Unity

Virtual Reality

Curricular framework

A resource for educators and administrators to bring virtual reality design and development to the classroom

Table of Contents

Introduction

Why this framework?

Virtual reality (VR), as a key technology in the current technological revolution, is dramatically impacting individuals, organizations, and environments. The industry is growing rapidly, and real-time 3D skills are in high demand. Beyond entertainment and video games, virtual reality is being applied in industries such as healthcare, automotive, manufacturing, architecture, construction and others. It is used for design visualization, virtual training, remote operation of equipment, and immersive marketing experiences.

The immersive nature of VR also makes it a suitable medium for empathy and behavior training. Demonstrating the effect of damaging behaviors in VR can be powerful and may lead to more positive behaviors for users, e.g., demonstrating the long-term effects of smoking on an avatar, or demonstrating the human impact over time on the planet.

3D engines like Unity have made it possible to create and deliver highly polished VR content. Such solutions make creating virtual reality experiences a much more accessible pursuit and with display methods becoming more affordable there is an opportunity to master VR content creation to engage a growing audience.

This curricular framework aims to enable teachers to quickly establish or expand innovative programs involving virtual reality. To do this, Unity has worked with industry experts and leading educators to develop clear, definable skills and learning objectives aligned with Unity certifications that address the skills and knowledge needed to use Unity in a professional setting and start a career in real-time 3D development.

The framework has three guiding principles:

- Professional targeting: The framework covers both technical and soft skills, including receiving critique, code and asset review, and portfolio development, all of which is crucial for budding professionals going into the field of game design or 3D asset development.
- Certification alignment: The framework's modules are marked to indicate where they align with exam objectives for all of Unity's Associate and Professional certifications.

- Learn-based resources: The skills outlined in the framework modules are scaffolded with Unity Learn resources that can be used to support instructor and student learning.

The curricular framework provides links to free learning resources from Unity Learn, the Unity Manual and suggested readings to meet the learning objectives and support all pedagogical approaches (synchronous, asynchronous, blended, in-person, or distance learning). These resources are updated as the virtual reality landscape and platform development tools change, so we recommend that you check back periodically to ensure you have the latest version.

Should you be using virtual reality at all?

Experiencing virtual reality for the first time is powerful. It's natural to start imagining the potential of the technology and the multitude of ways it might be applied. The medium, however, isn't suited for all experiences and has limitations, such as the ability to induce cyber sickness and cause discomfort when wearing VR hardware for extended periods of time. It's important to first ask if VR is the right medium for what you hope to achieve.

In his book Experience on Demand, Jeremy Bailenson (founding director of Stanford University's Virtual Human Interaction Lab) gives some useful guidance on experiences that work best in VR. The following acronym helps to summarize them: RIDE (Rare, Impossible, Dangerous, Expensive).

Rare

Experiences that are rare or very hard to reproduce are a great contender for VR. You could spend hours and a fortune to see a rare event in nature, or just experience it once in VR.

Impossible

Things that can't be achieved in real life make for compelling VR experiences, e.g., flying, time traveling, shrinking to the size of an atom, growing extra limbs, or becoming an animal. Things that aren't necessarily impossible but may be too expensive or exclusive for most people also make engaging VR experiences, e.g., climbing Mount Everest, traveling to the moon, or operating on a patient.

Dangerous

Virtual reality was developed for flight simulation in the 1920s and is used across industries today to transform the way people train for dangerous work, like firefighting, policing, and surgery.

Expensive

There are some experiences that will just forever be out of reach for the common person due to the excessive cost involved. VR gives the average person the ability to experience anything up close and personal in an immersive way. Some training simulations may also require expensive setup and management to reproduce, with high overheads due to trainee mistakes. VR offers a way to safely reproduce most situations in a cost-effective way.

Resources

Suggested resources throughout this framework support the mastery of skills outlined in each module. These include free learning resources from Unity Learn, our official online learning platform so that you can continue your learning and help students meet their objectives. We also highlight material from the Unity Manual, as well as other suggested readings. When using the Unity User Manual, ensure that it reflects the Unity version you are using by selecting the correct version from the dropdown menu in the upper-left corner of the page.

Common Terms

Unity maintains a glossary of terms that are common to the real-time 3D industries, as well as those specific to the Unity Editor and services.

Certification

The learning objectives in this framework have been aligned to Unity Certified Associate Certification exam objectives for educators aiming to prepare students to be certified. To ensure that students can be adequately prepared for the exams, it is recommended that educators take the exam themselves to gain a firm understanding of the exam's content and format.

How to use this document

Unity is used in various contexts across schools with learners of diverse backgrounds and prerequisite knowledge. For this reason, building a teaching guide that caters to the needs of all users is impractical. This document provides an in-depth overview of the skills involved in creating real-time 3D experiences to support the planning of your unique learning path.

Each module presents a table of skills , followed by a table of suggested learning objectives (see sample below) . This table displays the relevant learning objective , the available resources for the objective , and any certification exam objectives covered by the content .

Skill and Description	Learning Objectives	Resources	Related Certifications
Analyze the impact of art assets and lighting on performance (polycount, particles, visual effects, lighting, and shadows) Analyzing the impact on performance of factors such as poly count, particles, visual effects, lighting, and shadows involves assessing how these elements affect the frame rate and overall performance of a Unity project. **Assessment Suggestion:** Provide the learner with a Unity scene featuring various performance-intensive elements, and have them systematically identify, measure, and address performance issues related to factors like poly count, particles, visual effects, lighting, and shadows to optimize the scene's performance while maintaining an acceptable visual quality.	● Recognize the effects that Rigidbody and Collider components have on performance ● Set up the Unity Profiler to identify elements that cause performance impact ● Apply Unity's Stats window in order to investigate performance issues caused by assets	**Unity Learn** ● Optimization	

Sample of module

Learning objectives

Each module includes suggested learning objectives. We have identified these objectives based on typical knowledge or skills that are related to the specific module. While it is not critical to cover every learning objective in a module, the objectives are designed to complement each other in helping you fulfill the module aims.

Resources

Suggested resources throughout this framework support the mastery of skills outlined in each module. These include free learning resources from Unity Learn, our official online learning platform, so that you can continue your learning and help students meet their objectives. We also highlight material from the Unity User's Manual, as well as other suggested readings. When using the Unity User Manual, ensure that it reflects the Unity version you are using by selecting the correct version from the drop down menu in the upper left hand corner of the page.

Related Certifications

The learning objectives in this framework have been aligned to Unity Certified Associate Certification exam objectives for educators aiming to prepare students to be certified. To ensure that students can be adequately prepared for the exams, it is recommended that educators take the exam themselves to gain a firm understanding of the exam's content and format.

Preparing students for certification

One of the goals of this document is to help teachers develop programs that will lead students towards achieving a Unity Certification. Certifications test the core skills needed for a variety of roles so that students can validate their expertise and showcase their readiness for a role using Unity. The framework aligns with several Unity Certifications including the Associate: Game Developer, Associate: Programmer, and Associate Artist certifications. These are appropriate credentials for students looking to make the transition into professional work. Holding an Associate Certification indicates that a student has a mature understanding of Unity and is ready to begin in a junior or associate position on a professional team.

Unity Certified Associate **Game Developer**	Unity Certified Associate **Programmer**	Unity Certified Associate **Artist**
This certification is designed for future game developers who want to showcase their mastery of core Unity skills and concepts to obtain their first professional Unity role. Successful exam takers have a background in computer science or have a solid grasp of the skills required to become a Unity game developer. **Covers** ● Scripting, building, debugging, and optimization ● Prototyping art assets, whiteboxing levels	This certification is designed for students interested in a first professional role as a Unity developer, software engineer, software developer, mobile application developer, or gameplay programmer. Successful exam takers have a background in computer science. **Covers** ● Advanced Unity tools ● Scripting in C# for Unity ● Scripting UI elements	This certification is designed for students interested in a first professional role as a 3D artist, 3D generalist, game artist, level designer, environment artist, or 3D visualization artist. Successful exam takers have a mix of artistic and technical skills. **Covers** ● 3D asset creation, terrain generation, Scene building ● Basic understanding of C# in Unity

Additional teaching and learning resources

As well as providing the tutorials and projects that support the learning objectives throughout this framework, Unity Learn offers guided learning pathways that may be integrated into, or used in addition to, the materials in your program. These longer, self-paced experiences are designed to help anyone interested in coding and breaking into the gaming and tech industries expand their professional opportunities by gaining the skills they need to obtain a job, regardless of prior experience.

