

Junior Programmer Pathway



Syllabus

What is the Junior Programmer Pathway?

The Junior Programmer Pathway is 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

A 12 to 14-week learning journey that teaches programming in Unity, and is designed for anyone who wants to become familiar with the process of creating C# scripts.

The Junior Programmer Pathway covers all the basic concepts and skills to introduce you to C# in Unity, and get you started on the path to becoming a Unity developer.

Objectives

C# skills:	Students will gain a foundational knowledge of programming in C# and will feel confident that they can implement new features on their own with this knowledge.
Unity skills:	Students will have the confidence that, given enough time and resources, they could create anything they want in Unity.
Project management:	As students create their own personal projects, they will learn to manage the process from start to finish: outlining their concept, setting project milestones, and tracking progress.
Unity Certified User exam:	Should they choose, students will have the skills and confidence to pass the Unity Certified User Programming exam, earning an official certificate validating their skills

Teaching approaches and contexts

Who are your learners?

The Unity Junior Programmer Pathway is a comprehensive entry point for getting started with C# development in Unity, specifically designed for those with no prior experience. Depending on the profile and prior experience of your learners, you can use it to facilitate a range of different experiences to best meet their needs.

Learner age range	Delivery suggestion
Lower secondary (middle school and junior high)	<ul style="list-style-type: none">→ Structured, facilitated sessions throughout, that break down the self-paced technical instructions into sessions with extension opportunities to ensure the group keeps pace→ Scaffolding and extension options mapped to those sessions will help provide differentiated learning experiences→ The software installation/new user onboarding guidance is unlikely to be required for this age range
Upper secondary (high school)	<ul style="list-style-type: none">→ Independent completion of the self-paced technical learning content, with scaffolding and extension options to provide differentiated learning experiences→ Facilitated research and discussion sessions on creator skills and real-time industry exploration→ The software installation/new user onboarding guidance is unlikely to be required for this age range

Adapting Junior Programmer content for different teaching approaches and contexts

Instructor/facilitator guidance

As an instructor/facilitator for a learning experience based around Junior Programmer, your most valuable contributions are likely to be:

- Basic scripting techniques and paradigms of C# development in Unity (this is especially the case for less technically literate cohorts)
- Facilitating discussion and exploration of creator skills and workplace industries
- Questioning to consolidate and deepen understanding
- Troubleshooting participant technical issues

The following table offers some guidance on adapting this learning experience for your teaching approaches and circumstances:

Flipped classroom / instruction	Pre-class work can be assigned by tutorial or mission within the Junior Programmer pathway. Research tasks for developer skills and real-time industries group discussions, presentations, or peer review feedback sessions are also ideal for the flipped classroom.
Project-based	The Junior Programmer Pathway is broken into Missions, with each Mission containing smaller projects. This can be used for project-based learning.
Inquiry-based	The Junior Programmer Pathway covers basic software development fundamentals, and so has not been designed with inquiry-based learning as a priority. However, the career and real-time industry information within the Pathway could provide the foundation for identifying research questions for further inquiry-based/research-based learning that meets the particular needs of your group.
Careers and industry focus	There are no dependencies between the real-time industry content in The Junior Programmer Pathway and the technical tutorials. These can be adapted as best meets the needs of your class or integrated into a wider career-focused learning experience.

Self-paced learning time estimates

To assist you in planning the learning time, we provide this table to illustrate the estimated time needed for each section and unit.

Mission 1: Junior Programmer: Create with code 1

		Time (suggested)	
Project	Tutorial	Minimum	Maximum
Getting started	Course introduction	10 min	20 min

	Install Unity software	25 min	35 min
<u>Unit 1 - Player control</u>	Start your 3D engines	80 min	120 min
	Pedal to the metal	70 min	100 min
	High speed chase	50 min	80 min
	Step into the driver's seat	50 min	80 min
	Challenge 1 - Plane programming	30 min	60 min
	Lab 1 - Project Design Document	90 min	120 min
	Quiz 1	10 min	15 min
<u>Introduction to project management and teamwork</u>	Introduction to project management and teamwork	20 min	30 min
<u>Unit 2 - Basic gameplay</u>	Unit 2 - Introduction	5 min	10 min
	Lesson 2.1 - Player positioning	60 min	90 min
	Lesson 2.2 - Food fight	60 min	80 min
	Lesson 2.3 - Random animal stampede	60 min	80 min
	Lesson 2.4 - Collision decisions	50 min	80 min
	Challenge 2 - Play fetch	90 min	120 min
	Lab 2 - New project with primitives	60 min	90 min
	Quiz 2	15 min	20 min

