Welcome to Unity Game Development: Week 2 Kickoff

Welcome, future game developers! This week, we're diving straight into the practical world of game programming using the Unity platform. Our focus will be on setting up your environment, understanding the core tools, and starting our first major project: a simplified Solar System simulation. Get ready to turn abstract concepts into interactive 3D realities!

> This lecture is your essential guide to navigating the Unity engine, creating your first project, and importing the necessary assets to begin practical development.

Made with GAMMA

Understanding the Game Engine: Unity Overview

A game engine is a powerful software framework that provides a comprehensive set of tools and technologies for game developers. It handles complex, technical tasks so you can focus on creativity and gameplay design.

Rendering Engine

Handles all the graphics: drawing 3D objects, lighting, shadows, and camera views. Unity makes beautiful visuals accessible even to beginners.

Physics Engine

Simulates real-world physics like gravity, collisions, and forces, which are crucial for realistic movement and interactions in your game world.

Scripting & Logic

The core of game programming, where you write code (using C# in Unity) to define rules, behaviors, and gameplay mechanics. This is where the magic happens!

Asset Management

Provides tools to manage all the pieces of your game—images, sounds, 3D models, and scripts—keeping them organized and optimized.

Made with GAMMA

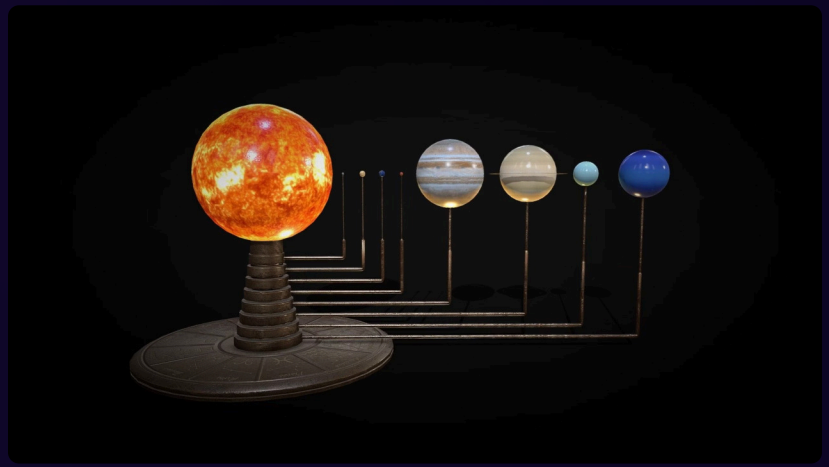
Project Focus: The Solar System Simulation

To start strong, we'll build a simplified Solar System. This project is perfect for learning fundamental Unity concepts like scene setup, object manipulation, and rotational scripting.

Why the Solar System?

- It teaches **coordinate systems and transformations** (position, rotation, scale).
- It's an excellent introduction to **parent-child relationships** (planets orbiting the sun).
- It requires practical application of **C# scripting** to manage continuous motion.
- It helps you understand the concept of a **game loop** (continuous updates).

By the end, you'll have a functional 3D simulation, laying the groundwork for more complex games!



Made with GAMMA

Step 1: Download and Install the Unity Platform

Before we code, you need the right tools. Unity operates through the Unity Hub, which manages different versions of the editor and all your projects.

Install Unity Hub

Go to the Unity website, download, and install the **Unity Hub**. This is the central application for managing all your Unity installations and projects.

Choose an Editor Version

Inside Unity Hub, go to the "Installs" tab and click "Add." Select the recommended **LTS (Long-Term Support)** version of Unity. LTS versions are stable and reliable for learning.

Add Modules

Make sure to select the **Windows Build Support** or **Mac Build Support** module, and critically, the **Visual Studio Community** module (or another IDE) for C# scripting.

Always install a reliable code editor like Visual Studio during this step. Without it, you cannot easily write and debug your C# scripts!

Made with GAMMA

Step 2: Creating Your First Unity Project

With the Unity Editor installed via the Hub, let's set up the project environment for our Solar System. The template you choose determines the default settings and rendering pipeline.

Project Creation Steps

- In Unity Hub, navigate to the **Projects** tab and click the New Project button.
- Select the **3D (Core)** or **3D URP (Universal Render Pipeline)** template. URP is more modern and efficient for most projects.
- Give your project a descriptive name, like "SolarSystem_Project," and choose a location on your computer.
- Click **Create Project**. The editor will open, and you're ready to start building!

Choosing the right template is crucial. For visual quality and modern features, we recommend URP.

Made with GAMMA

Step 3: Acquiring and Importing Project Assets

In professional game development, you rarely build everything from scratch. For our Solar System, we need 3D models and materials for the Sun and planets. We'll use pre-made assets to save time and focus on the programming logic.

Download Project Assets

Assets often come in compressed packages (like .unitypackage files). You can find these on the Unity Asset Store or provided by your instructor. **For this project, download the "SimplePlanets.unitypackage" file.**

Obtain Package

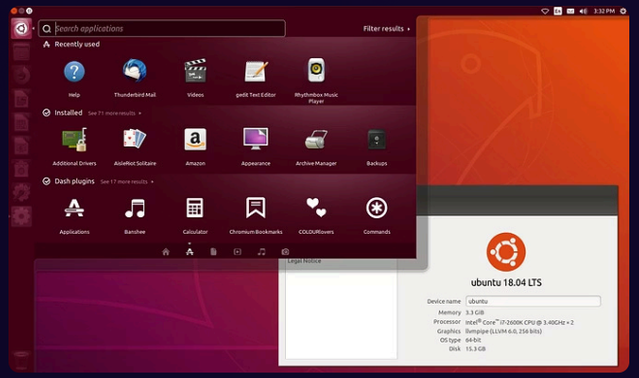
Download the required asset package (.unitypackage) from the specified source.