Unity Essentials pathway
Designed for anyone new to Unity, this guided learning journey is a first step toward gaining the background, context, and skills needed to confidently create in the Unity Editor. Completing this Pathway will equip students with the foundation needed to further their learning and specialize in their area of interest.

Junior Programmer pathway
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. By the end of the Junior Programmer pathway, students will be equipped to take the Unity Certified Associate: Programmer exam.

Creative Core pathway
Creative Core is your next step toward becoming a Unity creator. This free learning path will teach you all the core elements you need to bring your imagination to life with Unity. Once you've completed Unity Essentials as an introduction to the fundamentals of the Unity Editor, take this pathway to learn Visual Effects (VFX), Lighting, Animation, Audio, UI, and other creative skills, no programming required.

VR Development pathway
Welcome to VR Development! This learning pathway is designed for anyone interested in learning to create experiences for VR. This pathway assumes a basic knowledge of Unity and basic knowledge of programming.

Mobile AR Development Pathway
Ready to create AR experiences? In this learning pathway, you'll develop AR apps compatible with iOS and Android devices!

For those interested in how Unity can be a tool for Metaverse related technologies and applications, a Live Learning series, called Road to the Metaverse is available on Unity Learn.

Unity eBooks
You can find these and many more resources here.

- Unity Game Dev Field Guide - This guide will help you jump-start your familiarity with the latest in Unity’s rich feature set and intuitive workflows
- Unity for Technical Artists - provides an overview of the toolsets and systems in Unity that Technical Artists can use
- The definitive guide to lighting in the High Definition Render Pipeline - learn how to harness the power of physically based lighting in the HDRP
- Top tips for improving your workflows and productivity with Unity 2020 LTS - a guide that collects over 70 time-saving tips to improve your day-to-day aggregate workflow with Unity
- UI design and implementation - a treasure trove of useful tips for advancing your UI development skills with the default Unity UI and the new UI Toolkit.
- Create modular game architecture in Unity with ScriptableObjects - This guide provides tips and tricks from professional developers for deploying ScriptableObjects in production.
- The definitive guide to creating advanced visual effects in Unity- This e-book provides a complete overview of how to use visual effects authoring tools in Unity to create any kind of effect.
- User interface design and implementation in Unity - Written by experienced Unity creators and UI professionals, the e-book provides step-by-step guidance on how to make UIs that look great across a wide range of devices.
- Best Practices From Successful Mobile Indies - Learn best practices for mobile success with tips from indie experts.
- Introduction to the Universal Render Pipeline for advanced Unity creators - This e-book was created by a highly experienced Unity developer in collaboration with senior graphics engineers at Unity.
- Build Industrial Digital Twins| Free Guide for Robotics & Automation | Unity - provides a practical, hands-on guide for robotics engineers, automation professionals, and OEMs ready to accelerate their digital transformation with Unity Industry.

Unity hardware requirements

You can find the latest Unity hardware requirements in the Unity documentation. Go to Unity – Manual and then select **Working in Unity** > **Installing Unity** > **System requirements for Unity [version]**.

Getting support from the community

The Unity creator community is a vibrant and engaged network of Unity enthusiasts who embody vast knowledge. Whether you're researching your own area of interest or guiding students to troubleshoot, we recommend starting with the following resources within the Unity ecosystem:

Unity Discussions
Beginners and experts alike post to this platform, so they can help each other out with Unity. The built-in voting system helps you find the best answers faster.

While we would love for you to find the answers to all of your questions here on the Unity Learn platform or within the wider Unity learning ecosystem, we know that our community is much broader. We encourage you to research and connect in the many spaces in which our creator community lives. Here are a few of the better-known resources in the Unity creator community:

YouTube
There are many channels and videos dedicated to learning Unity. Some popular channels include Game Dev Unlocked (created by established creator David Wehle), Brackeys, Code Monkey, and Dani, as well as our own official Unity channel.

Discord
Discuss Unity in real-time. Join the Discord server.

Stack Exchange and Stack Overflow
These open communities help creators in diverse fields get their questions answered with a reputation award process. Stack Overflow is dedicated to programming. On Stack Exchange, check out questions tagged "unity" in the gamedev exchange.

Reddit
A network of communities based on people's interests. Take a look at the Unity, Unity3D, and Unity2D communities just for starters.

X (previously Twitter)
Follow @unity3D and watch #unity and other hashtags to see what the Unity community is creating.

Creating in Unity without programming

Although programming is a helpful skill to have when developing projects with complex interactivity in Unity, it is not necessary to be a coder to create with Unity. For example:

Certain types of projects, such as 3D visualizations and animations, don't require code at all.

- Visual scripting allows developers to implement logic in their projects using intuitive drag-and-drop graphical connectors without any knowledge of code or IDEs.
- The Unity Asset Store provides pre-made scripts and tools for the development of common features, such as a first-person controller or an inventory system.

Using Google, combined with sites like Unity Discussions, Unity Discussions, and Stack Overflow, developers can copy, paste, and modify the coding solutions provided by other developers. (It is surprising how far you can get with a little Googling and a lot of perseverance!

Introduction to the Unity real-time platform

Module introduction

Unity is the world's leading platform for creating and operating interactive, real-time 3D content, providing the tools to make amazing experiences and publish them to a wide range of devices.

The cross-platform nature of the Unity 3D platform means you can build your content once, and then deploy across over 20 platforms, including Windows, Mac, iOS, Android,PlayStation, Xbox, Nintendo Switch, and the leading AR and VR platforms.

This module is intended as an introduction to the Unity Editor and how to use it. Students who will be doing practical projects in game design need to be familiar and comfortable with the Unity Editor. If students will be using their own devices, they would ideally be given time outside of class to complete the first few steps of onboarding in Unity.

We suggest giving the following free resources to students for preparatory self-study before classes commence: Editor Essentials module from the Unity Essentials pathway.

If you are interested in a more comprehensive deep dive into 2D development in the Unity Editor, our most comprehensive 2D game development guide is now available, as well as the new Sprite Flight and 2D adventure Game tutorials.

Unity also provides a growing range of services, most with free tiers of use, to help developers build, manage, and grow their business from their applications, as well as extend and integrate into 3rd party applications. Below is a list of notable services that may be useful in the industries this curricular framework caters to, but the full range can be viewed on the Unity Gaming Services reference and the Unity Cloud onboarding guide.

Unity Gaming Services

- Accounts
- Mulltiplayer
- Content Management
- Analytics
- Community Tools
- Monetization tools
- Game Crash Reporting Tools

Unity Cloud

- Unity Asset Manager
- Unity Version Control
- Unity Build Automation

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create and manage projects in the Unity Hub Unity uses the Unity Hub to install and manage the various	● Install a version of the Unity Editor using the Unity Hub	**Unity Learn** ● Install Unity ● Install a new Unity Editor ● Create a new project	

Unity versions and additional components. A Unity ID is required to access a lot of the functionality of the Unity Editor and will have all the licenses and assets from the Asset Store linked to it. With purposeful organization, learners can avoid being overwhelmed and create Unity projects that are easy to navigate. **Assessment suggestion** Evaluate students' Unity understanding by having them install Unity Hub, create a Unity ID, access the Asset Store, and organize a Unity project for efficient navigation. Additionally, test their ability to switch between different Unity versions within Unity Hub.	● Create a new Unity project using a template in the Unity Hub ● Open an existing Unity project from the Unity Hub ● Explain the differences between and purposes of LTS and TECH Stream releases ● Update a project to a newer version of the Unity Editor using the Unity Hub ● Explain the role of Unity Hub in creating and managing projects ● Explain the purposes and uses of the sections of the Unity Hub interface ● Explain the uses of the 3D, 2D, and Microgame templates in the Unity Hub ● Add a Unity project from another source to the Unity Hub ● Explain why version control is essential in real-time development among teams	● Add new modules to a Unity Editor ● Install a package via the Package Manager ● Project Organization **Unity Manual** ● Install the Unity Hub ● The Project window	

Create and manage Scenes Scenes in Unity are fundamental containers that hold and organize game objects, assets, and the environment for a specific part or level of a game. They are crucial for game development because they allow developers to structure and manage different parts of their game, enabling seamless transitions between gameplay elements, efficient asset loading, and streamlined testing and iteration, ultimately contributing to a more organized and manageable game development process. **Assessment Suggestion** Have students demonstrate their comprehension by instructing them to generate a fresh scene within their project, labelling it as "New Scene." Request that they showcase their ability to identify their current working scene and explain the process of switching to the newly created scene.	● Explain the role of scenes in a Unity project ● Create a new empty 3D Scene ● Create a new empty 2D Scene ● Open a scene in a Unity project	**Unity Learn** ● Create a new Scene ● Open a Scene **Unity Manual** ● Scenes ● Scene view navigation	● Associate Game Developer

