



Unity Game Development Course: C# Scripting, 2D and 3D Gameplay and Prototyping



Unity Game Development Course: C# Scripting, 2D and 3D Gameplay and Prototyping

Introduction:

Unity game development with C# scripting, 2D and 3D gameplay and prototyping is a 5-day course for programmers, game designers and digital media teams that ends with a playable game prototype and its build package. Organisations that commission serious games, training simulations or interactive media often stall because staff can script or draw but cannot turn a design into a working Unity build. Nominees already write code or produce digital assets at work and spend each day in hands-on labs inside the Unity Editor. CoreConcept Training Center delivers this Unity game development course.

Course Objectives:

- Translate a game concept into a game design document with a core loop, rules and player controls that a Unity team can build from
- Structure Unity scenes, GameObjects and prefabs and write C# components that respond correctly to the MonoBehaviour lifecycle
- Apply Rigidbody physics, colliders and trigger callbacks to create reliable movement, hits and pickups
- Build 2D and 3D mechanics with Tilemaps, Cinemachine cameras, UI Canvas menus, Animator Controllers and mixed audio
- Program NavMesh-driven enemy behaviour, test gameplay code and profile frame time, then produce PC and mobile builds
- Assemble a playable game prototype with its build package and release notes from a scoped feature backlog

Target Audience:

- Software developers responsible for interactive applications who are moving into game and simulation work
 - Game and level designers responsible for turning mechanics and rules into playable builds
 - Digital media and multimedia production staff responsible for interactive content, animation and audio assets
 - E-learning and simulation developers responsible for serious games and training scenarios
 - Technical artists responsible for importing, animating and optimising assets inside a game engine
-

Course Outline:

Day 1: Game Design Foundations, the Game Loop and the Unity Editor

- Game Design Document With Core Loop, Goals and Player Verbs
- Game Loop Cycle of Input, Update, Physics and Rendering
- Unity Hub Project Setup With Render Pipeline Template Choice
- Unity Editor Scene, Hierarchy, Inspector and Project Window Navigation
- GameObjects, Transforms and Prefabs for Reusable Level Pieces

Day 2: C# Scripting Architecture, Components and Physics

- MonoBehaviour Lifecycle With Awake, Start, Update and LateUpdate
- C# Classes, Inheritance and Interfaces for Gameplay Components
- ScriptableObject Data Assets for Weapons, Items and Enemy Stats
- Rigidbody, Colliders and FixedUpdate for Stable Physics Movement
- OnCollisionEnter and OnTriggerEnter Callbacks for Hits and Pickups

Day 3: 2D and 3D Game Mechanics, Interface, Animation and Audio

- Tilemap and Sprite Workflow for 2D Platformer Levels
- Character Controller and Cinemachine Cameras for 3D Movement
- Unity UI Canvas for HUD, Health Bars and Menus
- Mecanim Animator Controller States, Transitions and Blend Trees
- Audio Source, Audio Mixer and Spatial Sound Effects Setup

Day 4: Enemy AI, Testing, Performance Profiling and Platform Builds

- NavMesh Baking and NavMeshAgent Pathfinding for Enemy Units
- Finite State Machine for Patrol, Chase and Attack Behaviour
- Unity Test Framework Edit Mode and Play Mode Tests
- Unity Profiler CPU, GPU and Memory Module Bottleneck Analysis
- Build Profiles for Windows, Android and iOS Player Targets

Day 5: Lab: Building a Playable Game Prototype

- Prototype Brief, Core Loop and Feature Backlog Scoping
 - Lab Build of Player Controller, Physics and Scene Prefabs
 - Lab Integration of Enemy State Machine, HUD and Audio
 - Playtest Session With Defect Log and Profiler Fix Pass
 - Playable Game Prototype Build Package and Release Notes Completion
-

Skills You Will Gain:

- Game Loop Design
- Unity Scene Composition
- C# Gameplay Scripting
- Physics and Collision Tuning
- Character Animation Setup
- Enemy Behaviour Programming
- Frame-Time Profiling
- Multi-Platform Build Packaging

Why Attend This Course:

- Deliver a playable game prototype with its build package to the product owner or creative lead who decides whether the concept moves into full production
- Decide whether a concept fits 2D or 3D, which physics approach suits it and which target platforms to build first
- Avoid frame-rate drops, broken collisions and failed PC or mobile builds that delay releases and add rework cost
- Share prefabs, C# script templates, the enemy state machine pattern and the build checklist with colleagues starting game or simulation projects

Conclusion:

Back at work, the participant hands the product owner or creative lead a playable game prototype, its build package and release notes, which the unit uses to decide whether a game, serious game or interactive simulation concept moves into full production and on which platforms. Developers and designers reuse the prefabs, C# component templates and enemy state machine on the next project. After the first playtest round with real users, the team should review frame time on target devices, crash reports, defect counts and player feedback against the design document.

Frequently Asked Questions FAQ:

What should participants know before the Unity game development course?

Participants should already write code in an object-oriented language or produce digital assets such as sprites, models or audio. Basic C# helps, but the labs start from the Unity component model. Familiarity with version control makes sharing lab projects easier.

How does Unity game development differ from a business mobile app or a gaming business management course?

Unity game development focuses on real-time interactive software: the game loop, physics, animation, audio, enemy behaviour and frame-time performance. Mobile app courses build form-based business apps, and gaming business courses cover tournaments and sponsorship; both overlap with this course only at build publishing.

Why does the game loop matter in Unity game development?

The game loop runs input, physics, gameplay logic and rendering every frame, so whether code sits in Update, FixedUpdate or LateUpdate decides if movement is smooth and collisions are reliable. Understanding it prevents jitter, missed hits and wasted processing on target devices.

What do participants take back from the Unity game development course?

Participants take back a playable game prototype with its Unity project, build package and release notes, plus a design document, reusable prefabs, C# component templates, an enemy state machine and a profiling checklist for the next game or simulation project.
