Welcome to Game Design and Development

Sebastian Grygorczuk - STEM Institute at CCNY

This Week's Agenda

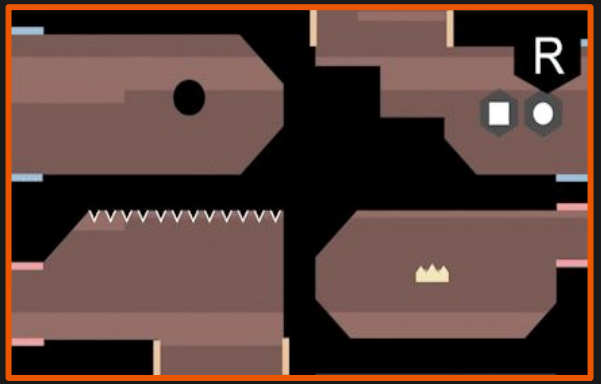
Getting Prepared for the Semester

- Introduce yourselves!
- Discuss the Video Game Industry and roles of Game Developers.
- Create the necessary accounts and downloading all the necessary software for the class projects.
- Learn how to copy, edit, and upload project files to GitHub.

Who are we?

My name is Sebastian Grygorczuk! I've been developing games for over 3 years, currently I'm working on a game called Claw Arena with a company called Mocha Chili.

My interests outside of gaming are robotics, hiking and history.



How About You?

What's your name?

Are you taking another class with the program? If so, what is it?

What's your favorite game and why?

What is your level of experience with programming/game design?

Avatar Mixer by Kenny

What Is The Goal Of This Class?

The purpose of this course is to provide students knowledge about the game development industry and the many careers that comprise it. Students will also learn the skills necessary for translating ideas into playable games, while preparing them for further study in the fields of engineering and design.

The Project

We will create Pixel Quest a 2D platformer game which will allow us to study the different aspects of Game Development inside of Unity.

GitHub Repository

We will be using a GitHub repository to store all of the lessons so that you can alway look back and read through them.

We will go over what a GitHub repository is later in the lesson but for now know that it's pretty much a Google Drive folder.

Each of the school computers will have the link to the website bookmarked, and if you want to look at it at home you can follow the QR code or web url provided to you on the paper.

https://github.com/Sgrygorczuk/WHEELS-Fall-2023-Semester

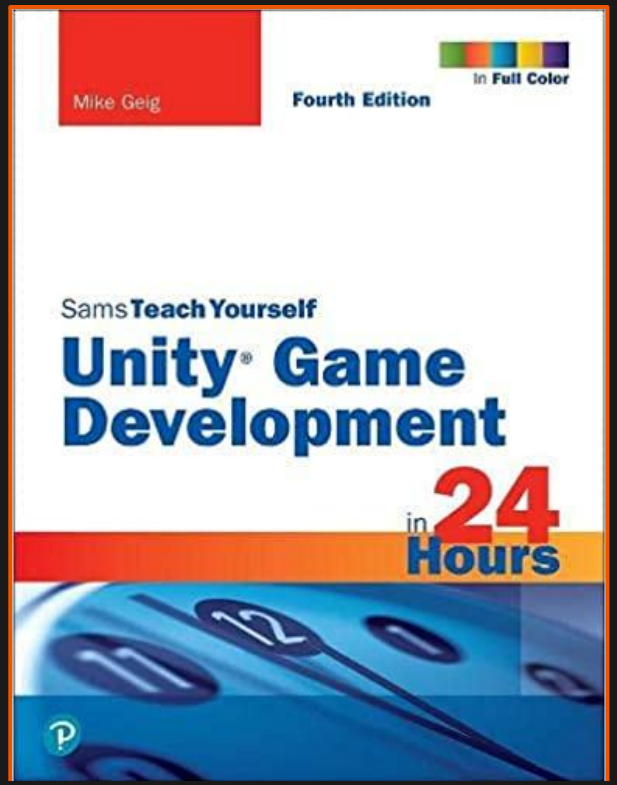
Computer Requirements

As this is a game development class the number one requirement is a modern computer capable of running the Unity and Visual Studio software. Windows, Mac, and Linux operating systems are compatible with the Unity Game Engine, however, Windows and Mac are highly recommended due to the limitations of Visual Studio on Linux platform.