Identify and use essential features of the Unity Editor The Unity Editor interface consists of various areas, each designed for specific tasks. Learners should familiarize themselves with these features before focusing on VR development. Since the Unity Editor is a professional tool, there's a lot to learn. The free Unity Essentials Pathway provides learners with a comprehensive guide to these essential features. **Assessment Suggestion** Have students demonstrate their comprehension by instructing them to open a new Unity project and identify the default layout of the Unity Editor. Ask them to take a screenshot and label the following windows: Scene view, Game view, Hierarchy, Project window, Inspector, and Console. Then, have them rearrange the layout by undocking the	● Identify and describe the windows that appear in the Unity Editor's default view ● Start and stop Play mode (Game view) ● Rearrange, dock, and undock windows in the Unity Editor ● Explain the differences between the Project and Hierarchy windows ● Explain the relationship between the Hierarchy window and the Scene view ● Explain when to use the Scene view and the Game view ● Explain the purpose and functionality of the Package Manager ● Use the Package Manager to add functionality to the Unity Editor ● Explain the relationship between the Assets folder in the Project window and the Asset folder in file explorer	**Unity Learn** ● Explore the Editor Interface **Unity Manual** ● The Project window ● The Editor interface ● Scene view navigation ● GameObjects ● Tags and Layers ● Unity's Asset Store	● Associate Game Developer

Inspector window and docking it on the opposite side of the Editor. Finally, ask students to explain the purpose of each window and how rearranging them might help their workflow.	• Organize assets using folders in the Project window		
Employ Unity Version Control in a project Unity Version Control is a version control solution built to help teams manage changes and collaborate effectively within the Unity Editor. Originally known as Plastic SCM, it was acquired by Unity and has since been integrated directly into the Unity development workflow, offering tools for branching, merging, and managing project history tailored to real-time content creation. **Assessment suggestion:** Have students explain how they collaborated with PlasticSCM and identify the successes and challenges of setting up collaboration.	• Identify changed files of publish and update operations • Describe additions during publish operations • Recognize when to perform a publish or update • Recognize when to revert unintended changes prior to publishing • Locate where a project resides in the Unity Development Dashboard • Restore previous commits by using the version history • Explain the primary purposes of version control when working in Unity	**Unity Learn** • Get started with Unity Version Control • Unity Version Control: Quick start guide • Collaborate with Unity Version Control **Unity Manual** • Unity Version Control (previously Plastic SCM)	• Associate Game Developer • Associate: Programmer

Evaluate Unity and real-time 3D in order to determine whether they are suited to your needs The term real-time is used frequently in creative industries but is rarely clearly explained. Understanding what this term means as well as the impact a real-time 3D platform like Unity has on the creative workflow will allow learners to determine potential use cases and identify the problems it solves for creators. **Assessment suggestion:** Have students articulate how they may be able to implement real-time functionality in their projects to assist or enhance the desired outcome.	● Define the term real-time ● Explain what the Unity real-time engine does ● Describe how real-time creation software is used in different industries ● Identify a variety of real-time creators by their usage of Unity or their job role ● Explain what a real-time game engine is and how it is used	**Unity Learn** ● Real-time creation	

Introduction to C# in Unity

Module introduction

While it’s certainly possible to create a VR experience in Unity without scripting, it will be severely restricted in functionality. C# scripting in Unity unlocks new functionality and allows you to create amazing VR experiences. In this module, you’ll learn about the goals of the Unity C# Scripting Fundamentals project, including scripting basics, controlling code flow, basic GameObject manipulation, and GameObject interactions.

As a primer, we suggest students complete the first two missions in the **Junior Programmer pathway** on their own time before the course begins.

Ideally, a Unity project should feel like it’s been developed by a single author, no matter how many developers actually work on it. A style guide can help unify your approach for creating a more cohesive codebase. In partnership with internal and external Unity experts, we released a new e-book, **Create a C# style guide** and **Level up your code with design patterns and SOLID**.

Write cleaner code that scales for inspiration, based on **Microsoft’s comprehensive C# style.**

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Analyze the principal pillars of object-oriented programming C# is an advanced scripting language with many features that enable complex functionality in Unity. Advanced skills and knowledge will give the student the freedom to create complex applications and achieve their required application goals. **Assessment suggestion:** Have students set up a version control system for their code.	● Define abstraction ● Define inheritance ● Define polymorphism ● Define encapsulation ● Explain how the pillars of OOP work together to create organized, efficient code	**Unity Learn** ● Apply object-oriented principles	
Create basic application interactions with Visual Scripting The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This is a useful approach if you are	● Create a player inventory using the List object type in a visual script ● Detect a button press or other user action in a visual script ● Play audio from a visual script ● Make a visual script that changes a GameObject's properties	**Unity Learn** ● Visual Scripting application **Unity Manual** ● Basic concepts of Visual Scripting ● Developing game flow using script graphs ● Developing logic transitions using state graphs	

not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'		● Developer's guide and references ● Basic concepts in Visual Scripting **Unity Resources** ● Visual scripting	
Apply events in visual scripts The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This is a useful approach if you are not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	● Add a new custom event trigger to a visual script ● Construct a visual script that responds to a custom event ● Pass any number of arguments from one script to another by way of a custom event	**Unity Learn** ● Visual Scripting application **Unity Manual** ● Basic concepts of Visual Scripting ● Developing game flow using script graphs ● Developing logic transitions using state graphs ● Developer's guide and references ● Basic concepts in Visual Scripting **Unity Resources** ● Unity Visual Scripting	

Apply variables in visual scripts The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This is a useful approach if you are not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	● Create Graph, Object, and Scene variables and explain their uses ● Add Get Variable nodes to a Graph using the Blackboard ● Make variables available to be changed in the Inspector window ● Troubleshoot adjusting variable values in Scene and Game views ● Explain the Scene Variables object that appears in the Hierarchy of projects with Visual Scripts	**Unity Learn** ● Visual Scripting application **Unity Manual** ● Basic concepts of Visual Scripting ● Developing game flow using script graphs ● Developing logic transitions using state graphs ● Developer's guide and references ● Basic concepts in Visual Scripting **Unity Resources** ● Visual scripting	
Construct a visual script with basic code flow and logic The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This	● Apply Boolean logic and conditional branching in visual scripts ● Use the switch statement in visual scripts ● Make mathematical calculations in visual scripts ● Detect keyboard input in a visual script	**Unity Learn** ● Visual Scripting application **Unity Manual** ● Basic concepts of Visual Scripting ● Developing game flow using script graphs	

is a useful approach if you are not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	• Use and interpret common object types in visual scripts • Identify essential programming structures in order to comprehend a visual script	• Developing logic transitions using state graphs • Developer's guide and references • Basic concepts in Visual Scripting **Unity Resources** • Visual scripting	
Control the execution of code with common logic structures As a rule, code will flow in a linear way. Operators and loops allow the user to stop and change the flow of code based on conditions. **Assessment suggestion:** Have students adjust the color script from above, but alter it to make the color loop through different values assigned to an array.	• Use if and if-else statements in code • Control the execution of code by using logical operators such as AND and OR in conditional statements • Control how many times certain lines of code run by using for loops, foreach loops, and while loops • Control the order and timing of executed code by using coroutines • Control the execution of code by using switch statements	**Unity Learn** • IF Statements • Loops • Switch Statements • Arrays • Enumerations • Implement data persistence between scenes • Implement data persistence between sessions	• Associate: Programmer • Associate Game Developer

	● Modify the values of numeric variables by using mathematical operators		
Create a GameObject component with a script Unity applications revolve around the GameObject. Accessing the GameObject via script at runtime is an essential skill for game coding and will give the student the ability to manipulate the GameObjects based on conditions and user input.	● Explain the relationship between scripts and components ● Make a new script component ● Open the IDE from the Unity Editor ● Explain the purpose of the default code generated within a newly created C# script ● Apply tags or layers to GameObjects in order to identify specific objects from within a script ● Add a script component to a GameObject ● Change a variable's accessibility in the Inspector by editing its access modifier to public or private ● Print debug messages to the console by calling the Debug.Log method	**Unity Learn** ● GetComponent ● Translate and rotate ● GetButton and GetKey ● Collision decisions ● Instantiate ● Destroy **Unity Manual** ● Instantiating Prefabs at runtime	● Associate: Artist ● Associate: Programmer ● Associate Game Developer

