</li><li>• Translate vs Rotate</li></ul>		

## Challenge 1 - Plane programming


Lesson link	<a href="#">Challenge 1 - Plane programming</a>	
Length	50 minutes	
<b>Challenge overview</b> Use the skills you learned in the driving simulation to fly a plane around obstacles in the sky. You will have to get the user's input from the up and down arrows in order to control the plane's pitch up and down. You will also have to make the camera follow alongside the plane so you can keep it in view.		
<b>Challenge outcome</b> <ul style="list-style-type: none"><li>• The plane moves forward at a constant rate</li><li>• The up/down arrows tilt the nose of the plane up and down</li><li>• The camera follows along beside the plane as it flies</li></ul>		
<b>Materials</b> <a href="#">Challenge 1 - Starter files</a> (.zip)		
<b>Skills</b> Basic code comprehension <ul style="list-style-type: none"><li>• Interpret simple code</li><li>• Improve simple code using the features of an IDE</li></ul> Basic debugging <ul style="list-style-type: none"><li>• Diagnose and fix code that compiles, but fails to perform as expected</li></ul>		
<b>Steps</b> <ol style="list-style-type: none"><li>1. Challenge 1 overview</li><li>2. Warning</li><li>3. The plane is going backward</li><li>4. The plane is going too fast</li><li>5. The plane is tilting automatically</li><li>6. The camera is in front of the plane</li><li>7. The camera is not following the plane</li><li>8. Bonus: The plane's propeller does not spin</li></ol>		

## Lab 1 - Project Design Document

Lesson link	<a href="#">Challenge 1 - Plane programming</a>
Length	50 minutes

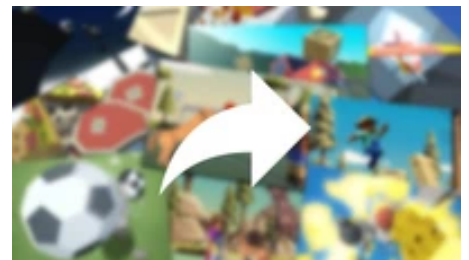
<p><b>Summary</b></p> <p>In this first ever Lab session, you will begin the preliminary work required to successfully create a personal project in this course. First, you'll learn what a personal project is, what the goals for it are, and what the potential limitations are. Then you will take the time to come up with an idea and outline it in detail in your Design Document, including a timeline for when you hope to complete certain features. Finally, you will take some time to draw a sketch of your project to help you visualize it and share your idea with others.</p> <p><b>Project outcome</b></p> <p>The Design Document will be filled out, including the concept, the timeline, and a preliminary sketch of the minimum viable product.</p>	
<p><b>Materials</b></p> <p><a href="#">Project Design Doc</a> (Google Doc)  <a href="#">Project Design Doc</a> (Word document)  <a href="#">Project Design Doc</a> (PDF)</p>	
<p><b>Skills</b></p> <p>Basic application scripting</p> <ul style="list-style-type: none"> <li>• Prototype new concepts</li> </ul>	
<p><b>Steps</b></p> <ol style="list-style-type: none"> <li>1. Understand what a personal project is</li> <li>2. Review Design Doc examples</li> <li>3. Complete your Project Concept V1</li> <li>4. Complete your Project Timeline</li> <li>5. Complete your MVP sketch</li> <li>6. Recap</li> </ol>	

## Quiz 1

Lesson link	<a href="#">Quiz 1</a>
Length	15 minutes
<p><b>Summary</b></p> <p>This quiz will assess your knowledge of the skills and concepts learned in Unit 1.</p> <p><b>Quiz objective</b></p> <p>This quiz will assess your ability to:</p> <ul style="list-style-type: none"> <li>• Navigate 3D space and the Unity Editor comfortably</li> <li>• Add and manipulate objects in the scene to position them where you want, including the camera</li> </ul>	

