Game Programming with Unity: Week 1

Welcome to your first week of game programming! This comprehensive guide will walk you through everything you need to know about getting started with Unity, one of the most popular game development platforms in the world. Whether you're dreaming of creating the next hit mobile game or an immersive 3D experience, Unity provides the tools to bring your ideas to life. Over the next two lectures, we'll cover the complete installation process, help you avoid common pitfalls, and introduce you to the essential features that make Unity such a powerful engine for both beginners and professionals.

What is Unity?

Definition & Purpose

Unity is a cross-platform game engine developed by Unity Technologies. It's a comprehensive software environment that allows developers to create both 2D and 3D games, simulations, and interactive experiences. Think of Unity as your digital workshop—it provides all the tools, materials, and workspace you need to build games without starting from scratch.

Originally launched in 2005, Unity has grown to become one of the most widely-used game engines globally, powering everything from indie mobile games to AAA console titles. What makes Unity special is its accessibility: it uses C# programming language, has an intuitive visual editor, and offers a massive library of ready-made assets and tools through the Unity Asset Store.

Why Choose Unity?

- **Beginner-Friendly:** Visual scripting options and extensive documentation make learning manageable
- **Cross-Platform:** Build once, deploy to 25+ platforms including PC, mobile, consoles, and VR
- **Free to Start:** Personal edition is completely free for students and hobbyists
- **Massive Community:** Millions of developers share tutorials, assets, and solutions
- **Industry Standard:** Used by major studios and indie developers alike
- **Asset Store:** Access thousands of pre-made assets, scripts, and tools

Complete Unity Installation Guide

01

Download Unity Hub

Visit unity.com and navigate to the "Get Started" section. Download Unity Hub—this is the central management application for all Unity versions and projects. Unity Hub is approximately 150MB and works on Windows 10/11, macOS 10.13+, and Linux.

02

Install Unity Hub

Run the installer and follow the on-screen prompts. Accept the license agreement and choose your installation directory. We recommend using the default location (C:\Program Files\Unity Hub on Windows). The installation typically takes 2-3 minutes.

03

Create Unity Account

Launch Unity Hub and create a free Unity ID account. You'll need this to activate your license. Use a valid email address—you'll receive a verification link. Student accounts qualify for additional benefits through Unity Student Plan.

04

Activate License

In Unity Hub, go to Preferences → Licenses → Add License. Select "Get a free personal license" if you're a student or hobbyist. This personal license is fully functional with no limitations for learning and small projects.

05

Install Unity Editor

Click "Installs" in Unity Hub, then "Install Editor". Choose the latest LTS (Long Term Support) version—currently Unity 2022.3 LTS is recommended for beginners. This ensures stability and long-term support.

06

Select Modules

Choose your build targets: start with your operating system (Windows/Mac/Linux) plus Android and WebGL. Documentation is essential—always include it. Visual Studio Community is recommended as your code editor. Total installation size: 6-12GB depending on modules.

Installation Time: The complete Unity installation typically takes 20-45 minutes depending on your internet speed and selected modules. Make sure you have at least 15GB of free disk space before starting.

System Requirements & Pre-Installation Checklist

Minimum Requirements

- **OS:** Windows 10 64-bit, macOS 10.13+, or Ubuntu 20.04+
- **CPU:** X64 architecture with SSE2 instruction set support
- **RAM:** 8GB minimum (16GB strongly recommended)
- **GPU:** Graphics card with DX10 (shader model 4.0) capabilities
- **Disk Space:** 15GB minimum for editor and modules
- **Internet:** Required for downloading and license activation

Before You Install

1. Update your graphics drivers to the latest version
2. Ensure Windows/macOS is fully updated
3. Disable antivirus temporarily (can interfere with installation)
4. Close all other applications to free up system resources
5. Have administrator privileges on your computer

Recommended Setup

- **OS:** Windows 11 or macOS Ventura (latest versions)
- **CPU:** Intel i5/i7 or AMD Ryzen 5/7 (4+ cores)
- **RAM:** 16GB or more for smooth performance
- **GPU:** NVIDIA GTX 1060 / AMD RX 580 or better
- **SSD:** Solid-state drive for faster loading times
- **Monitor:** 1920x1080 resolution minimum (dual monitors helpful)

Pro Tip: If you're on a laptop, plug in to power during installation and development. Unity can be resource-intensive and will drain battery quickly.