Diagnose and fix common compilation errors Very few people can write errorless code on the first try. Understanding how to debug your code will allow you to efficiently search for and fix errors in your scripts.	- Locate a bug in code that produces a compilation error - Recommend the fix for a compilation error - Recognize when a new namespace needs to be imported	**Unity Learn** - Introduction to the Console window	- Associate: Programmer
Employ a State Machine in a visual script The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This is a useful approach if you are not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	- Distinguish a State Graph from a Script Graph - Build a new State Graph - Build Script Graphs for the states in a State Machine - Navigate among the various scripts in a State Machine - Devise and configure transitions in a State Graph - Interpret an existing complex visual script - Adjust an existing Script Graph for use in a State Machine	**Unity Learn** - Visual Scripting application **Unity Manual** - Basic concepts of Visual Scripting - Developing game flow using script graphs - Developing logic transitions using state graphs - Developer's guide and references - Basic concepts in Visual Scripting **Unity Resources** - Visual scripting	

Interpret simple code within a code base C# scripts allow you to create and extend custom functionality and properties on a GameObject. A solid understanding of C# script anatomy will give you more freedom when creating new applications and enable you to create custom functionality. **Assessment suggestion:** Have students create a simple script and apply it to a GameObject. The script could be used to print the current material color on the object to the log.	● Identify the purpose of common methods found in MonoBehaviours such as Start() and Update() ● Define the major features of a script such as namespaces, classes, variables, and methods ● Identify essential programming structures in order to comprehend simple code ● Choose the appropriate data types for a specific situation including but not limited to floats, bools, and strings ● Distinguish object-oriented code from data-oriented code ● Distinguish an ECS (Entity Component System) class from any other type of class, given a code block containing a class definition ● Explain the Vector2 data type ● Recognize naming conventions conforming	**Unity Learn** ● Get Started with Visual Studio and Unity ● Essentials of Programming in Unity ● Scripts as behavior components ● Beginner scripting **Unity Manual** ● Visual Studio C# integration ● Creating and using scripts ● Creating and Using Scripts	● Associate Game Developer

	to Unity standards, given a set of code blocks		
Manage visual scripts in a project The Visual Scripting module (formerly known as Bolt) is a node-based tool that allows you to create the same logic and interaction in your scene as standard C# scripting, without requiring knowledge of C#. This is a useful approach if you are not familiar with coding but still want to add additional functionality to your scenes. **Assessment suggestion:** Have students can work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	• Group nodes in a visual script • Add titles and comments to a visual script using groups • Create and edit a subgraph that you can call from other visual scripts • Specify the inputs and outputs to a subgraph in the Graph Inspector	**Unity Learn** • Visual Scripting application **Unity Manual** • Basic concepts of Visual Scripting • Developing game flow using script graphs • Developing logic transitions using state graphs • Developer's guide and references • Basic concepts in Visual Scripting **Unity Resources** • Visual scripting	
Program efficient, organized, and comprehensible scripts by correctly implementing the principles of object-oriented programming	• Organize classes so that each has a single purpose, in order to enable easier readability and debugging	**Unity Learn** • ECS survival guide • Principles of object-oriented programming	

The Visual Scripting module (formerly known as Bolt) is a node-based visual scripting module that allows the user to create the same logic and interaction in their scene as standard C# scripting without requiring knowledge of the C# language. This is a useful approach for users who are not familiar with coding but still want to add additional functionality to their scenes. Assessment suggestion: Have students work through and complete the visual scripting course on Unity Learn - Visual Scripting application: Clive the Cat's 'Visual Crypting'	● Add new functionality to non-editable classes by applying extension methods ● Organize and prevent conflicts between scripts by using namespaces ● Use events to relay a GameObject's status changes to other objects in the application	● Introduction to ScriptableObjects **Unity Manual** ● Unity Manual: ScriptableObject	
Simplify code and make it reusable by correctly implementing the principles of inheritance and polymorphism C# is an advanced scripting language with many features that enable complex functionality in Unity. Advanced skills and knowledge will give the student	● Explain how abstraction is used to expose only necessary script components ● Explain how inheritance is used to share functionality between a parent and child class ● Define the relationship between a parent and	**Unity Learn** ● Principles of object-oriented programming	● Associate: Programmer

the freedom to create complex applications and achieve their required application goals.	child class, including what a child class can and cannot do with respect to its parent class ● Recognize opportunities where inheritance could be used to simplify code ● Describe how polymorphism can be applied at compile time (method overloads) and run time (method overrides) ● Explain how polymorphism is used to modify parent class functionality in a child class ● Explain how encapsulation is used to write code that can only be used as intended by the programmer ● Recommend a high-level system architecture for a given project		
Use appropriate data types for a specific situation	● Initialize variables of a given data type, including ints, floats, doubles,	**Unity Learn** ● Variables and Functions **Unity Manual**	● Associate: Programmer ● Associate Game Developer

Variables allow the user to store data in the code. Understanding how this works and how to implement it will give the user the ability to process data and access GameObjects in the script. **Assessment suggestion:** Have students create a simple script and apply it to a GameObject. The scripts could be used to print the current material color assigned to the object to the debug log, and change the material to a new color as specified in a public variable.	bools, strings, arrays, lists, and dictionaries ● Select the correct data type for a variable in a given situation ● Select appropriate variable modifiers including public, private, static, protected, and const ● Choose the appropriate commonly used data structures for a specific situation including but not limited to lists, arrays, and dictionaries	● Variables and the Inspector	

Introduction to virtual reality

Module introduction

This module is theoretical and acts as a good primer for courses on virtual reality. Virtual reality has a surprisingly long and fascinating history, especially for a technology generally considered new. Knowing this history and the technology's current capabilities by experiencing various applications will better enable students to assess the suitability of their ideas and designs.

With virtual reality, users are completely immersed in a three-dimensional space, and traditional design practice does not always translate well into this medium. The digital embodiment of a person in this space also brings new challenges, like inclusive design practice and user comfort considerations. By studying and being conscious of these factors, students can create applications that engage a wide audience.

For those interested in how Unity can be a tool for Metaverse-related technologies and applications, we have an eBook, Create VR & mixed reality experiences in Unity | Unity, and a Live Learning series, called Road to the Metaverse is available on Unity Learn.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Configure VR projects for deployment to various head-mounted displays (HMDs) to ensure compatibility and optimal user experience Unity has built-in tools that allow you to publish to various platforms with the click of a button. Some of the platforms require you to tweak some settings, but generally only a few changes are required before publishing. Understanding the different platform settings will allow students to share their applications with a wide audience with different hardware and software setups. **Assessment suggestion:** Have students successfully publish a Unity project to two different platforms.	● Deploy applications to tethered and untethered VR headsets ● Set up a new VR-compatible project for development ● Select appropriate, recommended settings for publication to a particular platform ● Build projects on Unity-supported head-mounted displays	**Unity Learn** ● VR Software Setup ● Introduction to XR: VR, AR, and MR Foundations **Unity Manual** ● Create an XR project	

Design systems that meet user needs Real-time technology is often not designed to be inclusive of all users. Learners should be able to implement design practices that enable and draw on the full range of human diversity. This means recognizing biases and understanding where exclusion happens and designing with users at the center from the start of the process. **Assessment suggestion:** Have students assess existing applications for inclusivity, and make suggestions on how to improve them.	• Identify the key features, systems and attributes of a real-time experience • Evaluate the accessibility of the features and systems in a real-time experience • Investigate accessibility considerations related to the systems required for a game	**Other Resources** • Accessibility and inclusion forum • Convention on the Rights of Persons with Disabilities (CRPD) | United Nations Enable • Mismatch: How Inclusion Shapes Design • XR Access Symposium – Resources • Accessibility of Virtual Reality Environments | University of Melbourne • Introducing the Accessibility VRCs | Oculus • Hamlet on the Holodeck, by Janet Murray	
Review and provide constructive feedback on proposed VR experiences in order to enhance quality and user satisfaction Unlike traditional media and video games, virtual reality requires a dedicated effort to cater to user comfort. The	• Describe the advantages and disadvantages of VR experiences, when compared with traditional screen-based experiences • Describe the design considerations specific to VR experiences, when compared with traditional screen-based experiences	**Other Resources** • Overview of immersive VR apps best practices • The VR Book: Human-Centered Design for Virtual Reality, by Jason Jerald	

immersive nature of VR can cause nausea, vertigo, headaches, and eye strain when not well designed. Experienced and even causal users of VR are often oblivious to the impact it can have on a new user. Recognizing best practices for user experience and comfort will enable creators to engage a wide audience. **Assessment suggestion:** Have students assess existing applications for comfort, and make suggestions on how to improve it.	● Explain the importance of optimization and performance for VR experiences ● Explain comfort and accessibility considerations specific to VR experiences		

Interaction in Virtual Reality

Module introduction

Designing interactions and movement in virtual reality (VR) applications is a multifaceted endeavour that encompasses user experience, comfort, hardware constraints, and physical surroundings. This module will cover common VR interactions, including XR Interaction Toolkit (XRIT) utilization for 3D and UI interactions. XRIT enables cross-platform XR controller input, haptic feedback, and visual cues for object interactions and basic canvas UI interactivity. Additionally, it explores various VR locomotion techniques such as teleporting, constant movement, room-scale navigation, and stationary interactions.