Mission 2: Create with code 2			
		Time (suggested)	
Project	Tutorial	Minimum	Maximum
<u>Unit 3 - Sound and effects</u>	Introduction	15 min	15 min
	Jump force	90 min	120 min
	Make the world whiz by	70 min	100 min
	Don't just stand there	10 min	10 min
	Particles and sound effects	60 min	90 min
	Challenge - Balloons, bombs, & booleans	60 min	90 min

	Lab 3 - Player control	60 min	90 min
	Quiz 3	15 mins	20mins
<u>Unit 4 - Gameplay mechanics</u>	Unit 4 - Introduction	5 mins	10 mins
	Lesson 4.1 - Watch where you're going	60 mins	90 mins
	Lesson 4.2 - Follow the player	60 mins	90 mins
	Lesson 4.3 - PowerUp and CountDown	60 mins	90 mins
	Lesson 4.4 - For-loops for waves	50 mins	80 mins
	Challenge 4 - Soccer scripting	60 mins	90 mins
	Lab 4 - Basic gameplay	90 mins	120 mins
	Quiz 4	15 mins	20 mins
<u>Unit 5 - User Interface</u>	Unit 5 - Introduction	5 mins	10 mins
	Lesson 5.1 - Clicky mouse	60 mins	90 mins
	Lesson 5.2 - Keeping score	70 Mins	100 Mins
	Lesson 5.3 - Game over	70 Mins	100 Mins
	Lesson 5.4 - What's the difficulty?	60 mins	90 mins
	Challenge 5 - Whack-a-food	60 mins	90 mins
	Lab 5 - Swap out your assets	90 Mins	120 Mins
	Quiz 5	15 Mins	25 Mins
	Introduction to user feedback and testing	20 mins	30 mins
<u>Next steps</u>	Next Steps - Introduction	10 min	15 min
	Lesson 6.1 - Project optimization	30 mins	45 mins
	Lesson 6.2 - Research and troubleshooting	60 mins	90 mins
	Lesson 6.3 - Sharing your projects	30 mins	45 mins
	ECS Survival Guide	10 min	15 min
	<u>Career research and preparation</u>	25 min	35 min
	<u>Introduction to portfolios</u>	25 min	35 min

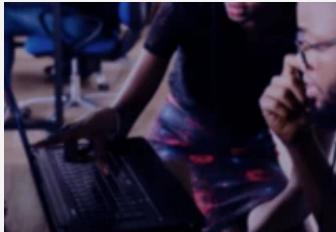
Mission 3: Junior Programmer: Manage scene flow and data

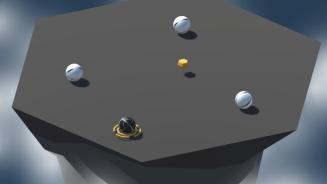
		Time (suggested)	
Project	Tutorial	Minimum	Maximum
	Introduction to real-time 3D experience design	20 mins	30 mins
	Set up version control	15 mins	20 mins
	Explore the sample project	15 mins	20 mins
	Principles of object-oriented programming	10 mins	15 mins
	Create a scene flow	20 mins	30 mins
	Implement data persistence between scenes	20 mins	30 mins
	Implement data persistence between sessions	20 mins	30 mins

Mission 4: Junior Programmer: Apply object-oriented principles

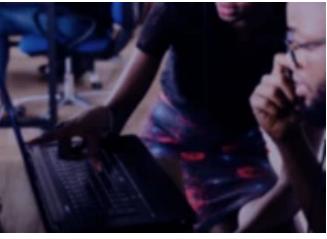
		Time (suggested)	
Project	Tutorial	Minimum	Maximum
	Abstraction in object-oriented programming	20 mins	30 mins
	Inheritance and polymorphism in object-oriented programming	20 mins	30 mins
	Encapsulation in object-oriented programming	20 mins	30 mins
	Profile code to identify issues	30 mins	45 mins
	Job preparation: Junior Programmer	15 mins	20 mins

