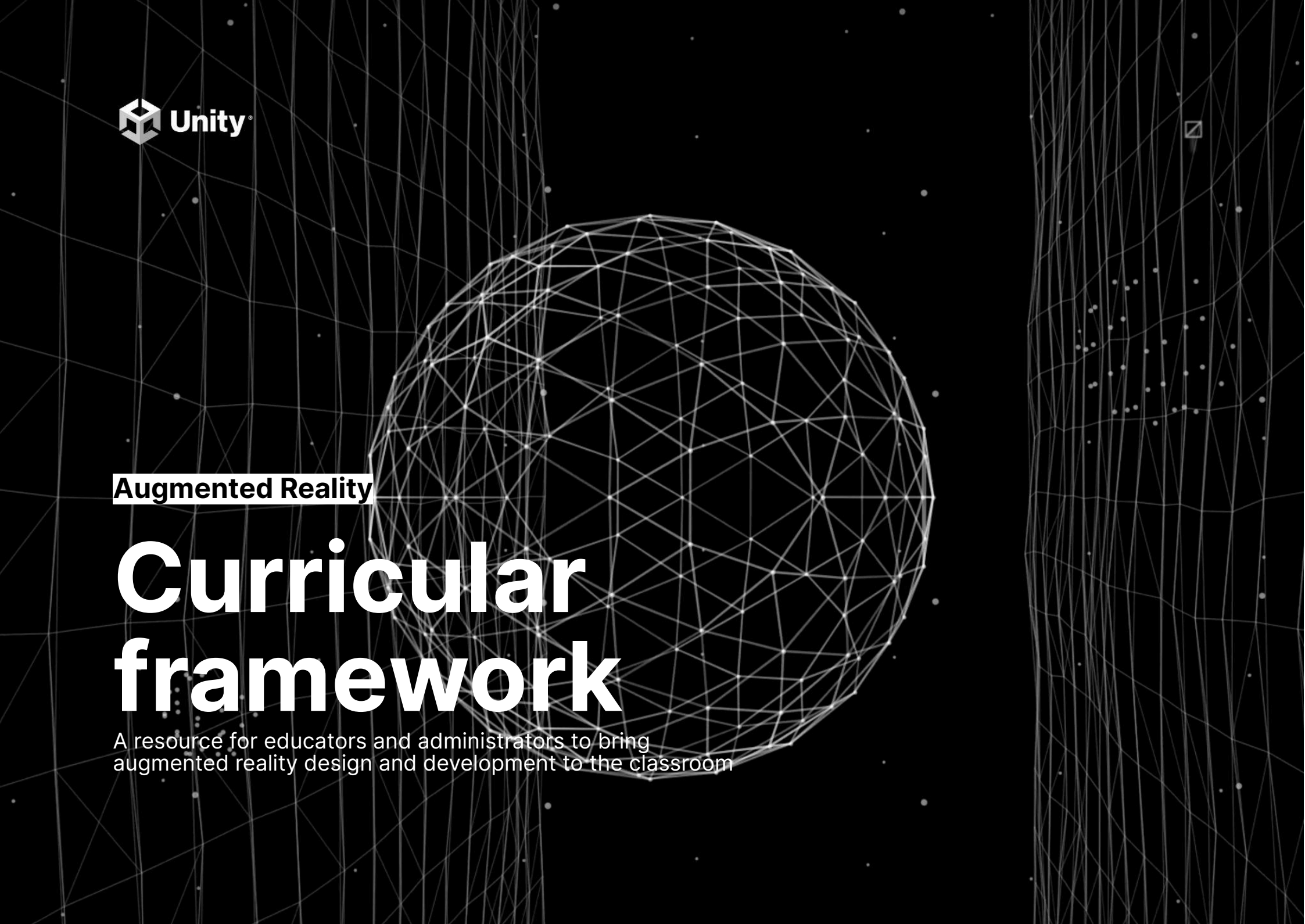


Augmented Reality

Curricular framework

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

Table of Contents

Introduction

Why this framework?

Augmented reality, as a key technology in the current technological revolution, is leading us to new frontiers and dramatically impacting individuals, organizations, and environments. Whereas VR removes the user from their real-world experience and replaces it with a completely simulated one, AR overlays information and virtual objects on real-world scenes in real-time to modify the user's experience of the environment. Mixed reality allows the user to interact with elements from both the digital world and the real world. Unity is the most widely used real-time 3D creator platform and powers the majority of extended reality (XR) and mixed reality (MR) experiences available today.

The XR industry is growing rapidly and real-time 3D skills are in high demand, across industries. Augmented reality and mixed reality are used for engagement across education and marketing experiences and have immensely practical applications in industries spanning field service, healthcare, retail, tourism, utilities and others. AR apps range from popular and accessible visualization aids (such as furniture placement or map navigation) to highly specialized tools (such as medical diagnostic instruments, training aids, and maintenance guides).

The goal of this curricular framework is to enable teachers to quickly establish or expand innovative programs involving augmented 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 User 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 real-time visualization landscape and platform development tools change, so we recommend that you check back periodically to ensure you have the latest version.

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)	• Recognize the effects that Rigidbody and Collider components have on performance	**Unity Learn** • Optimization	

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.	• 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		

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 Artist and Associate: Programmer certifications. These are appropriate credentials for students looking to make the transition into professional work. Holding an Associate Credential 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.

If you are interested in preparing your students for certification, we have provided a certification map in the appendix to highlight all the relevant certification exam objectives and the modules in which they can be found.

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	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.	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.

developer. **Covers** • Scripting, building, debugging, • and optimization. • Prototyping art assets, whiteboxing levels.	**Covers** • Advanced Unity tools. • Scripting in C# for Unity. • Scripting UI elements.	**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.

Unity hardware requirements

You can find the latest Unity hardware requirements in the Unity documentation. Go to Unity – Manual and then click on *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 have vast knowledge and experience. 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 Teach Facebook group
A place for Unity educators to come together, access resources, and share best practices, with the goal of enabling success for their students.

Unity Forums
An extensive archive of knowledge about the Unity ecosystem; consult the forums for insight and support. You can find out the latest developments, submit feedback, and engage with the developers of Unity.

Unity Discussions
Beginners and experts alike post on 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 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.