The module also addresses audio design for real-time 3D applications, emphasizing the creation of spatialised 3D audio effects by applying audio experience design principles. This comprehensive approach ensures immersive, user-friendly VR experiences by combining interaction design, locomotion options, and high-quality audio to captivate and engage users effectively.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create realistic spatialized 3D audio effects by applying audio experience design principles Spatial audio provides a method through which to build and place audio assets so that – from the VR user's perspective – a given sound originates from a particular position in a 3D scene. This is like surround-sound in a home theatre setup or at the cinema, and very important to presence and immersion in VR. **Assessment suggestion:** Have students set up location-based audio in a scene to enhance the immersion of the user.	● Explain the difference between diegetic and nondiegetic sound ● Explain the role of audio in developing atmosphere ● Explain the role of audio in supporting narrative and worldbuilding	**Unity Learn** ● Beginning Audio in Unity **Unity Manual** ● VR Audio Spatializers	
Create common VR interactions in VR applications. One of the first things most users do when entering VR is look down to find their hands. This is an instinctive action that	● Implement locomotion in VR ● Implement grabbable objects and sockets in VR ● Create a worldspace UI in VR ● Implement controller- and object-based interaction events in VR	**Unity Learn** ● Grabbable Objects **Unity Manual** ● Unity XR Input **Other Resources**	

allows the mind to anchor itself in the VR space. Implementing interaction between the user and the VR environment enhances the sense of immersion. The XR Interaction Toolkit includes components that enable students to easily create these interactions. **Assessment suggestion:** Have students download, install and set up the relevant packages to create interaction with objects in a VR scene.	● Implement spatial audio in VR	● Haptic Feedback ● SteamVR Vibration	
Create a plan to design audio for a real-time 3D application The Unity real-time engine allows the user to insert soundtracks and location-based sound effects into their scene. This functionality allows the artist to create immersive scenes and environments in their project. **Assessment suggestion:** Have students implement simple audio into a scene that will change	● Describe the science of audio in digital environments ● Describe the primary types of audio found in real-time projects	**Unity Learn** ● Essentials of real-time audio **Unity Manual** ● Audio ● VR Audio Spatializers	● Associate: Game Developer

based on the distance of the user from the source.			
Employ VR locomotion techniques, such as teleporting, constant movement, room scale, and stationary Movement in VR is a consideration that warrants dedicated study. Learners need to decide if the user must move in their experience and, if so, how they should do it. Every solution will bring its own design considerations and challenges. It's important to know what locomotion techniques are possible and which will be most effective for your experience. **Assessment suggestion:** Have students download, install and set up the relevant packages to create locomotion in a VR scene.	● Implement locomotion in VR	**Unity Learn** ● VR Locomotion **Unity Manual** ● Locomotion **Other Resources** ● Building with Unity	

World space UI & touch interfaces

Module introduction

Creating effective world space UI and touch interfaces for virtual reality (VR) and augmented reality (AR) experiences is a task that demands careful consideration. Just as with movement and interaction in VR, designing UI elements and touch interfaces in the spatial context of the virtual world requires a thoughtful approach. This module will delve into the intricacies of developing world space UI and touch interfaces for VR and AR, exploring the techniques and tools available to Unity developers. From understanding the principles of world space UI design to implementing touch interactions seamlessly, this module aims to equip you with the knowledge and skills necessary to enhance user experiences in immersive environments.

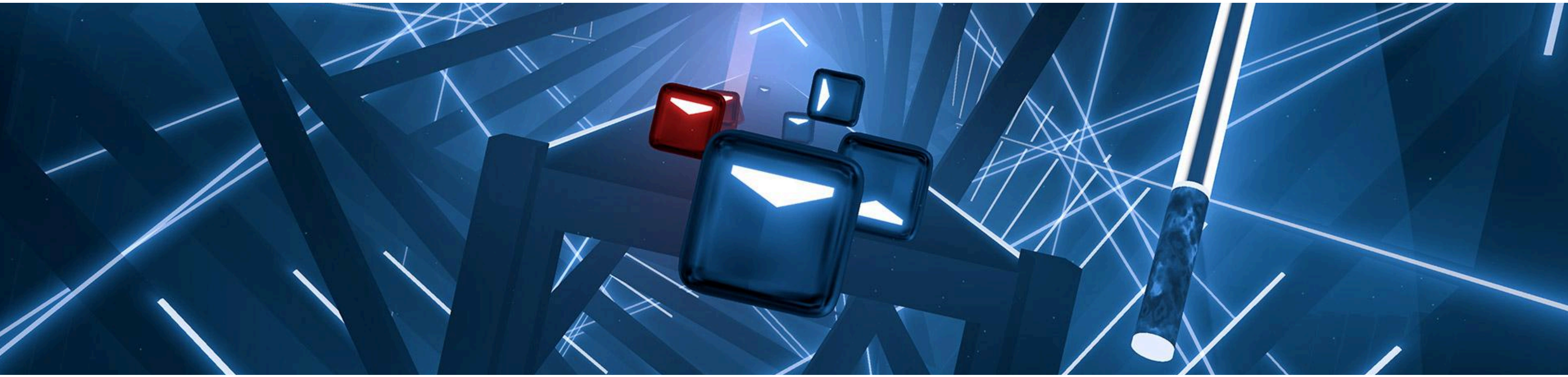
Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create common VR interactions in VR applications.	● Implement locomotion in VR	**Unity Learn** ● Grabbable Objects	

One of the first things most users do when entering VR is look down to find their hands. This is an instinctive action that allows the mind to anchor itself in the VR space. Implementing interaction between the user and the VR environment enhances the sense of immersion. The XR Interaction Toolkit includes components that enable students to easily create these interactions. **Assessment suggestion:** Have students download, install and set up the relevant packages to create interaction with objects in a VR scene.	● Implement grabbable objects and sockets in VR ● Create a worldspace UI in VR ● Implement controller- and object-based interaction events in VR ● Implement spatial audio in VR	**Unity Manual** ● Unity XR Input **Other Resources** ● Haptics in Unity ● SteamVR Vibration	
Create user interfaces as defined in design documents Unity offers a suite of advanced UI management tools to create complex UI interactions. **Assessment suggestion:** Have students design and implement a complex menu flow in the application state.	● Configure UI components to be used with scripts ● Arrange UI components on the canvas according to a defined layout using anchors, pivots, and groups ● Organize UI components using optimization best	**Unity Learn** ● Creating basic UI with uGUI ● VR User Interface **Unity Manual** ● Data binding ● Create user interfaces (UI)	● Associate: Programmer

	practices such as using nested canvases		
Decide on a user interface approach for a project Unity offers a suite of advanced UI management tools to create complex UI interactions. **Assessment suggestion:** Have students design and implement a complex menu flow in the application state.	- Define the acronym UI - Describe the role of user interfaces in real-time 3D experiences - Describe the importance of consistency and clarity in effective UI approaches - Differentiate between the following terms: User Interface Design (UI), User Experience Design (UX), User Interaction Design (IxD), Information Architecture (IA), and Visual Design - Distinguish between Unity's three available UI systems: uGUI (or Unity UI), IMGUI (or “Immediate Mode” GUI), and UI Toolkit - Recall essential accessibility considerations for UI, such as font choice, text size, color contrast, and content	**Unity Learn** - UI **Unity Manual** - Unity UI - UI Toolkit **Unity Resources** - User interface design and implementation in Unity	- Associate: Artist

Program scripts for interactive user interfaces Unity offers a suite of advanced UI management tools to create complex UI interactions. **Assessment suggestion:** Have students design and implement a complex menu flow in the application state.	● Program methods that can be called with UI event triggers to add UI functionality from Unity's Inspector window ● Program scripts to access UI components during runtime for systems such as tracking score or responding to user interaction ● Interpret existing code to predict the outcome of an event assigned to a UI component ● Interpret UX wireframes to create a defined menu flow ● Adjust the timing of GameObject movement based on the user's frame rate	**Unity Learn** ● Creating basic UI with uGUI ● User Interface **Unity Manual** ● Data binding ● Create user interfaces (UI)	● Associate: Programmer