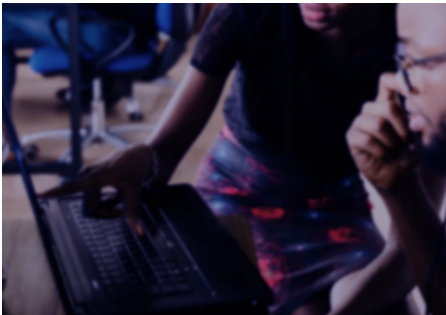
<ul style="list-style-type: none"> <li>• Control the layout of Unity Editor to suit your needs</li> <li>• Create C# scripts and apply them to objects</li> <li>• Use Visual Studio and a few of its basic features</li> <li>• Add Rigidbody and Collider components to allow objects to collide realistically</li> <li>• Declare variables properly and initialize/assign them with appropriate access modifiers</li> </ul>	
<b>Questions</b> <ol style="list-style-type: none"> <li>1. Which Unity window contains a list of all the game objects currently in your scene?</li> <li>2. True or False: Visual Studio is not a part of Unity. You could use a different code editor to edit your C# scripts if you wanted to.</li> <li>3. What best describes the difference between the below images, where the car in the second image is further along the road? Image 1 Image 2</li> <li>4. In what order do you put the words when you are declaring a new variable?</li> <li>5. Which of the following variables would be visible in the Inspector?</li> <li>6. What is a possible value for the horizontalInput variable?</li> <li>7. What is true about the following two lines of code?</li> <li>8. Which of the following lines of code is using standard Unity naming conventions?</li> <li>9. Which comment would best describe the code below?</li> <li>10. The image below shows the preferences window that allows you to...</li> </ol>	

## Bonus Features 1 - Share your work

<b>Lesson link</b>	<a href="#">Bonus features 1 - Share your work</a>
<b>Length</b>	<b>1 hour</b>
<b>Summary</b> <p>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.</p> <p>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!</p> <p>This tutorial is entirely optional, but highly recommended for anyone wishing to take their skills to a new level.</p>	
<b>Steps</b> <ol style="list-style-type: none"> <li>1. Overview</li> <li>2. Easy: Obstacle pyramids</li> </ol>	

3. Medium: Oncoming vehicles
4. Hard: Camera switcher
5. Expert: Local multiplayer
6. Solution walkthrough
7. Share your work

## Introduction to project management and teamwork

Lesson link	<a href="#">Introduction to project management and teamwork</a>	
Length	20 minutes	
<div><b>Summary</b> Most jobs in the design and development world require teamwork and will use project management and planning tools to ensure successful project delivery. In this tutorial, you'll be introduced to key concepts and best practices used for project planning, production phases, project management, and working in teams.</div> <div><b>Skills</b> Basic project management<ul style="list-style-type: none"><li>• Plan projects in the real-time development cycle</li><li>• Manage projects in the real-time development cycle</li></ul></div> <div><b>Materials</b><ul style="list-style-type: none"><li>• <a href="#">Example of Project Charter</a> (.pdf)</li><li>• <a href="#">Project Design Doc template</a> (.pdf)</li></ul></div>		<div></div>
<div><b>Steps</b><ol style="list-style-type: none"><li>1. Overview</li><li>2. Recap the phases of production</li><li>3. Overview of project planning</li><li>4. Design documents and project plans</li><li>5. Managing projects and tracking progress</li><li>6. Exercise: Identify tools to support your project management and tracking</li><li>7. Preparing to publish</li><li>8. Releasing your project</li><li>9. Operations activities and retrospectives</li><li>10. Summary</li></ol></div>		

## Unit 2 - Basic gameplay

<b>Lesson link</b>	<a href="#">Basic gameplay</a>
<b>Length</b>	<b>6 hours 20 minutes</b>

### Summary

In this Unit, you will program a top-down game with the objective of throwing food to hungry animals – who are stampeding towards you – before they can run past you. In order to do this, you will become much more familiar with some of the most important programming and Unity concepts, including if-then statements, random value generation, arrays, collision detection, Prefabs, and instantiation. In completing this Unit, you will learn how to program a basic game with the ability to launch projectiles and maneuver the player to keep the game alive.