Discord
Discuss Unity and the Partner School program in real-time. Join our Discord server, or join the Partner School 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. To start, take a look at the Unity, Unity3D, and Unity2D communities.

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

00.F

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 Forums, 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!)

Modules in this Curricular Framework

Introduction to the Unity real-time platform	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.
Introduction to augmented reality	Unlike virtual reality experiences, which are typically spatially contained and made for specific hardware, augmented reality can be implemented more widely across more diverse types of hardware. This module is theoretical and acts as a good primer for courses on augmented reality introducing students to the history and current trends and capabilities of the technology. Understanding the different opportunities, constraints, accessibility and safety requirements enables students to assess the suitability of their ideas and design AR applications that make the most of the technology.
Building scenes	In this module, we will go over common techniques for creating and importing objects into a scene in Unity as well as setting up and building your scene. Our goal is to outfit students with the skills necessary to build functional, dynamic environments. While the Unity real-time engine provides tools to quickly create prototypes or basic scenes, most advanced assets will probably be created in other dedicated software applications.
Working with materials and shaders	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.

Lighting in Unity	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.
Working with AR Foundation	AR Foundation serves as a powerful and flexible framework within Unity, designed to simplify the development of augmented reality (AR) applications across both iOS and Android devices. This framework is crucial as it bridges the gap between Unity and the native AR capabilities of different platforms, like ARKit for iOS and ARCore for Android. Throughout this module, we will guide you through the essential steps of setting up AR Foundation in a new project or scene, ensuring you have a strong foundation for AR application development.
Working with Unity MARS	This module will introduce you to Unity Mars, a powerful tool designed to simplify AR app development by overcoming challenges in authoring for dynamic physical environments and reducing development time. Unity Mars supports cross-platform experiences, including iOS, Android, and HoloLens, and enhances AR workflows with features like plain-language authoring, in-editor app testing, quick-start templates, and mobile device authoring. Additionally, the AR Companion app for Unity Mars further streamlines the process by capturing environment data, enabling object scanning (Mac required), and allowing light AR scene editing directly on the device. This module will give you the essential skills and knowledge to efficiently create immersive AR experiences using Unity Mars.
World space UI & touch interfaces	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.v
Animating in Unity	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.
VFX in Unity	In this module, you will learn about Visual Effects (VFX) in Unity, focusing on the dynamic and versatile tools of particle systems and the VFX Graph. These powerful features are essential for creating stunning visual effects that can bring your Unity projects to life. You will learn how to use particle systems for simulating complex phenomena like fire, smoke, and water, as well as how to leverage the VFX Graph for more advanced, customizable effects.
Introduction to C# in Unity	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.
Unity Gaming Services	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.
Professional skills	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.

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: Get started with Unity 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.

Unity also provides a growing range of services, most with free tiers of use, to help developers build, manage, and grow the 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 Unity versions and additional	• Install a version of the Unity Editor using the Unity Hub • Create a new Unity project using a template in the Unity Hub	**Unity Learn** • Start creating • Install a new Unity Editor • Create a new project • Add new modules to a Unity Editor	

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 punderstanding 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.	• 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	• Install a package via the Package Manager • Project Organization • Introduction to XR: VR, AR, and MR Foundations **Unity Manual** • Install the Unity Hub • The Project window • Using the Asset Store	
Create and manage Scenes	• Explain the role of scenes in a Unity project	**Unity Manual** • Scenes • Scene view navigation	• Associate: Game Developer

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, labeling 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.	• Create a new empty 3D Scene • Create a new empty 2D Scene • Open a scene in a Unity project		
Identify and use essential features of the Unity Editor	• Identify and describe the windows that appear in	**Unity Manual** • The Project window	• Associate: Game Developer

Different areas of the Unity Editor interface are used to complete different tasks. Ideally, learners should be able to identify and explore these features of the Editor prior to focusing on developing for VR. As the Unity Editor is a professional tool, there is a lot to learn. The free - Get started with Unitycourse provides learners with a comprehensive guide to these essential features.	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's interface • Scene view navigation • GameObjects • Tags and Layers • Using the Asset Store	

	• Organize assets using folders in the Project window		
Employ Unity Version Control in a project Plastic SCM is a version control to help teams focus on delivering work, one task at a time. Recently purchased by Unity, this tool is set to become the de facto collaboration tool for teams using the Unity Editor. **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 Manual** • Plastic SCM • Collaborate to Plastic SCM Migration Wizard Guide **Unity Resources** • Version Control for Games with Unity's Plastic SCM • Getting Started Webinar: Upgrading from Collaborate to Plastic SCM **Other Resources** • Collaborate to Plastic SCM Migration Wizard Guide • Plastic for Unity Plugin Gluon Mode guide	• Associate: Game Developer • Associate: Programmer
Evaluate Unity and real-time 3D in order to determine whether they are suited to your needs	• Define the term real-time • Explain what the Unity real-time engine does	**Unity Learn** • Real-time creation	

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.	• 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		

Introduction to augmented reality

Module introduction

Unlike virtual reality experiences, which are typically spatially contained and made for specific hardware, augmented reality can be implemented more widely across more diverse types of hardware. This module is theoretical and acts as a good primer for courses on augmented reality introducing students to the history and current trends and capabilities of the technology.

Understanding the different opportunities, constraints, accessibility and safety requirements enables students to assess the suitability of their ideas and design AR applications that make the most of the technology.