Common Installation Mistakes & Solutions

Mistake #1: Wrong Unity Version

Problem: Installing the latest beta or alpha version instead of LTS (Long Term Support). Beta versions have bugs and incomplete features that can frustrate beginners.

Solution: Always choose an LTS version (marked with a crown icon). As of 2024, Unity 2022.3 LTS is the recommended choice. LTS versions receive updates and support for 2+ years.

How to Fix: In Unity Hub → Installs → Add, filter by "LTS" and select the newest LTS version available.

Mistake #2: Skipping Required Modules

Problem: Not installing necessary build support modules during initial setup. Students often skip Android Build Support or Documentation, then can't build for mobile or find help resources.

Solution: During installation, check: Documentation, your platform (Windows/Mac/Linux), Android Build Support, WebGL Build Support, and Visual Studio or VS Code.

How to Fix: In Unity Hub → Installs → Click gear icon next to your Unity version → Add Modules → Select missing modules → Install.

Mistake #3: Insufficient Disk Space

Problem: Starting installation without checking available space. Unity requires 15-20GB, and installation fails halfway, corrupting files and wasting hours.

Solution: Before installing, verify you have at least 25GB free space. Use a disk cleanup tool if needed, or install to a different drive with more space.

How to Fix: If installation fails, completely uninstall Unity Hub and all Unity versions, free up space, restart computer, then reinstall fresh.

Mistake #4: Installing Without Visual Studio

Problem: Unity needs a code editor to write scripts. Many beginners uncheck Visual Studio during installation, then can't edit or create C# scripts properly.

Solution: Always install Visual Studio Community (free) or VS Code with the Unity module selected. This integrates seamlessly with Unity for debugging and IntelliSense code completion.

How to Fix: Download Visual Studio Community separately from visualstudio.microsoft.com, install with Unity development workload, then set as default in Unity → Edit → Preferences → External Tools.

Mistake #5: License Activation Errors

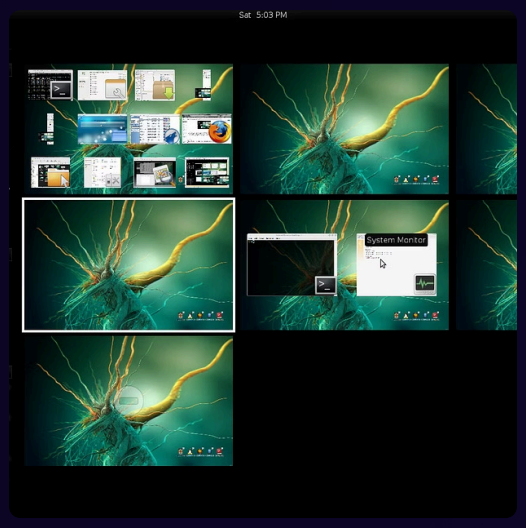
Problem: Not activating Unity license properly, resulting in "No valid Unity license" errors. This happens when skipping license steps or using offline installations without proper offline activation.

Solution: Immediately after installing Unity Hub, go to Preferences → Licenses → Manage Licenses → Add → Get a free personal license. Complete the online activation while connected to internet.

How to Fix: Sign out of Unity account in Hub, restart computer, sign back in, then re-request personal license. Ensure firewall isn't blocking Unity's license server connection.

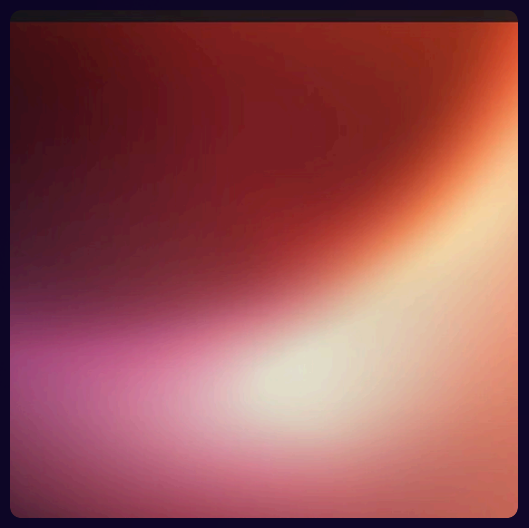
Unity Interface: Essential Components

Once Unity is installed, you'll encounter a workspace divided into several key areas. Understanding these panels is crucial—they're your primary tools for building games. Let's break down each component and what it does:



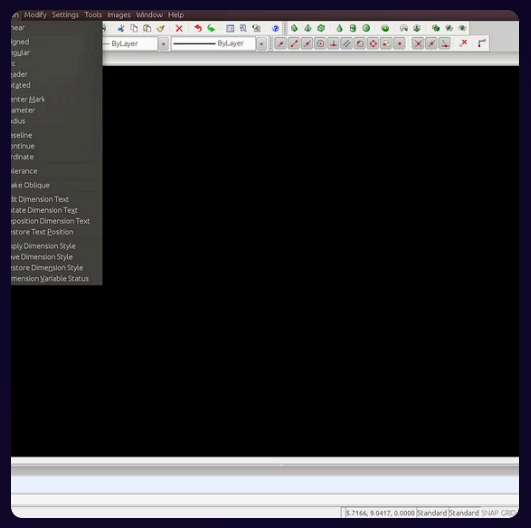
Scene View

Your visual workspace where you build and arrange game levels. Use mouse controls to navigate: middle-click to pan, right-click to rotate, scroll to zoom. This is where you'll spend most of your time positioning objects, adjusting lighting, and designing your game world.



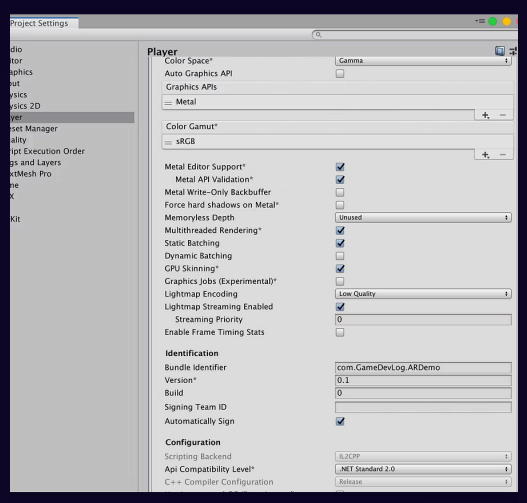
Game View

Shows exactly what players will see when they run your game. Click the Play button to test your game in real-time. Any changes made during Play mode are temporary—a common beginner mistake is editing during play and losing changes!



Hierarchy Panel

Lists all GameObjects in your current scene. Think of it as your scene's table of contents. Parent-child relationships are shown through indentation—children inherit transformations from their parents, a powerful organizational feature.

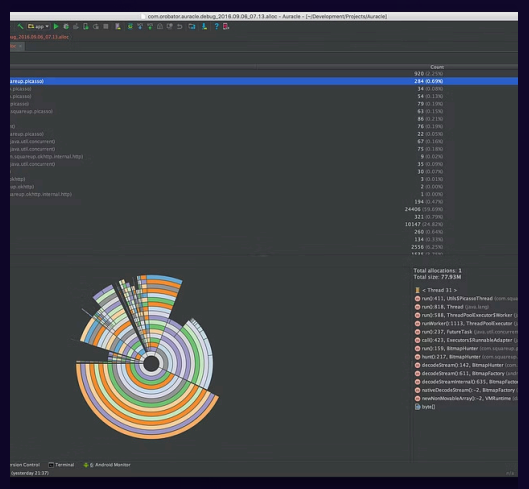


Inspector Panel

Displays detailed properties and components of selected GameObjects. This is where you'll adjust positions, attach scripts, modify materials, and fine-tune every aspect of objects. Every GameObject's components appear here for editing.

Project Window

Your asset library showing all files in your project: scripts, textures, models, audio, scenes, and more. Organized like a file explorer, this is where you import new assets and manage your project's resources. Keep it organized from day one!



Console Window

Displays errors, warnings, and debug messages from your code. Learn to love the Console—it's your best friend for troubleshooting. Red messages are errors (must fix), yellow are warnings (should fix), white are info logs.