### What you will learn

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.
- 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 2 - Introduction

Lesson link	<a href="#">Unit 2 - Introduction</a>	
Length	15 minutes	
Summary	<p>An introductory video for Unit 2, where you will learn to implement Basic Gameplay.</p> <p>Watch the Introduction video <a href="#">here</a>.</p>	
Steps	1. Unit 2 - Introduction	

## Lesson 2.1 - Player positioning

Lesson link	<a href="#">Lesson 2.1 - Player positioning</a>	
Length	1 hour	
Summary	<p>You will begin this unit by creating a new project for your second Prototype and getting basic player movement working. You will first choose which character you would like, which types of animals you would like to interact with, and which food you would like to feed those animals. You will give the player basic side-to-side movement just like you did in Prototype 1, but then you will use if-then statements to keep the Player in bounds.</p>	
Project outcome	<p>The player will be able to move left and right on the screen based on the user's left and right key presses, but will not be able to leave the play area on either side.</p>	
Skills	<p>Basic code comprehension</p> <ul style="list-style-type: none"><li>• Interpret simple code</li><li>• Improve simple code using the features of an IDE</li></ul> <p>Basic application scripting</p> <ul style="list-style-type: none"><li>• Use common logic structures to control the execution of code</li></ul>	
Materials	<p><a href="#">Prototype 2 - Starter files</a> (.zip)</p>	
Steps	<ol style="list-style-type: none"><li>1. Create a new Project for Prototype 2</li><li>2. Add the player, animals, and food</li><li>3. Get the user's horizontal input</li><li>4. Move the player left-to-right</li><li>5. Keep the player inbounds</li><li>6. Clean up your code and variables</li><li>7. Lesson recap</li></ol>	



## Lesson 2.2 - Food fight

Lesson link	<a href="#">Lesson 2.2 - Food fight</a>
Length	1 hour 10 minutes



## Summary

In this lesson, you will allow the player to launch the projectile through the scene. First you will write a new script to send the projectile forwards. Next you will store the projectile along with all of its scripts and properties using an important new concept in Unity called Prefabs. The player will be able to launch the projectile Prefab with a tap of the spacebar. Finally, you will add boundaries to the scene, removing any objects that leave the screen.

## Project outcome

The player will be able to press the spacebar and launch a projectile Prefab into the scene, which destroys itself when it leaves the game's boundaries. The animals will also be removed from the scene when they leave the game boundaries.

## Skills

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



## Steps

1. Make the projectile fly forwards
2. Make the projectile into a prefab
3. Test for spacebar press
4. Launch projectile on spacebar press
5. Make animals into prefabs
6. Destroy projectiles offscreen
7. Destroy animals offscreen
8. Lesson recap

## New functionality

- The player can move left and right based on the user's left and right key presses
- The player will not be able to leave the play area on either side

## New concepts & skills

- Adjust object scale
- If-statements
- Greater/less-than operators

## Lesson 2.3 - Random animal stampede

Lesson link	<a href="#">Lesson 2.3 - Random animal stampede</a>
Length	1 hour

## Summary

Our animal Prefabs walk across the screen and get destroyed out of bounds, but they don't actually appear in the game unless we drag them in! In this lesson we will allow the animals to spawn on their own, in a random location at the top of the screen. In order to do so, we will create a new object and a new script to manage the entire spawning process.

## Project outcome

When the user presses the S key, a randomly selected animal will spawn at a random position at the top of the screen, walking towards the player.

## Skills

Basic code comprehension

- Interpret simple code
- Improve simple code using the features of an IDE

Basic application scripting

- Implement appropriate data types

## Materials

[Lesson Plan 2.3 - Random animal stampede \(.pdf\)](#)



## Steps

1. Create a spawn manager
2. Spawn an animal if S is pressed
3. Spawn random animals from array
4. Randomize the spawn location
5. Change the perspective of the camera
6. Lesson recap

## New functionality

- The player can press the S to spawn an animal
- Animal selection and spawn location are randomized
- Camera projection (perspective/orthographic) selected


## New concepts & skills

- Spawn Manager
- Arrays
- Keycodes
- Random generation
- Local vs Global variables
- Perspective vs Isometric projections