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.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications

Configure an AR project for a mobile device. Apple (ARKit) and Google (ARCore) provide AR toolkits for their respective mobile operating systems. Knowing how to install the relevant packages and set up the Unity Editor for AR development with your preferred toolkit is crucial for creating augmented reality applications. **Assessment suggestion:** Have students download and install the relevant packages to create an AR development template project. The more adventurous can install the beta version of the Unity hub and explore the AR template that is provided.	• Set up a project with appropriate packages and project settings for AR development • Deploy a mobile application to an Android device • Deploy a mobile application to an iOS device	**Unity Learn** • Deploy your AR project to iOS or Android **Unity Manual** • Unity AR project template	
Explain the affordances and limitations of augmented reality as a medium. The potential use cases beyond what can currently be done with established media platforms makes TR an exciting new frontier for creators and	• Explain the current trends and future applications of augmented reality technologies • Evaluate the current state of augmented reality platforms	**Other Resources** • Unity Immersive Technologies Glossary • Practical Augmented Reality: A Guide to the Technologies, Applications,and Human Factors for AR and VR, by Steve Aukstakalnis	

researchers. What we consider to be best practice is changing frequently as new problems and solutions are discovered. Learners who understand the affordances and limitations of mediums will make better informed decisions when designing and developing VR. **Assessment suggestion:** Have students research and discuss the affordances and limitations of Augmented reality as a medium.		• Augmented Reality: Innovative Perspectives Across Art, Industry,and Academia, Edited by Sean Morey and John Tinnell	

Building scenes

Module introduction

In this module, we will go over common techniques for creating and importing objects into a scene in Unity as well as setting up and building your scene. Our goal is to outfit students with the skills necessary to build functional, dynamic environments. While the Unity real-time engine provides tools to quickly create prototypes or basic scenes, most advanced assets will probably be created in other dedicated software applications. This module will also go over the process of setting up a workflow between the Unity real-time engine and these third party 3D modeling applications. Section B in this module deals with the development of 2D applications using the Unity Editor. We provide links to relevant tutorials on the Unity Learn platform, but would also suggest the 2D game art, animation, and lighting for artists ebook as a useful resource for all things 2D in Unity.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications

Choose an appropriate render pipeline for a project, given certain requirements. Unity provides two render pipelines that are optimized for specific hardware and use cases, as well as one legacy system for backward compatibility. Knowing which pipeline to use will allow students to create optimized experiences from the initial planning to the final production stages. **Assessment suggestion:** Have students create a cinematic scene using the High Definition Render Pipeline (HDRP), and then recreate the same scene in the Universal Render Pipeline (URP) to make it mobile-friendly.	• Define key terms of real-time graphics including rendering, render pipeline, and scriptable render pipeline • Identify the differences between Unity's provided render pipelines, including advantages, disadvantages, and common use cases for each • Create a new project using a particular render pipeline	**Unity Learn** • Understanding Scriptable Render Pipelines • Introduction to URP • Introduction to HDRP **Unity Manual** • Render pipelines • Universal Render Pipeline overview • High Definition Render Pipeline overview	• Associate: Artist
Configure a post-processing profile to achieve a specific visual style Post-processing allows the user to enhance their visual aesthetic using cinematic effects that are applied by the real-time engine	• Set up a camera to allow for post-processing • Add a new global post-processing volume to the scene and assign a new post-processing profile	**Unity Learn** • Post-processing • Introduction to Post Processing Stack • Post-processing part of Unity for Animation: Road to Real-time Live Series	• Associate: Artist • Professional 3D Artist

just before rendering to the scene. This process allows for a global or local volume control over the visuals of your scene, and it can go a long way to adding a professional veneer to your work. **Assessment suggestions:** Have students try to recreate a chosen cinematic scene using post-processing effects. Alternatively, give students a still from a film and have them identify the effects used. CG-heavy films, like those from Marvel, are good candidates for this.	• Describe the purpose of a post-processing profile • Modify a post-processing profile by adding and removing module overrides • Describe common post-processing effects, such as bloom, depth of field, tonemapping, and color adjustments • Add a new local post-processing volume to the scene and edit its boundaries • Describe scenarios where a global volume or local volume would be more appropriate • Import post-processing effects into a Unity project	**Unity Manual** • Post-processing overview	
Employ basic physics for GameObjects The Unity physics system allows the artist to simulate real-world, or unrealistic physics in their scene. Implementing the correct physics allows an artist to create	• Apply Rigidbody or Rigidbody 2D components in order to enable GameObjects to act under the control of physics • Create and configure a Physic Material to add	**Unity Learn** • Add physical properties to 3D GameObjects	• Associate: Game Developer

the required interaction and effects in their scene. **Assessment suggestion:** Have students add physics to assets to make them behave like their real-world equivalents, for example changing a sphere into a bouncing ball.	physical properties to a GameObject • Modify the basic properties of a Rigidbody or Rigidbody 2D component in order to control how the GameObject is affected by physics, including its mass, its drag, and the Scene's gravity • Add a collider to a GameObject		
Employ prefabs in order to manage the GameObjects in a scene or project Prefabs and Nested Prefabs let students build complex objects with parts that have the flexibility and power of Prefabs. **Assessment suggestion:** Have students implement a Nested Prefab and understand how to move in and out of layered Prefab editors.	• Explain how to use prefabs in a scene • Add a prefab to a project • Identify a prefab in the Project window • Edit a prefab in prefab mode • Make a prefab variant • Apply or revert changes to a prefab variant • Identify when a nested prefab or prefab variant is in use • Describe the process and outcomes for changing a nested prefab or prefab variant	**Unity Learn** • Introduction to Nested Prefabs • Introduction to Nested Prefabs **Unity Manual** • Prefabs • Nested Prefabs	• Associate: Programmer • Associate: Game Developer

Import model files and custom packages into Unity Unity provides tools to optimize the workflow between the Editor and most industry-standard DCC applications. Understanding how to make use of this workflow will allow the student to speed up the process of bridging 3D art and assets in the Unity Editor. **Assessment suggestion:** Have students export a scene or asset from the Unity Editor using the FBX exporter, update the materials or model in the third-party modeling software of choice, and then re-import the model, keeping the same orientation, position, and scale.	• Configure import settings for FBX files • Complete an FBX round-trip • Configure import settings for OBJ files • Import models with materials and textures • Import and configure assets from a custom package	**Unity Learn** • Get 3D assets **Unity Manual** • Import a local asset	• User: Artist • Associate: Game Developer
Integrate external assets and tools into your prototype Unity provides tools to optimize the workflow between the Editor and most industry-standard DCC applications. Understanding how to make use of this workflow will allow the student to speed up	• Import a third-party character controller • Identify third-party assets and resources for a prototype • Create an asset inventory	**Unity Learn** • 3DS Max to Unity Pipeline • The Maya to Unity Pipeline • Get 3D assets **Unity Manual** • Importing objects from Maya	• Professional 3D Artist