Course Structure

Modules	Assessments	Personal Project
Mission 1: Create with code 1		
Getting started 	<i>In this introductory Unit, you will be introduced to the course, then you will download and install the Unity software.</i>	
Player control 	<i>Learn basic player control as you program a car that can steer down a floating road, avoiding (or hitting) obstacles in the way</i>	→ Challenge 1 → Quiz 1 → Bonus features → Lab 1
Introduction to project management and teamwork 	<i>Learn basic player control as you program a car that can steer down a floating road, avoiding (or hitting) obstacles in the way</i>	
Basic gameplay 	<i>Learn to implement basic gameplay with this top-down game where you throw food at animals, who are charging towards you.</i>	→ Challenge 2 → Quiz 2 → Bonus features → Lab 2
Sound and effects 	<i>Learn to add sound, animation, and effects with this side-scrolling game where the player needs to time their jumps over oncoming obstacles.</i>	→ Challenge 3 → Quiz 3 → Bonus features → Lab 3

Gameplay mechanics 	<i>Learn to program gameplay mechanics in a game where the player tries to knock off waves of enemies, using power-ups to help defeat them.</i>	→ Challenge 4 → Quiz 4 → Bonus features	→ Lab 4

Mission 2: Create with code 2

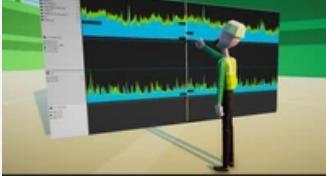
User Interface 	<i>Learn to implement a user interface in a game where the player needs to click on objects tossed in the air before they fall off the screen.</i>	→ Challenge 5 → Quiz 5 → Bonus features	→ Lab 5
Introduction to user feedback and testing 	<i>In this tutorial, you'll explore the purpose of user feedback and how user testing can be integrated into your design and development process.</i>	→	
Next steps 	<i>In this final Unit, you will prepare to continue developing your Unity and C# skills independently.</i>		
Career research and preparation 	<i>In this tutorial, you'll research career areas, job titles, and the necessary certifications and qualifications to help you identify career goals.</i>		
Introduction to portfolios	<i>In this tutorial, you'll:</i> <ul style="list-style-type: none">• <i>Review the goals and uses of a portfolio.</i>		

	<ul style="list-style-type: none"> • Plan your own portfolio using a flowchart. • Begin to select and organize content. 		
---	---	--	--

Mission 3: Manage scene flow and data

Introduction to real-time 3D experience design 	<p><i>In this tutorial, you'll explore the basics of real-time 3D experience design.</i></p>	
Set up version control 	<p><i>In this tutorial, you'll learn about the basics of Version Control, and the reasons to implement it in your own projects, even if you're developing applications by yourself</i></p>	
Explore the sample project 	<p><i>In this tutorial, you'll take some time to review the project brief and explore the Unity project for the application you're going to work on. You'll also consider all the resources that are available to support before you begin.</i></p>	
Principles of object-oriented programming 	<p><i>In this tutorial you'll learn about the basics of object-oriented programming paradigm and its four associated principles.</i></p>	
Create a scene flow	<p><i>In this tutorial, you'll set up the scene flow between the Menu and Main scenes, and the exit flow for the application in the application.</i></p>	

		
Implement data persistence between scenes 	<i>In this tutorial, you'll learn how to use data persistence to preserve information across different scenes by taking a color that the user selects in the Menu scene and applying it to the transporter units in the Main scene.</i>	
Implement data persistence between sessions 	<i>In this tutorial, you'll write code to save and load the color that the user selects so that it persists between sessions of the application.</i>	
Implement data persistence between sessions 	<i>In this tutorial, you'll write code to save and load the color that the user selects so that it persists between sessions of the application.</i>	
Mission 4: Apply object-oriented principles		
Abstraction in object-oriented programming 	<i>In this tutorial, you'll learn about the first pillar of object-oriented programming: Abstraction.</i>	
Inheritance and polymorphism in object-oriented programming	<i>In this tutorial, you'll learn about inheritance and polymorphism, two closely related pillars of OOP.</i>	

