Exam Objectives

Unity Certified Associate: Programmer

Prerequisites

Demonstrate core skills and competencies across programming, UI, debugging and asset management to help you obtain your first professional programming role with Unity.

Prerequisite experience:

- 2-3 semesters of post-secondary Unity classwork or equivalent independent study
- Experience with a diverse range of Unity projects
- Importing assets or code, including from the Unity Asset Store or Unity Package Manager, and addressing conflicts that arise as a result
- Performing debugging of non-complex problems
- Interpreting pre-existing, well-documented code
- Integrating and modifying pre-existing well-documented code
- Building basic scene management, including loading scenes
- Creating, editing, and using Prefabs
- Deploying a basic build

Core Skills

(Certification exam topics)

1. Unity Programming

    1.1. Evaluate code for integration into an existing system created/architected by a lead

    1.2. Make decisions required to prototype new concepts

    1.3. Determine code that would accomplish a specified interaction or programming logic

    1.4. Decide how to implement scene management and transitions

    1.5. Apply basic data persistence within a runtime session

    1.6. Given a situation, determine proper usage and application of the Unity API

    1.7. Decide the appropriate properties, scripts, and components of GameObjects for required tasks

    1.8. Apply concepts required to write code with basic inheritance and interfaces

    1.9. Choose the appropriate data structures for a specific situation

    1.10. Choose the appropriate data types for a specific situation

    1.11. Identify the steps required to deploy a basic build

2. UI

    2.1. Apply concepts required to lay out a user interface

    2.2. Identify the process required to bind data on the UI to application data

2.3. Decide how to capture and respond to UI input using the Event System

2.4. Decide how to create the menu flow in an application state

3. Debugging

3.1. Troubleshoot code that fails to perform as expected

3.2. Troubleshoot common compilation bugs

3.3. Troubleshoot runtime exceptions

3.4. Determine techniques required to refactor and improve code

3.5. Determine techniques required to profile and debug trivial performance issues

4. Asset Management

4.1. Identify the process required to create a prefab from art and code

4.2. Identify properties of nested prefabs and prefab variants

4.3. Identify the primary purposes of version control when working with Unity