the process of bridging 3D art and assets in the Unity Editor. **Assessment suggestion:** Have students export a scene or asset from the Unity Editor using the FBX exporter, update the materials or model in the third-party modelling software of choice, and then re-import the model, keeping the same orientation, position, and scale.		• Importing Objects from 3D Studio Max **Unity Resources** • Introduction to game level design	
Obtain assets from the Unity Asset Store GameObject assets are at the heart of every Unity scene. Understanding how to create high-quality assets will allow the student to create interesting scenes that meet the aesthetic requirements of the project. **Assessment suggestion:** Have students build a scene, based on a suggested theme, using assets from the Asset Store and other online repositories like - TurboSquid, - CGTrader • Poliigon, or - Poliigon	• Navigate to the Asset Store from the Unity Editor • Search and filter a search in the Asset Store • Import an asset from the Asset Store automatically via a Unity account • Navigate to the Asset Store in a web browser • Import an asset downloaded from the Asset Store in a web browser	**Unity Manual** • Using the Asset Store	• User: Artist • Associate: Artist • Associate: Game Developer

Refine a prototype environment using ProBuilder	• Explain how ProBuilder can support prototype development • Create meshes using Probuilder • Configure geometry to make basic scenery for a prototype • Set a Collider for a mesh • Set a mesh as a trigger	**Unity Learn** • Editing with ProBuilder • Working with Shapes in ProBuilder • Working with Poly Shapes in ProBuilder **Unity Resources** • Introduction to game level design	• Associate: Artist
Refine a prototype environment using Terrain	• Explain how Terrain can support prototype development • Create a new Terrain • Customize Terrain using the Paint Terrain tool	**Unity Learn** • Introduction to Terrain Editor • Working with the Terrain Toolbox **Unity Manual** • World building • Enhance your prototype with Terrain **Other Resources** • Introducing the Terrain Editor | Unite Now 2020 • How to build beautiful landscapes in Unity using Terrain Tools	• User: Artist
Set up a new NavMesh in a scene	• Explain backfilling with a NavMesh	**Unity Learn** • Unity 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	• NavMesh Baking - Unity Learn • NavMesh Agents - Unity Learn • Working with NavMesh Agents - Unity Learn **Unity Manual** • Building a NavMesh	
Use the sprite editor to slice sprite sheets Students can use the Sprite editor to build sprites. The rigger allows sprites to be given bones and weights, which will make them easier to animate. **Assessment suggestion:** Have students create and rig a sprite, ensuring that bones and weights	• Identify parameters in the sprite inspector to configure sprite sheets according to project requirements. (ProArt) • Determine sprite editor slice parameters to isolate sprites in sheets. (ProArt) • Determine settings to manually prepare sprite sheets, including specifying positions,	**Unity Learn** • Introduction to Sprite Editor and Sheets **Unity Manual** • Sprite Editor	• User: Artist • Associate: Artist • Professional 3D Artist

are placed and adjusted correctly.	borders, and pivots. (ProArt) • Recognize uses of the edit outline feature in the sprite editor to customize the shape of a sprite. (ProArt)		
Use tilemaps to create 2D environments Students can use the Sprite Editor to create tilemaps. **Assessment suggestion:** Have students create a tilemap and use it to decorate a background and a set of foreground tiles.	• Create a tilemap using the Sprite Editor • Use a tilemap on a background and a GameObject	**Unity Learn** • Introduction to Tilemaps **Unity Manual** • Tilemap	• Associate: Artist

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** • Shader Graph: Multiply • Shader Graph: TIme Node	• Professional 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	• Introduction to ShaderGraph • Get started with Shader Graph **Unity Manual** • Shader Graph	
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 and uses for shaders, including	• 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** • Shader Graph • Node Library	• Associate: Artist

object and environment applications.			
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 Manual** • Creating Materials for URP	• User: 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 **Unity Manual** • Shader Graph	• User: 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		
Decide the best approach for creating materials for the URP/Lit shader on 3D GameObjects, given project requirements A shader is a script that applies the properties contained in a material to render the meshes of your 3D objects to the 2D image on your screen. Each shader is	• Define material • Explain specular and diffuse reflectivity • Define texture and map as they are used in materials • Explain the maps that are configurable on the URP/Lit Shader Surface Inputs and their various effects: base map, specular/metallic, normal,	**Unity Learn** • Create effects for 3D GameObjects • Shaders and materials **Unity Manual** • Materials	• User: Artist • Professional 3D Artist • Associate: Game Developer

written for a specific render pipeline. **Assessment suggestion:** Have students determine the shader type for an object based on the design requirements	height, occlusion, emission • Define UVs • Explain how 3D modeling programs are used to create assets for Unity materials • Distinguish between Specular and Metallic properties and explain how each is configured • Explain the use of Detail Inputs for the URP/Lit shader • Explain High Dynamic Range color		
Determine materials and textures for objects, and identify advanced settings to achieve a desired effect (ProArt) 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	• Create material styles based on the design requirements, such as cell shading, realism, and minimalism (ProArt) • Identify which types of texture maps are needed to create specific types of materials, such as skin, rock, metals, and incandescent objects (ProArt) • Determine rendering modes for Standard	**Unity Manual** • Materials • Shaders and Materials | Universal RP • Shaders	• Professional 3D Artist

materials and textures created in and imported into Unity, and adjusted using Unity's native tools.	Material to achieve a desired effect, such as transparent or semi-transparent objects (ProArt) • Recognize processes for creating textures and materials for environmental elements such as cube maps and reflection maps		
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 light, and other effects of light on 3D surfaces. **Assessment suggestion:** Explain the difference between physically-based and	• 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 • Substance 3D Tutorials • Substance Forum	

non-physically-based rendering, and reasons for using each			

Lighting in Unity

Module introduction

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	• Describe the role of the Directional Light in a scene	**Unity Learn** • Lighting • Lighting in Unity	• Professional 3D Artist • Associate: Game Developer