An additional requirement for the class is a computer mouse. It should be brought to class to make game development easier.

Minimum requirements	Windows	macOS	Linux (Support in Preview)
Operating system version	Windows 7 (SP1+) and Windows 10, 64-bit versions only	High Sierra 10.13+	Ubuntu 16.04, Ubuntu 18.04, and CentOS 7
CPU	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support	X64 architecture with SSE2 instruction set support
Graphics API	DX10, DX11, and DX12-	Metal-capable Intel and	OpenGL 3.2+ or Vulkan-capable, Nvidia and AMD GPUs.

Class Resources



Class Textbook:

- *Unity Game Development* in 24 Hours 4th Edition by Mike Geig

Additional Core Materials:

- Unity User Manual
- Unity YouTube Channel
- Stack OverFlow

YouTube Resources:

- Brackeys
- Code Monkey
- Blackthornprod
- Tarodev

Unity Game Development in 24 Hours 4th Edition is an ideal textbook for this class because it's an introduction to Unity and many of its designer focused tools with programming included, but not programmer focused.

Additional Core Materials are official Unity resources that will explain how many of the systems technically work, but will not show expansive examples of how to utilize them.

Stack Overflow will help you with any programming bugs you may run into.

YouTube Resources are Independent Game Developers putting their craft online and showing how to use the tools Unity provides us.

Class Road Map

Introduction to Unity and Game Design

Introduction to the class and the software

2D

We will learn how to create 2D environments and game objects.

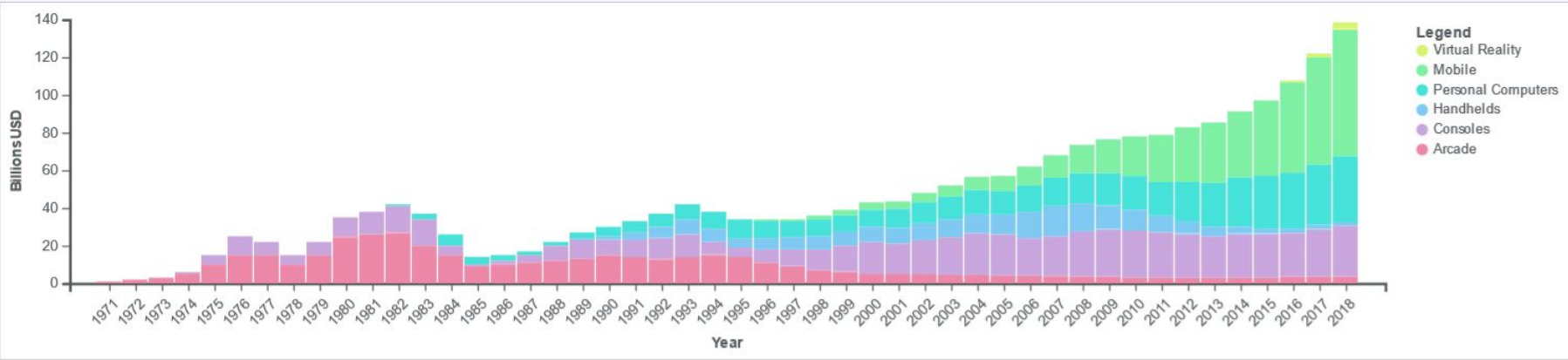
UI

We'll create the Main Menu and UI for the game.

What is Game Development

"Game Development is the art of creating games and describes the design, development and release of a game. It may involve concept generation, design, build, test and release. While you create a game, it is important to think about the game mechanics, rewards, player engagement and level design."[Source]

Video Game Industry



Global revenue estimates of the video game industry from 1971 to 2018, not adjusted for inflation, according to market research firm Pelham Smithers.[43]
The chart shows the impacts of the 1977 crash, the golden age of arcade games (1978–1983), the video game crash of 1983, the console revival (late 1980s), and the rise of mobile gaming since 2008.

