Exam Objectives

Unity Certified
Expert
Gameplay
Programmer

The Role

Gameplay Programming professionals focus on the execution of the game design. Gameplay Programmers bring the game to life by adding scripts to assets created by the art team. They are responsible for implementing the user interface (UI), environment, characters, and objects, as well as optimization and scripting necessary to execute the game mechanics. They also bring functionality to the non-player characters (NPCs) that populate the game.

Gameplay Programmers work side by side with designers (or they might be designers themselves) and reference the Game Design Document (GDD) to make the game a reality. These people are highly technical, and they understand what is required to bring ideas to life. This includes knowing when and how to use Unity components, writing their own components, configuring Prefabs, and properly organizing each scene. They do all of this with the Technical Design Document (TDD) in mind.

Job Titles for this Role

- Gameplay Programmer
- Game Scripter/Engineer
- Level Designer

Prerequisites

This expert certification is recommended for people who have spent several years in this field and have accrued a variety of advanced, practical application experience, such as:

- Experience in a video game development studio
- Experience designing immersive games, web/mobile games and digital games
- Good knowledge of scripting/programming using C#
- Experience in the full game lifecycle, working from early concept through launch
- Expert understanding of all aspects of game design (level, non-player characters [NPCs], game mechanics)
- Experience designing for and implementing Unity Services
- Understanding of game asset and animation pipelines, including character and environment setups
- Solid organization skills with adherence to file structures, naming conventions and other established protocols
- Ability to create and use a library of Prefabs to make the best game possible

Core Skills

The Expert Gameplay Programmer certification verifies that candidates have the skills necessary to effectively assemble and optimize a game. Successful candidates will have advanced proficiency in the following areas.

Prototyping

- Create and evaluate prototypes of core gameplay for rapid iteration
- Design and implement GameObjects to prove game mechanics' functionality
- Create a library of Prefabs based on the GDD, using assets generated by the art team
- Organize and set up the folder structure for a Prefab library

Level Design

- Design and create interactive-level prefabs using colliders and Rigidbody components
- Set the appropriate layer for physics masking on interactive prefabs
- Set up runtime-spawned Prefabs, and implement dynamic gameplay
- Set up and implement event triggers and scripted events using Colliders as triggers
- Create and script custom logic components for GameObjects to link state machines to interactions/triggers in scene
- Place and configure effects throughout the scene, including both static and runtime-generated placements
- Configure levels of detail (LOD) per platform specifications
- Evaluate GameObject placement and dependencies per platform specification
- Evaluate scene for streaming versus static scene-loading
- Construct game levels with multiple scenes
- Set up cinematics to enhance gameplay

Non-Player Character (NPC) Design

- Design and create NPC logic and artificial intelligence (AI) scripts
- Implement navigation and pathfinding for NPCs using NavMeshes, NavMeshAgents, NavMesh obstacles, and Off-Mesh links
- Set up NavMesh area types and costs
- Set up triggers to enable/disable NavMesh areas
- Implement NavMeshAgent avoidance and crowd simulation

- Evaluate NPC placement and dependencies per platform specification
- Set up frame-based audio and effects in animation clips

User Interface and Game Mechanic Design

- Use the Animation System for animator controllers, state machines and logic scripts for game mechanics
- Evaluate and optimize colliders, Rigidbody components and physics materials for interactive GameObjects
- Implement gameplay-related user interfaces, such as heads-up displays, mini-maps, radar systems, health bars, and other data-driven elements

Performance Optimizations and Target Platforms

- Implement AssetBundle download and placement in game levels
- Design and modify input and controller schemas for different platforms and/or virtual reality (VR)
- Analyze GameObjects and scenes for runtime and storage optimization per platform specification
- Optimize occlusion culling throughout scenes
- Debug and test game levels during runtime

Unity Services Implementation: Ads, In-App Purchases, and Analytics

- Implement simple and rewarded Unity Ads
- Implement Unity In-App Purchasing (IAP)
- Design Unity Analytics integration in the GDD and TDD
- Set up monitoring of custom data events using Analytics to monitor player behavior
- Analyze and evaluate existing levels and recommend changes based on Analytics data
- Use Unity Performance Reporting to modify and optimize game builds per platform

Certification Exam Topics

Section Description
Prototyping (Core Gameplay for Rapid Iteration)
• Core gameplay prototyping • Conflicts and solutions during prototype stage
Level Design Programming
• Physics configurations • Raycasting • Script-spawned prefabs during runtime • Level logic and behaviors • Population of levels with Particle Systems and effects • Platform optimizations • Scene loading and unloading • Methods to display cinematics
NPC Design Programming
• NPC logic and behavior • Navigation and pathfinding • NPC spawning and placement
User Interface Implementation
• UI coordinate systems and UI scripting
Performance Optimization and Target Platforms
• Rendering optimization • Asset Bundle downloading and configuration • Gameplay debugging • Platform differences and impact on gameplay
Unity Services Implementation
• Unity Ads • Unity In-App Purchasing • Unity Analytics • Unity Cloud Build

Sample Questions

Question 1

A Gameplay Programmer is creating a prototype in which the game is a side scrolling bullet shooter. The main character is a plane and the enemies are different types of UFOs. When the player destroys a UFO, an explosion effect is shown where the UFO was. When the player dies, a special player ship explosion effect is shown.