Essential Unity Concepts

GameObjects: The Building Blocks

In Unity, everything in your scene is a GameObject. A character, a camera, a light source, even an empty organizational container—all are GameObjects. Think of GameObjects as empty containers that hold Components, which give them functionality.

Creating GameObjects: Right-click in Hierarchy → Create → choose object type (3D Object, 2D Object, Light, etc.). Or use the top menu: GameObject → Create.

Components: Adding Functionality

Components are modular pieces of functionality you attach to GameObjects. Every GameObject has at least one component: the Transform component (controls position, rotation, and scale). Want your GameObject to have physics? Add a Rigidbody component. Need it to play sound? Add an Audio Source component.

Adding Components: Select a GameObject → Inspector panel → Add Component button → search for the component you need.

Scripts: Custom Behavior

Scripts are custom components you write in C# to create unique game behaviors. They define how objects interact, respond to player input, manage game rules, and more. Scripts are the "brains" of your game, while components are the "body parts."

The Component Model

```
// GameObject Structure
GameObject: Player
 ├── Transform (required)
 │    ├── Position: (0, 1, 0)
 │    ├── Rotation: (0, 0, 0)
 │    └── Scale: (1, 1, 1)
 ├── Mesh Renderer
 ├── Box Collider
 ├── Rigidbody
 └── PlayerController Script
```

Golden Rule: GameObjects are containers. Components are the actual functionality. Scripts are custom components that you create.

Your First Script Example

Let's create a simple script that rotates an object. This demonstrates the basic structure of a Unity C# script:

```
using UnityEngine;

// This script rotates a GameObject continuously
public class SimpleRotation : MonoBehaviour
{
    // Speed of rotation in degrees per second
    public float rotationSpeed = 50f;

    // Update is called once per frame
    void Update()
    {
        // Rotate around Y-axis over time
        transform.Rotate(0, rotationSpeed * Time.deltaTime, 0);
    }
}
```

How to use this script: Create this script in Project window (Right-click → Create → C# Script), name it "SimpleRotation", then drag it onto any GameObject in your scene. Press Play, and watch the object spin! The `Time.deltaTime` ensures smooth rotation regardless of frame rate.

Scenes & Project Organization

Scenes: Your Game Levels

A Scene is a container for your game content—think of it as a level, menu screen, or any distinct part of your game. Every Unity project starts with one scene (SampleScene), but you'll create many: MainMenu, Level1, Level2, GameOver, etc.

Creating Scenes: Project window → Right-click → Create → Scene. Double-click to open it. Save your scene regularly (Ctrl+S / Cmd+S)!

Folder Structure Best Practices

Organizing your project from the start prevents chaos later. Create folders in your Project window to categorize assets. A typical structure: Scenes, Scripts, Prefabs, Materials, Textures, Audio, Models, Animations.

Why It Matters: In a month, you'll have hundreds of files. Without organization, finding anything becomes impossible. Professional developers are meticulous about project structure.

Prefabs: Reusable Objects

A Prefab is a template GameObject you can reuse throughout your project. Create an enemy once as a Prefab, then place 100 copies in different scenes. Update the original Prefab, and all instances update automatically—massive time saver!

Creating Prefabs: Drag a GameObject from Hierarchy into Project window. The GameObject turns blue, indicating it's now a Prefab instance.

Recommended Folder Structure Example

```
Assets/
├── Scenes/
│   ├── MainMenu.unity
│   ├── Level01.unity
│   └── Level02.unity
├── Scripts/
│   ├── Player/
│   ├── Enemies/
│   └── UI/
├── Prefabs/
│   ├── Characters/
│   └── Environment/
├── Materials/
├── Textures/
├── Audio/
│   ├── Music/
│   └── SFX/
└── Models/
```

Pro Organization Tips:

- Use descriptive names: "PlayerController" not "Script1"
- Group related items in subfolders
- Keep imported assets in their own folder
- Delete unused assets regularly to save space
- Use consistent naming conventions (PascalCase for scripts)
- Create a "_Documentation" folder for notes and references

Basic Unity Workflow: Creating Your First Game Object

Let's walk through a practical example that demonstrates the complete workflow from creating an object to adding behavior. This is the foundation you'll use for every game you make.

Create the GameObject

Hierarchy → Right-click → 3D Object → Cube. A cube appears at position (0,0,0) in your scene. This is now a GameObject named "Cube" with a Transform, Mesh Renderer, and Box Collider component.