Import into Project

In Unity, navigate to **Assets > Import Package > Custom Package...** and select the downloaded file.

Confirm Selection

A window will pop up showing all the included items. Ensure all files are checked and click **Import**.



Made with GAMMA

Unity Editor Instructions: Essential Interface Guide

The Unity Editor is where you spend most of your time. Understanding its main windows is the key to efficient development.

Hierarchy Window

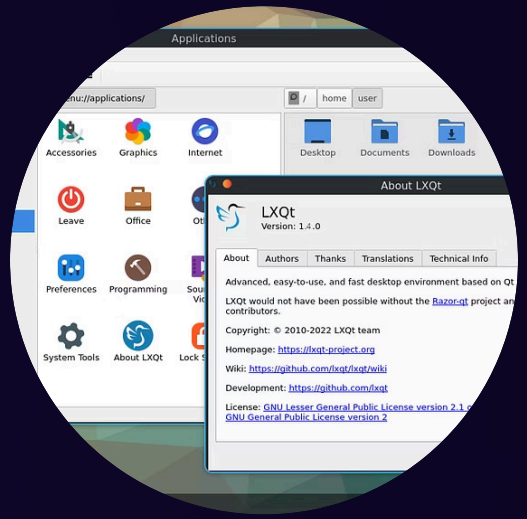
Lists every single object (GameObject) currently in your scene. Use it to organize objects into hierarchies, like placing planets as children of the Sun.

Scene View

Your 3D workspace. This is where you visually arrange objects, navigate the world, and build your levels. Use the right mouse button to look around and the WASD keys to move.

Inspector Window

Displays all the details (Components and properties) of the currently selected GameObject. This is where you modify position, rotation, scale, and attached scripts.



Project Window

Shows all the files (Assets) that belong to your project, organized like a file browser. This includes models, textures, sounds, and your C# scripts.

Made with GAMMA

Detailed Learning Material: Understanding GameObjects and Components

In Unity, everything in your scene is a **GameObject**. Every GameObject is simply a container that holds a list of **Components**. This Component-based architecture is fundamental to Unity programming.

The GameObject (The Thing)

The fundamental object in Unity. It has a name and, at minimum, one essential component: the **Transform** component.

The Component (The Behavior)

These are the functional pieces added to a GameObject that define what it is and how it behaves. Examples include the **Mesh Renderer** (to make it visible) and **Rigidbody** (to apply physics).

The Transform

Every GameObject must have this component. It stores the object's **Position** (where it is), **Rotation** (which way it's facing), and **Scale** (how big it is) in the 3D world.

For our Solar System, the Sun will be a GameObject with a Transform, a Mesh Renderer (to show the sphere model), and a custom C# script component (to make it glow and spin).

Made with GAMMA

Code Example: Simple Rotation Script

Scripting in C# allows us to define the specific logic for our game objects. For the Solar System, the planets need to rotate on their axis and revolve around the Sun. We will create a script called **Rotator.cs** and attach it to our planet GameObjects.

```
// Rotator.cs
using UnityEngine;

public class Rotator : MonoBehaviour
{
    // Public variable allows us to set the speed
    // in the Inspector window!
    public float rotationSpeed = 50f;

    // Update is called once per frame
    void Update()
    {
        // Use the Transform component to rotate the object
        // Time.deltaTime ensures the rotation is frame-rate independent
        transform.Rotate(0, rotationSpeed * Time.deltaTime, 0);
    }
}
```

How the Code Works (Easy Language)

- The `Update()` function runs constantly, every single frame the game displays.
- `public float rotationSpeed` creates a box in the Inspector where you can easily type in how fast you want the planet to spin.
- `transform.Rotate()` is the command that tells the object to spin.
- `Time.deltaTime` is like a stopwatch measuring the time since the last frame. We multiply the speed by this to make sure the speed is the same on a fast computer as it is on a slow one (consistency!)

Attach this script to any planet, and adjust the **Rotation Speed** value in the Inspector to control its axial spin!

Made with GAMMA

Next Steps and Maintaining Integrity

You've successfully set up your environment, imported assets, and learned the core structure of GameObjects and Components. The next week will focus entirely on applying these concepts to complete the Solar System project.

Consistency and Correlation

Remember that the principles we learned (GameObjects, Components, Transform) must correlate across all your work. If your planet isn't moving, check its Transform component and its attached C# script in the Inspector. Unity is all about interconnectivity.

Integrity and Learning

Do not use vague or irrelevant concepts. Stick to clear, practical steps. Your learning path is step-by-step: setup first, then scripting, then polishing. Every concept builds on the last.

Action Item: Practice Navigation

Spend time mastering the Scene View camera controls. Being able to quickly move around your 3D world is the single most important skill for a new Unity developer.

Made with GAMMA