Working with materials and shaders

Module introduction

This module provides an in-depth exploration of materials and shaders within the context of computer graphics. Through this module, you will develop a comprehensive understanding of techniques for creating and manipulating textures, surfaces, and visual effects to enhance digital projects. Whether you have prior experience in 3D art or are new to computer graphics, this module offers essential knowledge and practical skills for achieving realistic and visually compelling results.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create a simple shader and material using Shader Graph	● Explain Shader Graph and its uses	**Unity Learn**	● 3D Artist ● Associate: Artist

The Shader Graph tool in the Unity real-time engine allows the user to create custom shaders without code. Understanding this functionality will allow the artist to create special and custom effects for specific render pipelines that are optimized for the target publishing hardware without the need for shader coding knowledge. **Assessment suggestion:** Have students use Shader Graph to create a simple shader effect, like a shimmering material. The Creative Core pathway can be used as a guide for this.	• Create a new shader in Shader Graph • Navigate in the Shader Graph editor window • Connect commonly used Shader Graph nodes to create desired effects • Make a shader with configurable material properties • Make a material from a custom Shader Graph shader	• Make a Flag Wave with Shadergraph • Introduction to ShaderGraph • Get started with Shader Graph **Unity Manual** • Shader Graph **Other Resources** Create shaders and visual effects with URP	
Create and edit shaders using Shader Graph Students must understand the basics of what shaders are and how they are used to affect how the audience experiences objects in Unity. **Assessment suggestion:** Have students describe the creation of	• Create a shader using Unity Shader Graph	**Unity Learn** • Get started with Shader Graph • Introduction to ShaderGraph • Shader Graph: Multiply • Shader Graph: TIme Node **Unity Manual** • Getting started with Shader Graph	• Associate: Artist

and uses for shaders, including object and environment applications.		- Shader Graph - Node Library	
Create materials for the URP/Lit Shader on a 3D GameObject Students will learn to use Unity's fully-featured suite of tools to create, apply, and alter textures and materials to modify the appearance of their models. **Assessment suggestion:** Have students dress models using materials and textures created in and imported into Unity, and adjusted using Unity's native tools.	- Create a new material - Organize materials as project assets - Adjust the Base Map of a material using a color - Adjust the Base Map of a material using an image - Apply the Specular and Metallic workflows to achieve desired effects - Apply alpha clipping in a material - Apply the transparent surface type to a material - Add a normal map to a material - Fix broken (magenta) materials	**Unity Learn** - Shaders and Materials **Unity Manual** - Unity - Manual: Materials and shaders	- User Digital Artist - Associate Game Developer
Decide among common shaders to use for a given project Students must understand the basics of what shaders are and how they are used to affect how the audience experiences objects in Unity.	- Define a mesh, its characteristics, and its use in rendering a 3D GameObject - Determine the shader type for an object based on the design requirements	**Unity Learn** - Introduction to ShaderGraph - Shaders and Materials **Unity Manual** - Shader Graph	- User Digital Artist

Assessment suggestion: Have students describe the creation of and uses for shaders, including object and environment applications.	● Explain the role of shaders in the rendering process ● Explain the difference between physically-based and non-physically-based rendering, and reasons for using each ● Explain the difference between a Lit and Unlit shader, and the reasons for using each ● Explain vertex and fragment (pixel) shaders ● Describe use cases for the Universal Render Pipeline shaders provided with Unity	● Unity - Manual: Materials and shaders	
Simulate common substances with physically-based materials As computers have become more powerful and rendering technology has evolved, Physically Based Rendering (PBR) has become more widely available. PBR simulates the real-world principles of physics and light to generate realistic shadows, reflections, ambient	● Identify the characteristics of a real-world surface to be configured in a new material ● Adjust material properties to simulate a given solid substance ● Given a collection of texture files, select appropriate maps to simulate a material	**Unity Learn** ● Physically based shaders and rendering ● Unity DCC live link with Substance Painter ● Baking Texture Maps in Substance Painter - Unity Learn **Other Resources** ● Adobe Substance 3D	

light, and other effects of light on 3D surfaces. **Assessment suggestion:** Explain the difference between physically-based and non-physically-based rendering, and reasons for using each		● Substance 3D plugin for Unity - Adobe Substance 3D ● Substance 3D Tutorials ● Substance Forum	

Lighting in Unity

Module introduction

Welcome to the module on lighting in Unity, where we'll delve into the essential aspects of illuminating your virtual worlds. Throughout this module, we'll explore the diverse array of light types available in Unity, including directional, point, and spotlights, and how to effectively utilize them to shape your scenes. Additionally, we'll discuss the concept of light baking, a vital technique for optimizing real-time rendering performance. Understanding the pivotal role of lighting in crafting cinematic graphics, we'll guide you through the principles and practices that bring your virtual environments to life with dynamic and visually compelling illumination.

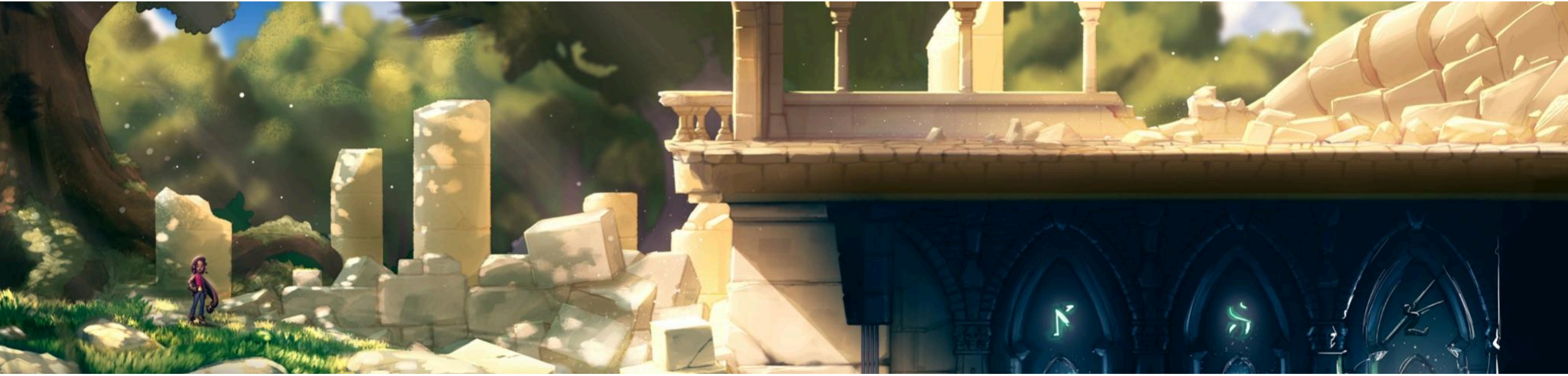
Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Configure light sources and shadows in order to functionally light a scene The Unity Editor provides different light types that simulate various real-world light sources. Understanding when and where to use a specific light type will assist students in creating believable and well-designed scenes. **Assessment suggestions:** Have students use different light types and techniques to create two different aesthetics or moods in a scene.	● Describe the role of the Directional Light in a scene ● Configure the Directional Light in a scene to achieve common effects ● Identify the differences between the different types of Light component ● Configure Light components to achieve common lighting effects ● Configure shadows in the Render Pipeline asset to achieve realistic effects ● Add emissive materials to a scene ● Check emissive materials in a diagnostic view	**Unity Learn** ● Lighting ● Lighting in Unity ● Introduction to Lighting and Rendering **Unity Manual** ● Lighting ● Types of light **Other Sources** ● 7 Ways to Optimize your Unity Project with URP	● 3D Artist ● Associate Game Developer
Decide the appropriate lighting system in order to achieve common outcomes in a Universal Render Pipeline (URP) project Lighting is a complex topic, and the intricacies of lighting can	● Define the term global illumination ● Identify Unity's Global Illumination system for URP ● Explain the main differences between	**Unity Learn** ● Bake a lightmap for your scene - Unity Learn ● Configuring Light Probes **Unity Manual** ● Lighting configuration workflow	● Associate: Artist

make a huge difference both in how a scene is perceived and how it performs in play. **Assessment suggestion:** Have students light a scene to resemble a famous still from a given movie, paying attention to the quality, color, and performance of the scene.	real-time and baked lighting in Unity	- Pipeline | Universal RP - Lighting in the Universal Render Pipeline - Direct and indirect lighting **Other Sources** - Guide to lighting in the HDRP - The URP 3D Sample - Introduction to the Universal Render Pipeline for advanced creators	
Identify and configure lighting techniques and effects using Light settings Unity's advanced lighting options allow for much more complex lighting, creating deeper, more immersive experiences. **Assessment suggestion:** Have students use Unity's advanced lighting options to create scenes with expressive, interesting lighting.	- Interpret design requirements to determine whether to use real time or mixed lighting modes (ProArt) - Configure shadow settings, including width and bias, to achieve realistic effects - Determine the settings of the mixed lighting mode to satisfy application design requirements (ProArt) - Create lighting effects such as halos and flares - Select lighting effects to achieve stylistic results	**Unity Learn** - Lighting - Creating Believable Visuals - Configuring Light Probes - Types of light **Unity Manual** - Introduction to lighting - Lighting - Types of light	- Associate Game Developer