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.	• 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	• Introduction to Lighting and Rendering **Unity Manual** • Lighting • Types of light	
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 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	• Define the term global illumination • Identify Unity's Global Illumination system for URP • Explain the main differences between real-time and baked lighting in Unity	**Unity Learn** • Lighting design for cinematic realism • Configuring Light Probes **Unity Manual** • Pipeline | Universal RP • Lighting in the Universal Render Pipeline • Light Baking	• Associate: Artist

resemble a famous still from a given movie, paying attention to the quality, color, and performance of the scene.			
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) • Recognize uses of cookies to achieve complex shadow effects for performance-constrained platforms. (ProArt) • Create lighting effects such as halos and flares • Select lighting effects to achieve stylistic results	**Unity Learn** • Lighting design for cinematic realism • Configuring Light Probes • Lighting • Introduction to Lighting and Rendering • Types of light **Unity Manual** • Light Baking • Lighting • Types of light	• Associate: Game Developer

	• Use layers to create a culling mask that excludes objects from being affected by a light source • 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.	• 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	**Unity Learn** • Lighting design for cinematic realism • Configuring Light Probes • Lighting • Introduction to Lighting and Rendering • Types of light **Unity Manual**	• User: Artist • Associate: Game Developer

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.	lighting and post-processing • Identify key considerations for lighting indoor and outdoor scenes realistically • Identify research topics and resources to develop your understanding of foundational lighting science and design principles	• Light Baking • Lighting • Types of light	

Working with AR Foundation

Module introduction

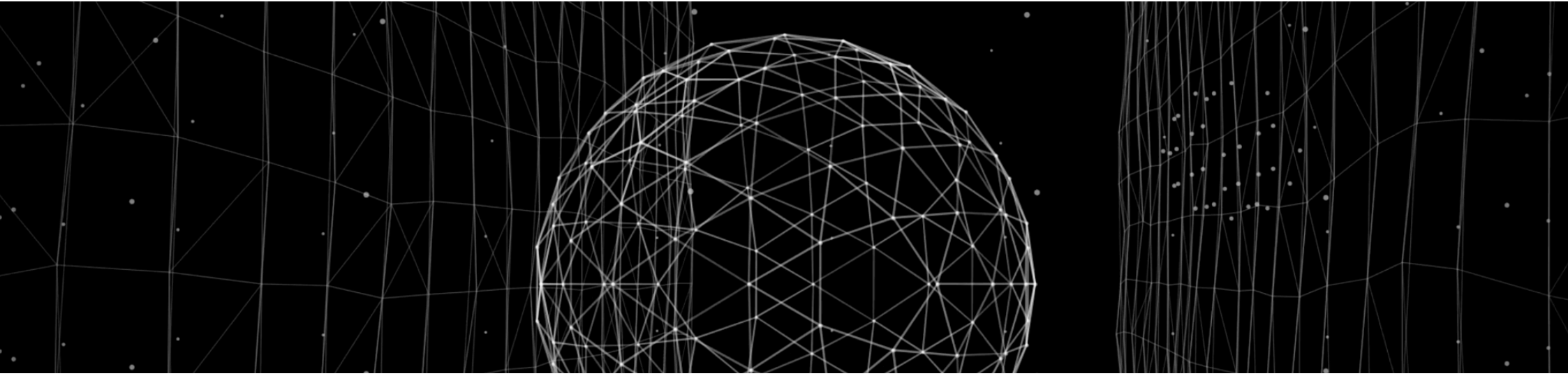
AR Foundation serves as a powerful and flexible framework within Unity, designed to simplify the development of augmented reality (AR) applications across both iOS and Android devices. This framework is crucial as it bridges the gap between Unity and the native AR capabilities of different platforms, like ARKit for iOS and ARCore for Android. Throughout this module, we will guide you through the essential steps of setting up AR Foundation in a new project or scene, ensuring you have a strong foundation for AR application development.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications

Install AR Foundation and set up AR Foundation Installing AR Foundation can be a time-consuming and technically challenging process. Understanding the process and specific requirements will allow students to implement AR Foundation in their own applications. **Assessment suggestion:** Have students download and set up the relevant packages to create an AR-ready Unity project template.	• Set up AR Foundation in a new project or scene for AR application development (CFW) • Install the AR Foundation package and all dependencies manually in a project (CFW) • Explain the role of AR Foundation in the development cycle of an AR application in Unity (CFW) • Download and explore AR Foundation samples (CFW) • Describe the relationship between AR Foundation and ARKit/ARCore	**Unity Learn** • Setting Up AR Foundation **Unity Manual** • Install AR Foundation	
Create Augmented Reality applications using AR Foundation	• Employ AR Foundation to implement custom marker detection • Employ AR Foundation to implement plane detection • Employ AR Foundation to implement face detection • Set up AR Foundation in a new project or scene for	**Unity Learn** • Configuring Plane Detection for AR Foundation **Unity Manual** • AR Plane Manager component • AR Tracked Image Manager component	

	AR application development (CFW) • Set up AR Foundation plane detection to find flat surfaces in your surroundings (CFW)	• AR Tracked Object Manager component • AR Face Manager component	



Working with Unity MARS (Complete)

Module introduction

This module will introduce you to Unity Mars, a powerful tool designed to simplify AR app development by overcoming challenges in authoring for dynamic physical environments and reducing development time. Unity Mars supports cross-platform experiences, including iOS, Android, and HoloLens, and enhances AR workflows with features like plain-language authoring, in-editor app testing, quick-start templates, and mobile device authoring. Additionally, the AR Companion app for Unity Mars further streamlines the process by capturing environment data, enabling object scanning (Mac required), and allowing light AR scene editing directly on the device. This module will give you the essential skills and knowledge to efficiently create immersive AR experiences using Unity Mars.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Create AR applications making use of the Unity MARS feature set	• Use the MARS Panel for creating MARS projects (CFW)	**Unity Learn** • Create and build a testing scene	