The Video Game Industry is multi billion machine rivaling any all of the older media such as movies, music, and books.

There's an immeasurable number of Publisher and Game Development Studio that release new games daily and it's only getting bigger, if you really want to get a job in it there will be a place somewhere for you.

Additional Resources: Chart

Sebastian Grygorczuk - STEM Institute at CCNY

Stages of Game Development

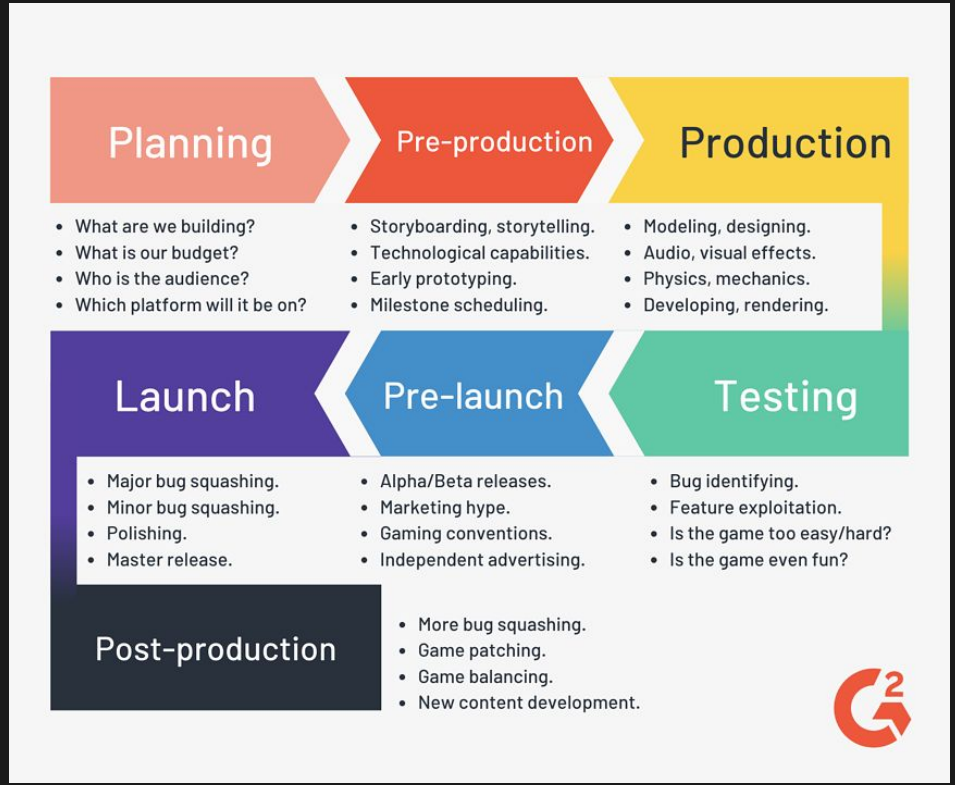
As we enter Week 4 and onwards of this class we will be following these stages of Game Development. We will mostly focus on the first four but we will talk about how to show off your game once it's done.

[1] It is ideal to know what you want the game to accomplish, the theme of the game, the gameplay style, who it's for.

[2] After this step, the ideas are built out. On many occasions, the idea on paper sounds great but sometimes doesn't work out once you start developing the ideas. You don't actually program any game in this part; rather you write out a Game Design Document that includes how everything in the game should behave and play out.

[3] Production starts once you've settled on parts that were agreed on as good and move forwards to crafting a game out of them. Things agreed on in pre production may change once developed but the game should adhere to the Game Design Document.

Additional Resource: **The 7 Stages of Game Development**



Sebastian Grygorczuk - STEM Institute at CCNY

What is a Game Developer?

Game Developer, unlike most Software Developers, doesn't specifically refers to a programmer.

Since games consist of many assets such as code, art, animations, sound and more, a game developers is someone who fulfills the role of developing assets for the game.

