Unity

CERTIFIED
User
VR Developer

Exam Objectives

Unity Certified User VR Developer

The Unity Certified User VR Developer certification exam will test the candidate's ability to create VR experiences and programs within Unity software. The exam objectives are aligned with current industry standards set by professionals and educators.

Audience Description

Candidates for this certification have a foundational knowledge of the procedures used to develop VR experiences and programs within Unity software. Successful candidates will be able to design, create, and troubleshoot Unity programs using VR as a delivery medium. They should be able to demonstrate knowledge of writing, applying, and troubleshooting basic C# scripts. Candidates should be able to set up appropriate lighting and sound for a scene, optimize that scene to run smoothly, identify good practices to prevent player health issues, and conform to safety standards. They should understand the preplanning stages of creating a VR experience and the tools used to plan the creation of a VR project. Candidates should have at least 150 hours of instruction and/or hands-on experience creating VR projects in Unity.

To be successful on the test, the candidate is also expected to have the following prerequisite knowledge and skills:

- 8th grade reading skills
- Algebra I
- An understanding of how to use desktop computer software and hardware
- Digital literacy skills, including the ability to research, create content, and solve problems using technology
- Computational thinking skills, including the ability to decompose a problem into smaller parts and solve problems through automation

Although not required, the following skills will help a candidate learn about Unity VR Development more easily:

- Familiarity with C# programming language
- Familiarity with VR software and hardware, including tethered and stand-alone VR headsets

1. Basic Unity Concepts for VR Development

1.1. Define essential VR concepts, including but not limited to stereoscopic vision, how VR differs from other forms of XR, tracking methods, and VR

1.2. Demonstrate how to import or modify assets, including Prefabs

1.3. Given a scenario, identify how to use the Transform component to position, rotate, and scale an object in the scene

1.4. Identify the correct primary interface window to complete a given task while using the default workspace

1.5. Manage components in the Inspector Window

2. Building a Scene for VR

2.1. Given a scenario, identify common preplanning techniques, including design documents, flow charts, animatics, character model sheets, prototyping, greyboxing, storyboarding, concept art, and proportional level scaling

2.2. Identify correct methods to implement environment design with 3D objects using finalized assets

2.3. Identify various types of light and when to use Baked vs Realtime lighting

3. UX Implementation for VR

3.1. Identify the steps to create a basic UI using World Space for a VR scene, including the use of Canvas, Button, Image, Text, and the Event System

3.2. Given a scenario, determine the components needed for a user to physically manipulate objects, including but not limited to Colliders, the XR Grab Interactable, and Rigidbodies

3.3. Identify types of player locomotion, including degrees of freedom, moving an avatar, 3-axis motion, 6-axis rotation, and rotating along an axis

3.4. Given a scenario, identify optimal VR interactions regarding health and safety

3.5. Differentiate between attributes of audio sources, including but not limited to 2D and spatial audio

4. Scripting with Unity

4.1. Given a scenario, select the appropriate basic C# code to achieve a goal that requires knowledge of properties, variables, methods, basic data types, or binary operators

4.2. Given a scenario, select the appropriate Unity structure to achieve a goal that requires knowledge of data structures, such as Vector3, GameObject, Collider, Rigidbody, or AudioSource

4.3. Given a scenario, identify how to handle a collision or trigger Enter, Stay, or Exit event

5. Troubleshooting and Playtesting

5.1. Given a problem discovered in playtesting, identify areas to troubleshoot, including Static objects, missing Colliders, missing Rigidbodies, incorrect settings, Is Trigger, Is Kinematic, or Use Gravity

5.2. Identify types of logs in the Console

5.3. Identify errors output to the Console and the steps to fix them, including null references, missing end of line markers, or syntax errors

5.4. Identify correct optimization methods when working on a VR scene, including but not limited to camera occlusion culling, removing unused objects, or level of detail (LOD)