Unity MARS makes developing AR applications for world space usage faster and more streamlined. Understanding the functionality of the Unity MARS toolkit will allow students to make the most of it and create interesting AR experiences that will work as intended in different real-world scenarios. **Assessment suggestion:** Have students create a simple AR application that will adjust its placement parameters based on the immediate surroundings of the user.	• Build a MARS project for Android and ARCore (CFW) • Build a MARS project for iOS and ARKit (CFW) • Add actions in Unity MARS (CFW) • Adjust settings in Unity MARS to improve graphic quality (CFW) • Evaluate a scene in Unity MARS (CFW) • Set up marker conditions in Unity MARS (CFW)	• Configure and build your basic app • Explore the example game	
Explain the role and use case of Unity MARS in the AR development workflow Unity MARS is a Unity extension that adds new functionality to support augmented and mixed reality content creation. Understanding the role and features of Unity MARS will allow the student to decide when the	• Explain the role of Unity MARS in the AR workflow (CFW)	**Unity Learn** • What is Unity Mars	

use of MARS would be viable in their projects. **Assessment suggestion:** Have students research and discuss the use cases where a product like Unity MARS would make the AR development workflow more intuitive and streamlined.			
Install and set up Unity MARS Installing and setting up Unity MARS can be a time-consuming process. Understanding this process will allow students to set up their Unity MARS projects fast and correctly. **Assessment suggestion:** Have students download and install Unity MARS, and set up a project template ready for development.	• Install and set up Unity MARS (CFW)	**Unity Learn** • Set up Unity Mars • Explore the Unity Mars UI	

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** • Oculus 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 **Unity Manual** • SerializedObject 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 • UI Toolkit Forum	• 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 **Unity Manual** • SerializedObject data binding • Create user interfaces (UI)	• Associate: Programmer

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 Real-time 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.

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** • Animation Rigging **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 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	• Apply imported animation clips to rigged models in Unity • Configure a humanoid rig to share animations between characters • Trim animation clips to access specific keyframed sequences within them • Identify the purpose of a specified parameter of an animator controller • Describe the relationship between parameters and transitions	**Unity Manual** • Animation	• Associate: Game Developer

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.	• 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		
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; 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	• 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 • Record a GameObject animation using Record Mode • Set up a new Animation Clip • Move an animation in a scene using a parent GameObject	**Unity Learn** • Introduction to 3D animation systems • Rigging a Sprite with the 2D Animation Package • Unity for Animation: Road to Real-time Live Series • Get started with animation **Unity Manual** • Animation • Animation Rigging • Blend Trees **Unity Resources** • 2D game art, animation, and lighting for artists	• Associate: Artist • Professional 3D Artist

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.	• 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		
Create basic 2D animations with Sprites Parameters allow students to control a number of elements about an animation, such as its speed or state. **Assessment suggestion:** Have students create three animation states for an animation and three parameters that can be altered at runtime with a script.	• Recognize the process for automatically generating 2D animation from Sprite sheets (ProArt) • Identify methods within the animation controller used to customize animation states for Sprites (ProArt) • Identify editor animation parameters necessary to fine-tune sprite animations (ProArt)	**Unity Manual** • Introduction to 2D Animation • 2D Animation	• Professional 3D Artist

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** • Control animation with an Animator	• Professional 3D Artist • Associate: Game Developer
Set up a new NavMesh in a scene 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	• Explain backfilling with a NavMesh • Explain how maximum slope is used with a NavMesh • Explain how obstacle avoidance is used with a NavMesh	**Unity Learn** • Unity NavMesh • NavMesh Baking - Unity Learn • NavMesh Agents - Unity Learn • Working with NavMesh Agents - Unity Learn **Unity Manual**	

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.		• Building a NavMesh	
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 setting up an efficient workflow and technology stack. **Assessment suggestion:** Have students use Substance Designer or Substance Painter to create a workflow setup between	• 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) • Identify approaches to working with different parameter types for animation state transitions. (ProArt) • Modify individual animations for use within a state machine. (ProArt)	**Unity Learn** • Explore State Machines	• Associate: Artist • Professional 3D Artist • Associate: Game Developer

the Unity Editor and their Substance application and create materials for at least one of their assets in a scene.			

VFX in Unity

Module introduction

In this module, you will learn about Visual Effects (VFX) in Unity, focusing on the dynamic and versatile tools of particle systems and the VFX Graph. These powerful features are essential for creating stunning visual effects that can bring your Unity projects to life. You will learn how to use particle systems for simulating complex phenomena like fire, smoke, and water, as well as how to leverage the VFX Graph for more advanced, customizable effects.

When you are ready to take a deep dive into this topic, check out the free eBook, The Definitive Guide to Creating Advanced Visual Effects in Unity.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Build and customize a particle system	• Control a particle system with scripting	**Unity Learn**	

Particles are a low-overhead way to create simple but impressive visual effects. **Assessment suggestion:** Have students use particle effects to both recreate environmental phenomena and highlight player achievement.		• Introduction to Particle System	
Decide whether to use Unity's Particle Systems or VFX Graph in order to produce an effect in your scene Using Unity's Particle Systems or VFX Graph to produce an effect in your scene is to create visually compelling and dynamic special effects or simulations by harnessing the power of real-time particle-based visual effects. **Assessment Suggestion:** Challenge the learner to design and implement a custom visual effect, like a magical spell, weather simulation, or explosion, using either Unity's Particle	• Define the acronym VFX • Explain different applications of VFX in real-time 3D experiences, such as gameplay and environmental effects • Describe the impact that VFX can have on the level of polish in a project • Understand the differences between Unity's Particle System and VFX Graph in order to select the appropriate tool for a given use case	**Unity Learn** • Get started with VFX • Introduction to Particle System **Unity Manual** • Visual Effect Graph • Built-in Particle System • Particle System	• Associate: Artist • Associate: Game Developer

Systems or VFX Graph, and assess their proficiency in configuring particle behaviors, textures, shaders, and interactions to achieve the desired effect within the scene.			
Interpret a simple VFX graph Unity's VFX Graph is crucial for generating complex, high-quality visual effects in a scene by manipulating particle systems, shaders, and simulations to achieve immersive and realistic results. **Assessment Suggestion:** Encourage the learner to create a Unity project where they leverage the VFX Graph to craft an intricate visual effect	• Recognize whether a particle effect has been created using the Particle System or the VFX graph • Add a new VFX graph to the scene • Explain the role of each of the four default context nodes in a VFX Graph: Spawn, Initialize Particle, Update Particle, and Output Particle • Navigate in the VFX graph editor window by using the keyboard and mouse • Make simple edits to an existing VFX Graph, such as changing the emission rate or particle lifetime	**Unity Learn** • Get started with VFX	
Produce environmental and burst effects by configuring Unity's Particle System object	• Set up a new Particle System in the scene • Play, pause, stop, and restart a particle system	**Unity Learn** • Create a burst particle	• Professional 3D Artist