This includes: Game Designers, Programmers, 2D and 3D Artists, Animators, Sound Engineers, and Testers. Each of these disciplines can be broken down even furthers, and throughout this class you will wear the hats of each of these roles.

Depending on the size of the team creating the game each person may fulfill only one of these roles or a combination of all of them. Be ready to wear a lot of hats.

Additional Resource: The Big List of: Video Game Development Team

Game Designer

Game Designer is the creative lead of the project, coming up with the infrastructure of how and why the game will work.

If the project become large enough the Game Designer might be broken down into two roles, the Level Designer and the System Designer.

Level Designers take all of the assets created by the Programmers, Artists and Sound Engineers and put them together into levels for the player to play.

System Designers work on the larger scope of the game creating systems, such as enemy behavior, enemy, questing.

Game Designers tend to be the most recognizable names of any Game Development team such as **Shigeru Miyamoto**

Examples of Game Designers

Shigeru Miyamoto is a legendary designer who created many of the beloved franchisees today such as the original *Zelda*, *Donkey Kong*, and *Super Mario Bros* and helped bring back the game industry from the grave. He is now the General Manager of Nintendo.

Sid Meier is the creator of the *Civilization* game series. This series translated the board game appeal onto computer screens. Now he is the Director of Creative Development at Firaxis Games.

Hideo Kojima is the creator of the *Metal Gear Solid* series. This series popularized the Stealth genre. He is the CEO of Kojima Productions.

Peter Molyneux is the creator of the *Populus*, *Dungeon Keeper* and *Black & White* games. Which invented the genre of God Games and is the creator of the *Fable* RPG series. He is now the CEO of 22cans.

Game Programmer

Game Programmers make the game tick and provide a function to the assets created by the the artists. However, depending on the scope of the project the game the programmer could work on everything or a small system.

Programing roles can break down into many sub roles:

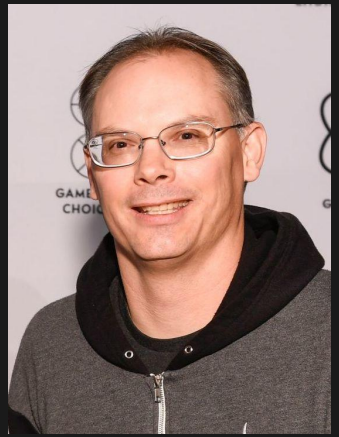
- Gameplay Programmer role
- Backend / Server / Online Programmer role
- AI Programmer role Graphics Programmer role
- UI Programmer role
- Animation Programmer role
- Physics Programmer role
- VFX Programmer role
- Audio Programmer role
- Tools Programmer role

Game Programmers tend to be less known than the flashy designers but you may still recognize a few names such as John Carmack.

Examples of Game Programmers

John Carmack is responsible for the current state of today's FPS genre. He programmed *Wolfenstein 3D*, *DOOM and Quake* which became foundations for modern FPS shooters like Call of Duty and Overwatch. Now is the Chief Technology Officer of Oculus VR.

Tim Schafer is a programmer that lead the point-and click-adventure games through their golden age. Working on *Secrets of Monkey Island* Franches and *Grim Fandango*. Now CEO of Double Fine.



Tim Sweeney is an engine developer who created the original Unreal engine. This helped push the FPS games from the 2D plane to full 3D. Now CEO of Epic Games.