		
Encapsulation in object-oriented programming 	<i>In this tutorial, you'll learn about the second pillar in object-oriented programming: encapsulation.</i>	
Profile code to identify issues 	<i>In this tutorial, you'll learn how to use the Profiler to analyze a scene and identify where optimization bottlenecks are occurring.</i>	
Job preparation: Junior Programmer 	<i>In this tutorial, you'll review guidance on:</i> <ul style="list-style-type: none"> • Evidencing your progress • Updating your portfolio and resume • Preparing for interviews 	

Grading and rubrics

Overview

- 40%: Prototypes | 5 × 8% each
- 15%: Challenges | 5 × 3% each
- 10%: Quizzes | 5 × 2% each
- 35%: Personal Project | 1 × 35% each

* Note that these weight values are only suggestions

Prototypes

Weight	40% (5 × 8% each)
Description	Students follow along step-by-step over the course of 4 lessons to create

	a prototype with the same functionality as the instructor, but with a few of their own creative choices.
Purpose	To teach students all of the concepts and skills they'll need to complete the challenges and quizzes, and to provide examples of core components that they could add to their personal projects.

4 - Excellent	3 - Good	2 - Fair	1 - Unsatisfactory
<ul style="list-style-type: none"> - Project runs without error - All functionality present and operating as expected - Code and hierarchy are neat and commented, using correct conventions 	<ul style="list-style-type: none"> - Project runs without error - All functionality present and operating mostly as expected - Code and hierarchy are mostly neat and commented, using correct conventions 	<ul style="list-style-type: none"> - Project runs with some issues - Some functionality missing, and overall not operating as expected - Code and hierarchy are disorganized, using inconsistent conventions 	<ul style="list-style-type: none"> - Project barely runs - Most functionality absent - Code and hierarchy are messy, with no sign of consistency in conventions

Challenges

Weight	15% (5 × 3% each)
Description	Students are provided with an incomplete or broken version of a project and tasked with 5 items to implement or resolve, including a couple of Bonus challenges. They are also provided with hints and an example of a completed challenge.
Purpose	To allow students to apply the skills they learned while creating the prototype to a new, but somewhat similar context, solidifying the concepts and extending their skills.

4 - Excellent	3 - Good	2 - Fair	1 - Unsatisfactory
-All 5 tasks have been completed fully	-4 out of 5 of the tasks have been completed	-3 out of 5 of the tasks have been completed	-2 or less of the tasks have been completed

Quizzes

Weight	10% (5 × 2% each)
Description	Students complete 10 multiple choice questions.
Purpose	To give students the opportunity to apply and check their knowledge in a decontextualized environment, which will also help prepare them for the Unity Certified User exam.

4 - Excellent	3 - Good	2 - Fair	1 - Unsatisfactory
- 9-10 out of 10 correct	- 7-8 out of 10 correct	- 5-6 out of 10 correct	- Less than 5 out of 10 correct

Personal Project

Weight	35% (1 × 35% each)
Description	Students conceptualize, plan, and complete their own personal project throughout the course, integrating features they learned during the prototypes and extending them beyond. Students will be evaluated on completeness and uniqueness of their project.
Purpose	To give students an opportunity to extend their skills to a project that is uniquely their own, further solidifying the skills they learned and giving them the confidence that they can create whatever they want with the power of Unity and C#.

4 - Excellent	3 - Good	2 - Fair	1 - Unsatisfactory
<ul style="list-style-type: none"> - Project contains all of the features outlined in their project plan - Stayed on track with their planned milestones - Used their Unity and/or C# skills in a novel and creative ways - Code and hierarchy are neat & commented, using correct conventions 	<ul style="list-style-type: none"> - Project contains most of the features outlined in their project plan - Stayed mostly on track with their planned milestones - Used their Unity and/or C# skills in new, but not necessarily creative ways - Code and hierarchy are mostly neat & commented, using correct conventions 	<ul style="list-style-type: none"> - Project contains a few of the features outlined in their project plan - Did not really stay on track with their planned milestones - Did not use their Unity or C# skills in any new ways - Code and hierarchy are disorganized, using inconsistent conventions 	<ul style="list-style-type: none"> - Project does not contain any of the features outlined in their project plan - Did not stick to their planned milestones at all - Did not demonstrate an ability to apply skills they learned in the course - Code and hierarchy are messy, with no sign of consistency in conventions