	● Recognize uses of area lights to create lighting for specialized scenarios such as shaped lights and architectural visualizations. (ProArt) ● Determine Lightmapper type based on project needs (ProArt) ● Identify the light type required for a specific effect based on project needs. (ProArt) ● Configure the shape of a light to achieve a desired effect		
Light a scene in a manner that will simulate the real-world behavior of light Lighting is a complex topic, and the intricacies of lighting can make a huge difference both in how a scene is perceived and how it performs in play. **Assessment suggestion:** Have students light a scene to resemble a famous still from a given movie,	● Describe the fundamentals of the behavior of light ● Explain at a high level the difference between direct and indirect light ● Identify light sources in an image ● Explain at a high level the relationship between lighting and post-processing ● Identify key considerations for lighting	**Unity Learn** ● Lighting ● Configuring Light Probes ● Lighting ● Types of light **Unity Manual** ● Direct and indirect lighting ● Lighting ● Types of light ● Post-Processing overview	● User Digital Artist ● Associate Game Developer

paying attention to the quality, color, and performance of the scene.	indoor and outdoor scenes realistically ● Identify research topics and resources to develop your understanding of foundational lighting science and design principles		



Animating in Unity

Module introduction

In this module, you will learn the essentials of animating in Unity. We will cover key concepts such as keyframe animation, rigging, and using the Animator component, alongside practical scripting techniques. By the end of this course, you'll be equipped to create dynamic and responsive animations for your Unity projects.

For a deep dive into the whole animation development cycle in Unity, we suggest you make use of Unity for Animation: Road to Realtime Live Series, a series of live sessions in Unity that takes a deep dive into every part of the animated storytelling process, from previz to final pixels, as well as the The definitive guide to animation in Unity ebook.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Animate using the 2D Animation package The 2D Animation package is a solution for effortless skeletal animation of 2D sprites, featuring in-editor rigging, bone weight painting, and Inverse Kinematics support. Assessment Suggestion: Have students install the 2D Animation package and animate a simple sprite using the available tools.	● Create bones using the 2D Animation Package's Skinning Editor ● Generate a mesh for a 2D rig in the 2D Animation Package ● Adjust weights on a 2D rig in the 2D Animation Package ● Use a Sprite Skin component to generate a 2D rig in the 2D Animation Rigging package	**Unity Learn** ● Rigging a Sprite with the 2D Animation Package ● Animating a Sprite with the 2D Animation Package **Unity Manual** ● 2D Sprite Shape ● Animation Rigging ● Introduction to 2D Animation ● AI Navigation | AI Navigation | 2.0.8 **Unity Blog** ● Getting Started with Unity's 2D Animation Package	
Configure animation clips imported from digital content creation (DCC) software or the Asset Store for use in a project The world around you is in constant motion. The same is true for digital worlds. A static	● Apply imported animation clips to rigged models in Unity ● Configure a humanoid rig to share animations between characters ● Trim animation clips to access specific	**Unity Learn** ● Creative Core: Animation ● Working with Animation Rigging ● Working with Animations and Animation Curves ● Retargeting and Reusing Animation	● Associate Game Developer

environment tends to appear unfinished or cold and unfeeling; animation is all about creating the illusion of life. In this mission, you will learn how to create animations in the Unity Editor and how to configure animations imported from an external program. You'll apply these concepts to add animation to objects and characters in your scenes, and you'll even control when the animation gets played. **Assessment suggestion:** Have students create a looping animation using imported animated assets or create their own animated asset in the Unity Editor.	keyframed sequences within them ● Identify the purpose of a specified parameter of an animator controller ● Describe the relationship between parameters and transitions ● Define animator ● Define different rig types and their uses ● Describe how an avatar is used with a humanoid rig to share animation ● Describe how an avatar is used with an animator controller to control animation ● Define the different rig types and their uses	**Unity Manual** ● Animation	
Create a keyframed animation sequence using Unity's animation editor The world around you is in constant motion. The same is true for digital worlds. A static environment tends to appear unfinished or cold and unfeeling;	● Add keyframes to an Animation Clip ● Move keyframes along the timeline in the Animation Editor ● Assign an animator controller to a GameObject's Animator component	**Unity Learn** ● Creative Core: Animation ● Introduction to 3D animation systems ● Unity for Animation: Road to Realtime Live Series ● Get started with animation **Unity Manual**	● Associate: Artist ● 3D Artist

animation is all about creating the illusion of life. In this module, you will learn how to create animations in the Unity Editor and how to configure animations imported from an external program. You'll apply these concepts to add animation to objects and characters in your scenes, and even control when the animation gets played. **Assessment suggestion:** Have students create a looping animation using imported animated assets, or create their own animated asset in the Unity Editor.	● Record a GameObject animation using Record Mode ● Set up a new Animation Clip ● Move an animation in a scene using a parent GameObject ● Open the Animation Editor window ● Define keyframes ● Define tweening ● Explain what a playhead does ● Select the view of the Animation Editor's timeline to display seconds or frames ● Set the sample rate of an animation clip ● Explain how the dopesheet is used in the Animation Editor window	● Animation ● Animation Rigging ● Blend Trees	
Create basic 2D animations with Sprites Parameters allow students to control a number of elements about an animation, such as its speed or state.	● Recognize the process for automatically generating 2D animation from Sprite sheets (ProArt) ● Identify methods within the animation controller used to customize	**Unity Manual** ● Sprite editor ● Introduction to 2D Animation ● 2D Animation ● AI Navigation	● 3D Artist

Assessment suggestion: Have students create three animation states for an animation and three parameters that can be altered at runtime with a script.	animation states for Sprites (ProArt) ● Identify editor animation parameters necessary to fine-tune sprite animations (ProArt)		
Evaluate the various animation types in order to determine which one to use Unity provides internal animation tools, as well as the option to import animations from their party applications. Understanding the difference between imported and Unity-created animations will assist the user in determining the most efficient workflow. **Assessment suggestion:** Have students import an animation and create a native keyframe animation in Unity, then contrast and discuss the advantages and disadvantages of each.	● Explain the difference between animations imported into Unity and animations created within Unity ● Name software products from which you can import models with animations into Unity ● DIfferentiate movements created with physics from those created with animation	**Unity Learn** ● Editor Essentials ● Creative Core: Animation ● Control animation with an Animator ● AI Navigation ● Introduction to 3D animation systems ● Unity for Animation: Road to Realtime Live Series	● 3D Artist ● Associate Game Developer
Set up a new NavMesh in a scene	● Explain backfilling with a NavMesh	**Unity Manual** ● Navmesh | AI Navigation ● Create a NavMesh	

Unity's navigation system lets developers create AI-powered enemy agents that can intelligently move around the game world, using navigation meshes (NavMeshes) created automatically from the scene geometry. Students implementing navigation and pathfinding should have some understanding of scripting. **Assessment suggestion:** Have students bake a NavMesh allowing agents to move across the terrain, avoid objects, and chase a player avatar.	● Explain how maximum slope is used with a NavMesh ● Explain how obstacle avoidance is used with a NavMesh	● Create a NavMesh Agent ● Create a Navmesh Obstacle ● Create a NavMesh Link ● Use NavMesh Agent with Other Components ● Build a HeightMesh for Accurate Character Placement	
Use basic state machines and blend trees to create and manage multiple animations The Unity real-time engine provides numerous tools and plugins that allow the artist to link the Unity Editor with their modeling or texturing applications of choice. Understanding the available options and how to implement them will assist the artist in	● Distinguish between transition-based and layer-based approaches to building state machines. (ProArt) ● Recognize uses for sub-state machines within an animator controller. (ProArt) ● Recognize uses for blend trees within an animator controller. (ProArt)	**Unity Learn** ● Control animation with an Animator ● Explore the Animator Controller ● Explore State Machines	● Associate: Artist ● 3D Artist ● Associate Game Developer

setting up an efficient workflow and technology stack. **Assessment suggestion:** Have students use Substance Designer or Substance Painter to create a workflow setup between the Unity Editor and their Substance application and create materials for at least one of their assets in a scene.	● Identify approaches to working with different parameter types for animation state transitions. (ProArt) ● Modify individual animations for use within a state machine. (ProArt)		

Professional skills

Module introduction

Professional skills are some of the most broadly applicable and easily transferable of the skills that are highlighted in the curricular framework. The learning objectives here focus on the soft skills students should have to secure a position in the industry and for ongoing growth and success as part of a team.