Producing environmental and burst effects by configuring Unity's Particle System object is to create dynamic and immersive visual elements within a game or application, such as fire, smoke, rain, or explosions, by controlling the behavior and appearance of particle emitters. **Assessment Suggestion:** Have the learners create a Unity scene where they configure a Particle System to generate a specific environmental or burst effect (e.g., rain, fireworks, or a campfire), and then assess their ability to adjust parameters like particle emission rate, size, color, and movement to achieve the desired visual result in real-time.	in Scene view by using the Particle Effect window ● Explain how individual Particle Systems can be combined to create more complex effects ● Locate the Main module of a Particle System ● Recall the three modules enabled by default in a Particle System: Emission, Shape, and Renderer ● Configure a Particle System's main properties, such as lifetime, size, and max particles by modifying the Main module ● Control the location and initial direction of particles by modifying the Shape module ● Control the rate and timing of particles by modifying the Emission module ● Control the appearance of individual particles by modifying the Renderer module		

	• Create a single burst of particles, rather than a continuous emission over time by using the Bursts section of the Emission module • Add randomness to a Particle System by using the Random Between Two Constants feature • Change the color of a particle over its lifetime by using the gradient editor and the Color Over Lifetime module • Change the size of a particle over its lifetime by using the curve editor in the Size Over Lifetime module		

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: 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	
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.	• 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	

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 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 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'	• 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 • Use and interpret common object types in visual scripts • Identify essential programming structures in order to comprehend a visual script	**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	
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.	• 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	**Unity Learn** • IF Statements • Loops • Switch Statements • Arrays • Enumerations • Implement data persistence between scenes	• Associate: Programmer • Associate: Game Developer

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.	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 • Modify the values of numeric variables by using mathematical operators	• Implement data persistence between sessions	
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	**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

	• 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		
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 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 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 • Developer's guide and references • Basic concepts in Visual Scripting **Unity Resources** • Visual scripting	

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	

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
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	
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	• 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 Manual** • Multiplay • Matchmaker • Vivox Unity SDK • Friends • Safe Voice **Unity Blog** • Master multiplayer	

building and maintaining servers and related online products. **Assessment suggestion:** Have students register their game with UGS and implement basic multiplayer functions.		**Other Resources** • Lobby • Relay • Vivox voice chatting • How to set up Matchmaker • VALORANT: A Unity case study	

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.

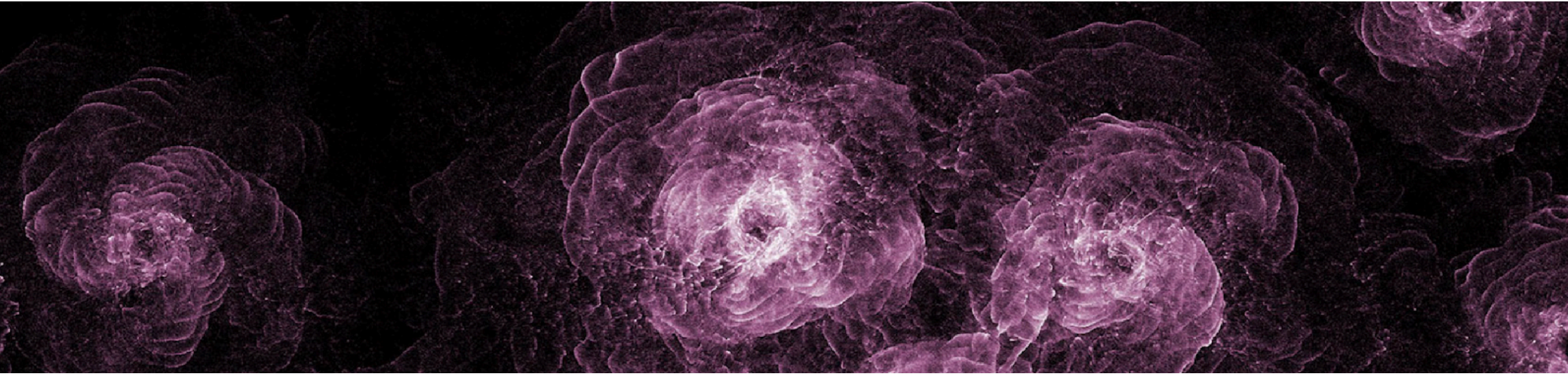
Suggested skills and learning objectives

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 various types of portfolios • Describe tools for building a portfolio • 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 In the industry, successful teams use various technologies within a design process to identify and solve problems by creating new,	• Explain how downloaded AssetBundles and content catalogs are cached • Advise clients with contextual information to make the technology more understandable to them	**Unity Learn** • Roles and careers for real-time creators • Career research and preparation • Develop your learning plan • Job preparation	

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.	• Solve problems to address client needs with efficiency and creativity		
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	• Explain the importance of defining purpose, goal, and audience	**Unity Learn** • Introduction to real-time 3D experience design	

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.	• Describe the structure and content of design documents • Explain the uses of a project charter • Organize project tasks based on production roles • Investigate appropriate applications for project management • Explain how a design document or project brief is used in a project	• Introduction to user feedback and testing • The real-time production cycle • Introduction to project management and teamwork	



Optimization and publishing

Module introduction

In this module, you will learn to balance aesthetics and performance in Unity by analyzing the impact of art assets and lighting. This includes understanding how polycount, particles, visual effects, and shadows affect performance. You'll create and deploy basic builds, implement Level of Detail (LOD) groups and objects to optimize scenes, and use mobile SDKs for testing and publishing applications. The module emphasizes optimizing application performance for smooth framerates, ensuring an immersive and responsive experience. You will also gain expertise in optimizing scene art assets and lighting for specific build targets, which is crucial for maintaining high performance without compromising visual quality.