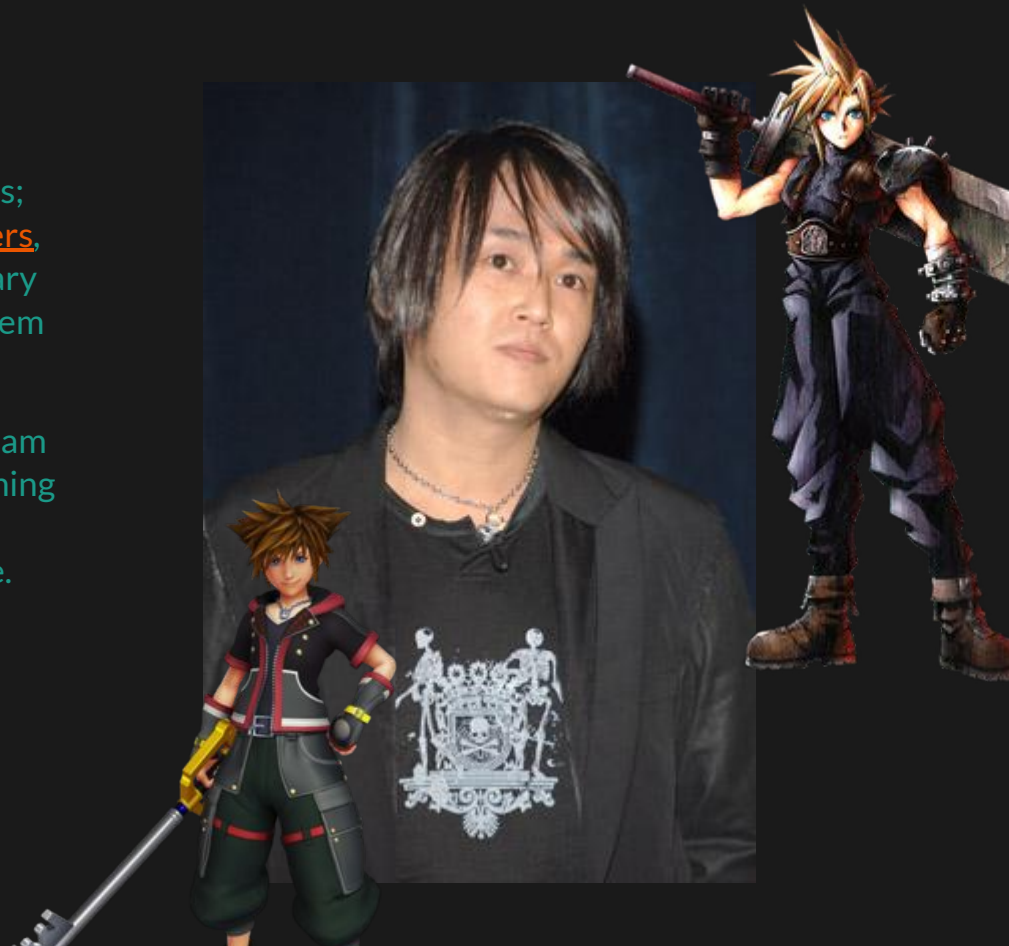
Yuji Naka is the programmer behind the original *Sonic* Trilogy. His work at SEGA allowed them to compete with Nintendo's Mario. He is the CEO of Prope.

Game Artist

Game Artist can be broken down into some many roles; concept artist, sprite artists, texture artist, 3D modelers, riggers, animators, lighting. Depending on the necessary fidelity of the game one person can work on a single item for years.

An example of that would be the cape in Batman Arkham Asylum where "there was one person working on nothing but the cape for two years, so there are over 700 animations and sound clips attached to the cape alone. That's why it looks so beautifully realistic." [Source]

Although it's the art the conveys the game to us game Artists are hardly recognized for their hard work. An example of a known artist in game industry would be Tetsuya Nomura.



Examples of Game Artists

Tetsuya Nomura was an artist on most of the modern *Final Fantasy* games with one of the most famous designs being Cloud and Sora from *Kingdom Hearts*. Now is a Game Director at Square Enix.

Josh Scherr was the Lead Cinematic Animator of the first three *Uncharted* games. These games created movie blockbuster experiences that were not really seen in games at the time and is hard to replace even today. Now is a write at Naughty Dog. [Example of Work]

Keijiro Inoue is a Visual Effects Artist behind games such as *Pikmin*, *Wii Sports*, *Super Mario Odyssey* and *Zelda Breath of the Wild*. Now is a Lead Effect Artist at Nintendo. [Example of Work]

Jamie McNulty was the Environment Artists on *BioShock*, *BioShock Infinite and Gear of War 4* and *5*. Now is working as a Environment Artist at Deviation Games. [Example of Work]

Sound Designers

Sound Designers, like artists, are under appreciated. No matter how engaging the game can be, without the proper Sound Effects to feedback to give the player actions or the music to build the tone of the gameplay the games isn't going to be enjoyable.