This module prepares students for a new career move by introducing them to the specific roles available to them in the industry, as well as the importance of showcasing their work and skills through the creation of compelling portfolios that present them in the best light possible. Students are also introduced to different iterative design approaches and the fundamentals of project management.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create a portfolio for a job in real-time development To successfully begin a career journey in their chosen industry, students should take an active role in choosing, achieving, and demonstrating competency in their learning goals and using that knowledge to prepare for work. **Assessment suggestion:** Have students write a short description of a specific role or set of roles in a game studio, explaining the skills required to complete the role successfully, the kinds of duties usually associated with the role, and the expectations that the role requires of applicants.	● Describe the goals, purposes, and uses of a portfolio ● Describe tools for building a portfolio ● Describe various types of portfolios ● Explain what goes into a professional portfolio ● Plan a portfolio by using a flowchart ● Organize content in a portfolio	**Unity Learn** ● Introduction to portfolios	
Lead projects in the real-time development cycle	● Explain how downloaded AssetBundles and content catalogs are cached	**Unity Learn** ● Roles and careers for real-time creators	

In the industry, successful teams use various technologies within a design process to identify and solve problems by creating new, practical, or imaginative solutions. **Assessment suggestion:** Describe and enact the steps of iterative design: identifying a problem, researching the context, enacting a solution, and iterating on the solution.	• Advise clients with contextual information to make the technology more understandable to them • Solve problems to address client needs with efficiency and creativity	• Career research and preparation • Develop your learning plan • Job preparation	
Manage projects in the real-time development cycle In the industry, successful teams use various technologies within a design process to identify and solve problems by creating new, practical, or imaginative solutions. **Assessment suggestion:** Describe and enact the steps of iterative design: identifying a problem, researching the context, enacting a solution, and iterating on the solution.	• Explain the importance of time management in the project management process • Explain the roles of communication and professionalism in the project management process • Organize a QA testing plan for a project • Explain the reasons to conduct a retrospective after a project is completed	**Unity Learn** • Introduction to project management and teamwork • Introduction to user feedback and testing • The real-time production cycle	

Plan projects in the real-time development cycle In the industry, successful teams use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions. **Assessment suggestion:** Have students describe and enact the steps of iterative design, identifying a problem, researching the context, enacting a solution, and then iterating on the solution.	● Explain how a design document or project brief is used in a project ● Explain the importance of defining purpose, goal, and audience ● Organize project tasks based on production roles ● Describe the structure and content of design documents ● Explain the uses of a project charter ● Investigate appropriate applications for project management	**Unity Learn** ● Introduction to real-time 3D experience design ● Introduction to user feedback and testing ● The real-time production cycle ● Introduction to project management and teamwork	

Unity Gaming Services

Module introduction

Unity Gaming Services is an end-to-end platform that is designed to help you build, engage, and grow your game.
These services allow you to take your game to the next level without having to worry about maintaining or scaling your back-end infrastructure and simplify many game development tasks and challenges.
UGS support your entire development lifecycle and can be used to build your foundation, engage your players, and grow your game.

Examples include:
-Add multiplayer and social features to your game.
-Use server-side game logic to ensure a level playing field.
-Enable your players to access their game data across various gaming platforms.
-Run A/B tests and measure gameplay data from various services to inform design decisions.
-Deliver fresh content without updating your app.
-Run scheduled events and provide varied content to your game during those events.
-Engage players with fun, progressive reward and loyalty systems.

Read more about Unity Gaming Services

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create a multiplayer game using Unity services (CFW) Unity Gaming Services (UGS) provides a host of services to assist you in creating multiplayer functionality in your game without having to worry about building and maintaining servers and related online products. **Assessment suggestion** Have students register their game with UGS and implement basic multiplayer functions.	- Set up multiplayer over internet functionality for a Unity game using Unity Gaming Services (CFW) - Set up local multiplayer functionality for a Unity game using Unity Gaming Services (CFW)	**Unity Learn** - Unity 6: Multiplayer Creation - **Unity Manual** - Multiplay - Matchmaker - Vivox Unity SDK - Friends **Unity Blog** - Master multiplayer **Other Resources** - Netcode for GameObjects - Boss Room - Lobby - Relay - Vivox voice chatting - How to set up Matchmaker - VALORANT: A Unity case study	

Set up backend services for a game using Unity services (CFW) Unity Gaming Services (UGS) provides a host of services to assist you in building and growing your user base, as well as increasing engagement for user retention. **Assessment suggestion: Have** students register their game with UGS and implement basic engagement tools.	- Set up backend services to manage and improve player retention (CFW)	**Unity Manual** - Unity Analytics - Unity Authentication - Cloud Code - Unity Cloud Content delivery - Unity Cloud Diagnostics - Cloud Save - Economy - Unity Game Overrides - Leaderboards - Push Notifications - User Generated Content	

Unity AI

AI can help you to be more productive while staying fully in control of your vision. It offers the possibility of in-game features and capabilities that couldn’t be built otherwise, potentially revolutionizing player experiences by embedding AI models in the runtime so content reacts and responds to players and users in new ways.

We’re harnessing the power of AI to drive innovation, accelerate content creation, and increase your productivity across games, entertainment, and industrial use cases. We’ve been building a suite of AI tools that promise to accelerate creation time and complement your workflows by finding information and generating draft assets as quickly as typing in a text prompt or scribbling a sketch. From there, you could integrate work with familiar tooling to revise and edit the assets you need at a speed that’s unimaginable with today’s workflows.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Set up your project with Unity AI Set up your Unity project and prepare to use Unity AI tools in your prototyping workflow. **Assessment suggestion:** Have students set up a new Unity project and configure it for use with Unity AI tools, demonstrating their readiness to integrate AI-assisted features in a prototyping workflow.	- Set up your Unity project - Use Unity AI tools in your prototyping workflow.	**Unity Learn** - Set up your project with Unity AI	
Use Assistant to set up your scene Use Unity's built-in AI Assistant to block out your scene through natural language prompts. **Assessment suggestion:** Have students use Unity's built-in AI Assistant to block out a basic scene by entering natural language prompts, demonstrating their ability to translate design intentions into	- Block out a scene in Unity using natural language prompts with the built-in AI Assistant.	**Unity Learn** - Use Assistant to set up your scene	

functional scene layouts using AI tools.			
Use Animation Generator to bring your character to life. Use Animation Generator to bring your character to life	- Animate a character using the Animation Generator to bring it to life within a Unity project.	**Unity Learn** - Use Animation Generator to bring your character to life	
Use Material and Texture Generators to make the environment Customize the look of your scene by generating a background texture and a ground material. **Assessment suggestion:** Have students use the Animation Generator tool to animate a character model, demonstrating how AI-generated animations can be applied to bring characters to life within a Unity project.	- Customise the visual style of your scene by generating a background texture and ground material.	**Unity Learn** - Use Material and Texture Generators to make the environment	
Use Sprite Generator to make an avatar icon	- Create a character avatar using the Sprite Generator with natural language	**Unity Learn** - Use Sprite Generator to make an avatar icon	

Use the Sprite Generator to create a character avatar through natural language prompts and image references. **Assessment suggestion:** Have students use the Sprite Generator to create a character avatar by combining natural language prompts with image references, demonstrating how to generate and refine 2D assets using AI-assisted workflows.	prompts and image references.		
Use Sound Generator for background audio Use the Sound Generator to create custom sound effects, which can play at random intervals in the background to make your scene feel more immersive. **Assessment suggestion:** Have students use the Sound Generator to create custom ambient sound effects and implement them to play at	● Generate custom sound effects with the Sound Generator and implement them to play at random intervals for enhanced scene immersion.	Use Sound Generator for background audio - Unity Learn	

random intervals in the background, demonstrating their ability to enhance scene immersion through procedural audio design.			