Suggested skills and learning objectives

Skill and Description	Learning Objectives	Resources	Related Certifications
Analyze the impact of art assets and lighting on performance (polycount,	• Recognize the effects that Rigidbody and Collider components have on performance	**Unity Learn** • Optimization	

particles, visual effects, lighting, and shadows) Analyzing the impact on performance of factors such as polycount, 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 polycount, particles, visual effects, lighting, and shadows to optimize the scene's performance while maintaining an acceptable visual quality.	• 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		
Create and deploy a basic build of a project Unity provides project settings and analysis tools that allow your application to target different	• Add the required modules for a basic build to the current Unity install • Adjust Build Settings to create a basic build	**Unity Learn** • Publish your project **Unity Manual** • Publishing builds	• Associate: Programmer

hardware and software platforms. Knowing which templates to use and which packages to install will allow a student to create optimized applications. **Assessment suggestion:** Have students identify the target build platform and adjust the build settings to produce the optimized build outcome.	• Deploy a build as a Unity Learn submission		
Create LOD groups and objects to optimize a scene Levels of Detail are used to render objects at a distance as an optimizing tool. **Assessment suggestion:** Have students create and rig Levels of Detail on complex objects to optimize performance.	• Recognize processes for creating LOD groups • Determine adjustments needed to transition phases for LOD objects to satisfy design requirements • Recognize uses of different fade modes and their application to an LOD group • Interpret design requirements to identify needs for LOD groups	**Unity Learn** • Working with LODs **Unity Manual** • Level of Detail (LOD) for meshes	• Associate: Artist • Professional 3D Artist • Associate: Game Developer
Employ mobile SDKs to test and publish applications.	• Deploy a mobile application to an Android device	**Unity Learn** • Publish your project • Publish to Android	

Employing mobile SDKs to test and publish applications involves utilizing software development kits (SDKs) provided by mobile platform providers to build, test, and deploy mobile applications on specific platforms like iOS or Android. **Assessment Suggestion:** Require the learner to develop a mobile application (either for iOS or Android) using a chosen mobile SDK, and assess their ability to successfully build, test, and publish the app to an app store, ensuring it meets platform-specific requirements and guidelines.	• Deploy a mobile application to an iOS device • Build an application to WebGL or a personal computer	• Publish to iOS • Create and publish WebGL builds **Other Resources** • Publishing for Oculus • Publishing for MS HoloLens	
Optimize application performance to achieve smooth framerates in order to ensure an immersive and responsive experience Optimizing application performance to achieve smooth framerates in order to ensure an immersive and responsive experience involves fine-tuning	• Recall approximate frame rate performance targets for popular head-mounted displays • Select appropriate profiling tools to identify the sources of performance problems	**Unity Learn** • Optimization • Introduction/Optimization and Profiling • Optimizing your VR/AR Experiences • Lighting optimization with Precomputed Realtime GI	• Associate: Programmer

various aspects of a Unity project, such as rendering settings, asset management, and scripting, to maintain consistent and high frame rates during gameplay. **Assessment Suggestion:** Challenge the learner to take an existing Unity project with performance issues and task them with identifying and implementing optimizations across different areas (e.g., rendering, asset management, scripting) to achieve and maintain a stable and smooth frame rate while ensuring the game or application remains immersive and responsive			
Optimize lighting for performance Optimizing lighting for performance in Unity entails adjusting lighting settings, baking techniques, and shader choices to enhance rendering efficiency while preserving the desired visual quality.	• Lighting Optimization • Configure baked lighting to improve performance at runtime • Explain how to use light baking, reflection probes, and other light optimization techniques to increase performance	**Unity Learn** • Creative Core: Lighting • Lighting optimization with Precomputed Realtime GI • Working with Lighting in Unity	

Assessment Suggestion: Present the learner with a Unity project containing a scene with complex lighting setups, and ask them to demonstrate their ability to analyze and optimize the lighting elements to achieve improved performance without compromising the scene's overall visual appeal and realism.			
Optimize scene art assets for a particular build target The Unity Asset Store provides almost any conceivable assets you may want for your application, but in some cases creating your own assets may be the only option. Students who know how to use external Digital Content Creation (DCC) applications like Blender, Maya or 3ds Max can use their skills to create more advanced content for their own projects. **Assessment suggestion:** Have students create a product model	• Select appropriate profiling tools to identify the sources of performance problems • Decrease polycount of assets to optimize graphics	**Unity Learn** • Performance Requirements • Introduction/Optimization and Profiling • Working with the Stats Window • Working with the Frame Debugger **Unity Manual** • Working with the Frame Debugger • The Rendering Statistics window • Frame Debugger **Other Resources**	• Associate: Artist

in a 3rd party modeling application and optimize it for use in an Augmented reality application using the tools provided by		• Performance Optimization Tips Webinar	

AI in Unity - Sentis and Muse

Module introduction

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
Create AI-assisted assets and functionality in a Unity Project	• Generate game-ready textures using Unity Muse	**Unity Learn** • Install Muse Tools	

Unity Sentis enables you to embed an AI model in the Unity Runtime for your game or application, enhancing gameplay and other functionality directly on end-user platforms.	• Generate game-ready Sprites using Unity Muse • Generate usable code through conversation using Unity Muse • Generate ideas for your project with Unity Muse	• Get started with Muse Texture • Get started with Muse Sprite • Get started with Muse Chat • Create a game prototype with Muse • Get started with Unity's AI-powered Muse Tools **Unity Manual** • Sentis overview **Unity Blog** • Introducing Unity Muse **Other Resources** • Bring AI into your project with Unity Sentis • AI Unity Forum • Unity Muse • Introduction to Unity Muse	
Implement AI in your Unity project Unity Muse is an AI platform that accelerates the creation of real-time 3D applications and	• Implement AI Object identification in your game or application using Sentis • Identify handwritten numbers, letters, and	**Unity Manual** • Profile a Sentis model • Get started with Sentis • Create a model with Sentis	

experiences like video games and digital twins. The eventual goal of Muse is to enable you to create almost anything in the Unity Editor using natural input such as text prompts and sketches.	symbols for unique gameplay interactions with Sentis • Convert live speech to in-game text using a machine learning language model for natural language interactions between players • Upscale a low-resolution image or texture in your game to get to production quality, or optimize assets for specific devices with Sentis • Automate dialog and create meaningful interactions between players and NPCs with Sentis • Add visual effects to a scene with any theme with Sentis	• Run an imported model with Sentis **Unity Blog** • Introducing Unity Muse **Other Resources** • Introduction to Unity Muse • AI Unity Forum	