Position It

With Cube selected, look at Inspector → Transform. Change Position Y to 2 (raises it 2 units above ground). Use Scene view's move tool (W key) to drag it visually, or type exact values in Inspector.

Add Material/Color

Project window → Right-click → Create → Material. Name it "RedMaterial". In Inspector, click the white box next to Albedo, choose red. Drag this material onto your Cube. It turns red!

Add Physics

Select Cube → Inspector → Add Component → Rigidbody. This gives your cube gravity and physics. Press Play, and watch it fall! The Rigidbody component makes objects respond to physics forces.

Create Ground

Create another cube (Hierarchy → 3D Object → Cube), name it "Ground". In Inspector, set Scale X=10, Y=0.5, Z=10 to make it a flat platform. Position Y=0. Press Play—your first cube now lands on the ground!

Adding Script Behavior

Now let's make the cube jump when you press Space. This introduces input handling and physics manipulation:

```
using UnityEngine;

public class JumpingCube : MonoBehaviour
{
    // Reference to Rigidbody component
    private Rigidbody rb;

    // Force applied when jumping
    public float jumpForce = 300f;

    // Run once at start
    void Start()
    {
        // Get the Rigidbody component attached to this GameObject
        rb = GetComponent();
    }

    // Run every frame
    void Update()
    {
        // Check if Space key is pressed
        if (Input.GetKeyDown(KeyCode.Space))
        {
            // Apply upward force to make cube jump
            rb.AddForce(Vector3.up * jumpForce);
        }
    }
}
```

To use: Create this script (Project → Right-click → Create → C# Script → name it "JumpingCube"), drag it onto your Cube GameObject. Press Play, then hit Space to make it jump! Adjust the jumpForce value in Inspector to change jump height.

Understanding the Code: `Start()` runs once when the game begins. `Update()` runs every frame (typically 60+ times per second). `GetComponent` finds other components on the same GameObject. `Input.GetKeyDown` detects key presses. `Vector3.up` means direction (0,1,0) pointing upward.

Next Steps: Your Game Development Journey

What You've Learned

Congratulations! You've completed Week 1 of game programming. You now know how to:

- Install Unity and Unity Hub properly
- Avoid common installation mistakes
- Navigate the Unity interface confidently
- Understand GameObjects, Components, and Scripts
- Create your first interactive object with physics
- Organize projects professionally
- Write and attach basic C# scripts

Practice Exercises

1. **Customize the Jumping Cube:** Change its color, size, and jump force. Add multiple cubes with different properties.
2. **Create a Simple Scene:** Build a small environment with a ground plane, walls, and several objects. Practice positioning and scaling.
3. **Modify the Rotation Script:** Make an object rotate on different axes. Try speeding up or slowing down the rotation.
4. **Experiment with Prefabs:** Create a prefab from one of your cubes and place multiple instances in your scene.

Essential Resources

Official Unity Resources:

- Unity Learn (learn.unity.com) - Free official tutorials
- Unity Documentation - Comprehensive reference guide
- Unity Forums - Community help and discussions
- Unity Asset Store - Free and paid assets

Recommended Learning Path:

1. Complete Unity's "Create with Code" course (beginner-friendly)
2. Follow along with the "Roll-a-Ball" tutorial (classic starter project)
3. Explore the "Ruby's Adventure" 2D game tutorial
4. Join Unity's student community Discord servers
5. Practice daily, even if just 30 minutes

Common Beginner Struggles

Don't worry if you feel overwhelmed—every game developer started exactly where you are. Common challenges include understanding 3D space, learning C# syntax, and debugging errors. The key is consistent practice and asking for help when stuck.

30

Days to Basic Competency

With daily practice, most beginners feel comfortable with Unity fundamentals within a month

100+

Free Learning Hours

Unity Learn offers over 100 hours of free, high-quality tutorials for all skill levels

1.5M

Active Developers

Join a global community of 1.5+ million Unity developers ready to help you succeed

Week 2 Preview: Next week, we'll dive into C# programming fundamentals for Unity, explore the Physics system in depth, create player movement controls, and build your first playable prototype. Come prepared with questions from this week's practice exercises!