In addition to Music and Sound Effects, Sound Designers will work with Voice Actors.

Some sounds and songs are synonymous with the games they were made for. One of my personal favorite is Stormwind Theme by Jayson Hayes.

Examples of Sound Designers

Jason Hayes is a veteran composer creating pieces for *Starcraft*, *Warcraft* and *Diablo* and is continuing his work at Blizzard Entertainment. [Example of Work]

Kōji Kondō a veteran composer creati*ng Legend of Zelda* and *Mario* music currently in existence. [Example of Work] Now he countries to work at Nintendo as a Composer.

Masato Nakamura was the sound produce *Sonic the Hedgehog 1 and 2* soundtracks and is most likely responsible for the famous Sonic Ring Sound Effect. Is a bassist for the band Dream Come True but frequently works on Sound Design in games.

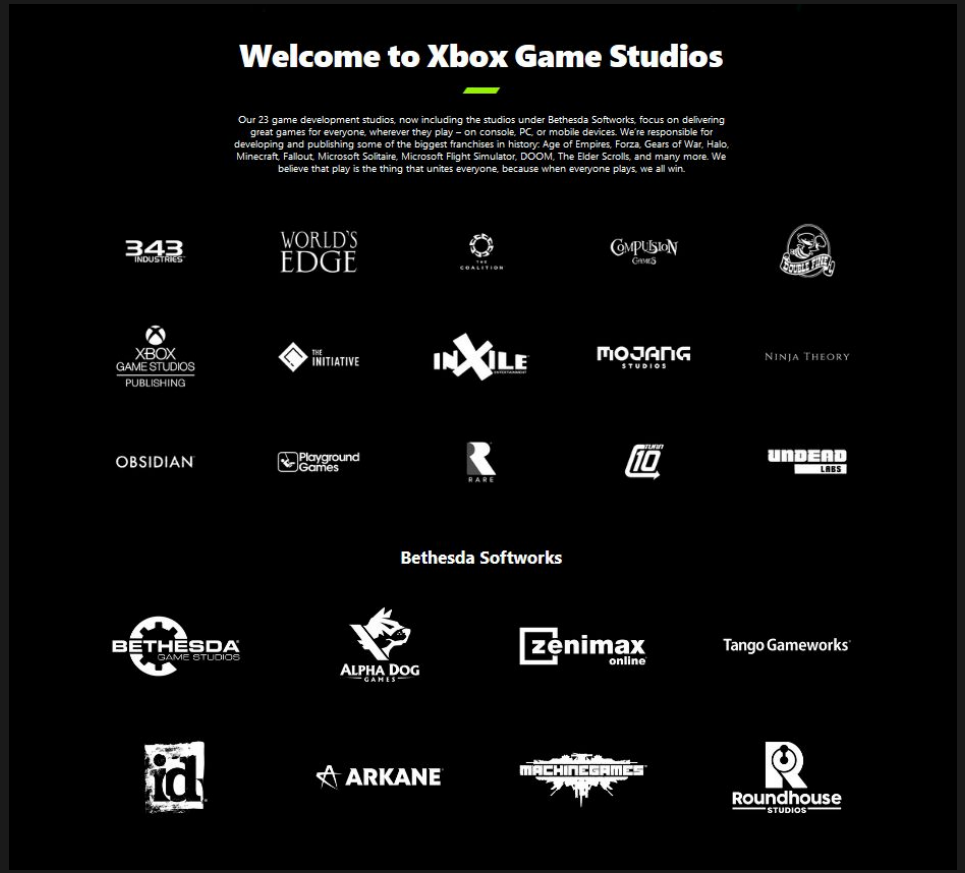
Phillip Kovats was the sound director for *God of War, The Last of Us, Uncharted* and many more Sony projects. He was responsible how the game sounds and mixes the voice acting. [Example of Work]. Now he is the Sr. Director of Sound at PlayStation Studios.