## Lesson 2.4 - Collision decisions

Lesson link

[Lesson 2.4 - Collision decisions](#)

<b>Length</b>	<b>50 minutes</b>	
<p><b>Summary</b></p> <p>Our game is coming along nicely, but there are some critical things we must add before it's finished. First off, instead of pressing S to spawn the animals, we will spawn them on a timer so that they appear every few seconds. Next we will add colliders to all of our Prefabs and make it so launching a projectile into an animal will destroy it. Finally, we will display a "Game Over" message if any animals make it past the player.</p> <p><b>Project outcome</b></p> <p>The animals will spawn on a timed interval and walk down the screen, triggering a "Game Over" message if they make it past the player. If the player hits them with a projectile to feed them, they will be destroyed.</p> <p><b>Skills</b></p> <p>Basic code comprehension</p> <ul style="list-style-type: none"> <li>• Interpret simple code</li> <li>• Improve simple code using the features of an IDE</li> </ul> <p>Basic application scripting</p> <ul style="list-style-type: none"> <li>• Write code that integrates into an existing system</li> <li>• Implement a code style that is efficient and easy to read</li> </ul>		
<p><b>Steps</b></p> <ol style="list-style-type: none"> <li>1. Make a new method to spawn animals</li> <li>2. Spawn the animals at timed intervals</li> <li>3. Add collider and trigger components</li> <li>4. Destroy objects on collision</li> <li>5. Trigger a "Game Over" message</li> <li>6. Lesson recap</li> </ol>		
<p><b>New functionality</b></p> <ul style="list-style-type: none"> <li>• Animals spawn on a timed interval and walk down the screen</li> <li>• When animals get past the player, it triggers a "Game Over" message</li> <li>• If a projectile collides with an animal, both objects are removed</li> </ul> <p><b>New concepts &amp; skills</b></p> <ul style="list-style-type: none"> <li>• Create custom methods/functions</li> <li>• InvokeRepeating() to repeat code</li> <li>• Colliders and Triggers</li> <li>• Override functions</li> <li>• Log Debug messages to console</li> </ul>		

## Challenge 2 - Play fetch

<b>Lesson link</b>	<a href="#">Challenge 2 - Play fetch</a>
<b>Length</b>	<b>1 hour</b>

## Challenge summary

Use your array and random number generation skills to program this challenge where balls are randomly falling from the sky and you have to send your dog out to catch them before they hit the ground. To complete this challenge, you will have to make sure your variables are assigned properly, your if-statements are programmed correctly, your collisions are being detected perfectly, and that objects are being generated randomly.

## Challenge outcome

- A random ball (of 3) is generated at a random x position above the screen
- When the user presses spacebar, a dog is spawned and runs to catch the ball
- If the dog collides with the ball, the ball is destroyed
- If the ball hits the ground, a "Game Over" debug message is displayed
- The dogs and balls are removed from the scene when they leave the screen

## Skills

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.
- Basic Debugging
- Diagnose and fix code that compiles, but fails to perform as expected

## Materials

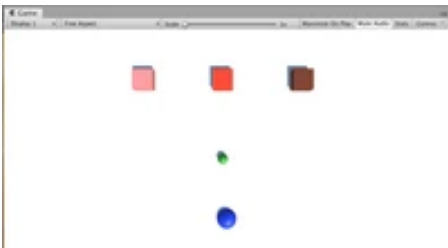
[Challenge 2 - Starter files](#) (.zip)



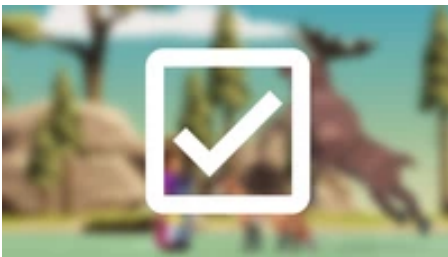
## Steps

1. Overview
2. Warning
3. Dogs are spawning at the top of the screen
4. The player is spawning green balls instead of dogs
5. The balls are destroyed if anywhere near the dog
6. Nothing is being destroyed off screen
7. Only one type of ball is being spawned
8. Bonus: The spawn interval is always the same
9. Bonus: The player can "spam" the spacebar key