The plane can shoot up to 64 bullets on screen. The maximum number of UFOs of any type that can be shown on screen is 128. The maximum number of UFO bullets on screen is 1024. The maximum number of explosion effects on screen is 128.

The Gameplay Programmer needs to determine which GameObjects should be placed in the editor and which should be spawned at runtime (i.e., from an object pool).

How should this be accomplished?

a. The player, UFOs, player explosion effect, and UFO explosion effect should be placed in the editor. The player's bullets and UFO's bullets should be spawned at runtime.

b. The player, player bullets, UFOs, and UFO bullets should be placed in the editor. The player explosion effect and UFO's explosion effect should be spawned at runtime.

c. The player and the UFOs should be placed in the editor with the player bullets, UFO bullets, player explosion effect , and UFO explosion effect spawned at runtime.

d. The player and player explosion effect should be placed in the editor. The UFOs, UFO bullets, player bullets, and UFO explosion effect should be spawned at runtime.

Question 2

The Game Design Document (GDD) is a 3rd person open world game. The GDD specifies two types of gameplay for the main player:

1. Walking around the world on foot
2. Driving around in a motorcycle

The relative height of each mode is similar, but when the player is on the motorcycle, they can go much faster than on foot.

The Gameplay Programmer determines several areas of the game must be tweaked during the switch between walking and riding:

- Camera Field of View (FOV)
- Level of Details (LOD) distances
- Level streaming

When the player is on the motorcycle, which strategy should the Gameplay Programmer use to address each area for this GDD?

a. The Camera FOV must be a smaller value.
   The level must be streamed much faster.
b. The Camera FOV must be a larger value.
   The level must be streamed much slower.
c. The LOD distances must be a larger value.
   The level must be streamed much slower.
d. The Camera FOV must be a larger value.
   The level must be streamed much faster.

Question 3

The GDD is set in a large city where the player can walk around and interact with anyone in the city. The player can then complete various random quests for the citizens. The player can only have 5 active quests at any time. If the player walked from one end of the city to the other, it would take roughly 4 hours.

The game contains a top down mini-map in the Heads-Up Display (HUD) which shows a 2D icon for each of the following:

- the player's location
- all active quests
- all the various citizens walking around the city

The mini-map can be zoomed in and out. When it is fully zoomed in, it shows only the player's icon. When it is fully zoomed out, 25% of the entire city can be seen. There is slowdown that occurs only when the player zooms in fully on the mini-map.

What is a possible cause of the slowdown?

- There are too many unload calls for all the citizen icons' textures from memory.
- The active quest icons are querying too often to get how many are currently active.
- The mini-map's culling algorithm is taking too long to cull out most of the city.
- The HUD is too complicated and is being recreated every frame.

Question 4

The GDD is a mobile game running on a tight memory budget. The game has sections of the game loaded in AssetBundles to ensure smooth performance. Each section is independent of another section, so AssetBundles can be loaded and unloaded without affecting another section.

However, sometimes there are pink textures that appear.

What could be the cause of this issue?

- AssetBundle.mainAsset is broken.
- AssetBundle.Unload() is being called too early.
- AssetBundle.LoadAllAssets() is being called with a wrong Type.
- AssetBundle.LoadAssetWithSubAssets() is being called with a wrong string and Type.

Question 5

The GDD is a mobile match 3D puzzle game with 300+ levels. The game contains the following Analytics custom events for each level:

1. “levelStarted”: Triggered when a player starts the level
2. “levelCompleted”: Triggered when a player completes the level

The team would like to determine which are the top 5 most difficult levels in the game. The levels are time limited.

What additional custom event would be needed to determine the top 5?

- Add a “levelTime” to send the amount of time spent in the level.
- Add a “levelRestarted” to be triggered when the mobile app is closed by the operating system.
- Add a “levelFailed” to be triggered when a player fails or quits the level early.
- Add a “levelResumed” to be triggered when the mobile app resumes from background process.

Correct Answers: C, D, C, B, C