## Lab 2 - New project with primitives

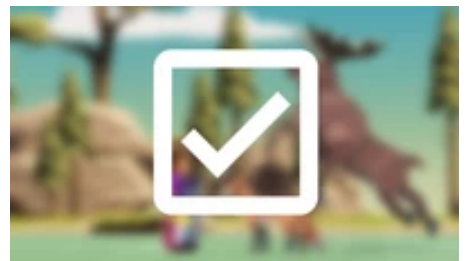
Lesson link	<a href="#">Lab 2 - New project with primitives</a>	
Length	1 hour	
<b>Challenge summary</b> You will create and set up the project that will soon transform into your very own Personal Project. For now, you will use “primitive” shapes (such as spheres, cubes, and planes) as placeholders for your objects so that you can add functionality as efficiently as possible without getting bogged down by graphics. To make it clear which object is which, you will also give each object a unique colored material.		
<b>Project outcome</b> <ul style="list-style-type: none"><li>All key objects are in the scene as primitive objects with the camera positioned properly for your project type.</li></ul>		
<b>Skills</b> Basic application scripting <ul style="list-style-type: none"><li>Prototype new concepts</li></ul>		
<b>Steps</b> <ol style="list-style-type: none"><li>Create a new project and rename your scene</li><li>Create a background plane</li><li>Create primitive Player with a new material</li><li>Position camera based on project type</li><li>Enemies, obstacles, projectiles &amp; materials</li><li>Export a Unity Package backup file</li><li>Lesson recap</li></ol>		

## Quiz 2

Lesson link	<a href="#">Quiz 2</a>	
Length	15 minutes	
<b>Quiz overview</b> This quiz will assess your knowledge of the skills and concepts learned in Unit 2.		
<b>Quiz objective</b> This quiz will assess your ability to: <ul style="list-style-type: none"><li>• Create an if-then statement in order to implement basic logic in your project, including the use of greater than (&gt;) and less than (&lt;) operators</li><li>• Transform a GameObject into a Prefab that can be instantiated into the scene</li></ul>		

- Work with Prefabs to add efficiencies in your workflows
- Get user input with GetKey and KeyCode to test for specific keyboard presses
- Use arrays to create an accessible list of objects or values and randomly select an object from that array
- Randomly generate values to randomize spawn positions
- Change the camera's perspective to better suit your game
- Repeat functions on a timer with InvokeRepeating
- Write custom methods to make your code more readable
- Detect collisions and destroy objects that collide with each other
- Display messages in the console with Debug Log

## Bonus features 2 - Share your work

<b>Lesson link</b>	<a href="#">Bonus features 2 - Share your work</a>	
<b>Length</b>	<b>1 hour</b>	
<p><b>Summary</b></p> <p>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.</p> <p>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!</p> <p>This tutorial is entirely optional, but highly recommended for anyone wishing to take their skills to a new level.</p>		
<p><b>Steps</b></p> <ol style="list-style-type: none"> <li>1. Overview</li> <li>2. Easy: Vertical player movement</li> <li>3. Medium: Aggressive animals</li> <li>4. Hard: Game user interface</li> <li>5. Expert: Animal hunger bar</li> <li>6. Solution walkthrough</li> <li>7. Share your work</li> </ol>		

## Mission 1 checkpoint

<b>Quiz:</b> <a href="#">Manage scene flow and data</a>
<b>Submission task:</b> <a href="#">Data persistence in a new repo</a>
<b>A successful submission will include:</b>

- A link to your project's GitHub repo, showing multiple commits with commit messages
- Some kind of data persistence between scenes
- Some kind of data persistence between sessions

**Steps**

8. Overview
9. Easy: Vertical player movement
10. Medium: Aggressive animals
11. Hard: Game user interface
12. Expert: Animal hunger bar
13. Solution walkthrough
14. Share your work