Publisher and Game Development Studio

Let's get a few more definitions and relationships out of the way:

Game Development Studio is the company who plans and produces the game. Not all Game Development Studios need to use Publishers but most will need the financial and commercial backing. The studio will compose of a lot of people not all of them directly connect to making of the game as some will be dealing with finance, advertising, legal, and more, but at the core there will be the Developers that make the game.

Publisher is the company who distributes the funds to the game development studio to create the game. Making games can be expensive, since it requires lots of people and money and as the game development process takes many years without a game selling. The Publisher front loads the costs of Game Development in agreement that they will get paid out for their investment. They can have a lot of sway in what the game becomes.



Indie Developers

Let's give honorable mentions to some inspiring Indie Game Developers that take all the talents of a Game Development Studio and perform them all by themselves.

Toby Fox, is a breakout star as a Game Designer, Programmer, Artist and Sound Design creating *Undertale* as a one man team.

Supergiant Games is a small team that has created hits such as *Bastion*, *Transistor* and *Hades*. Known for beautiful art, music and innovative approaches to storytelling.

Scott Cawthon is the creator of the massive franchise of *Five Nights at Freddy's* which is now a multimedia empire.

If you'd like to look into what Indie Game Development is like you can watch Developing Hell, a documentary of how Supergiant Games created their latest game *Hades*.

Indie Vs Big Studio

Almost all of the people we've mentioned worked in larger studios for big companies. However we do live in the age of the internet where any can sit down and create a game. If you want to be an Independent Game Developer or part of a larger team at a Studio will be up to your personal preferences.

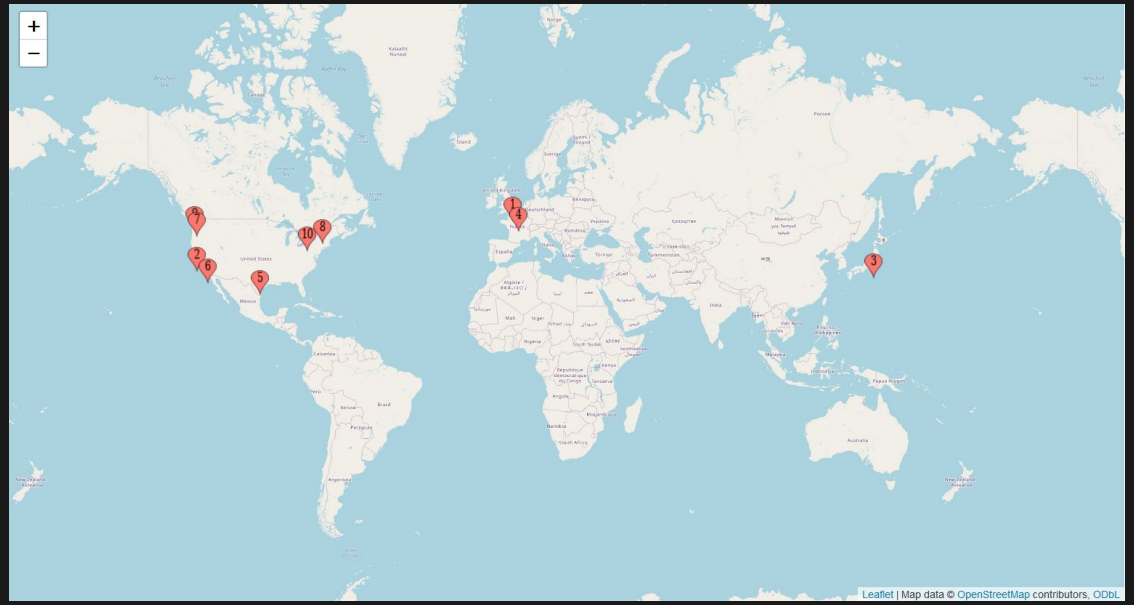
Do want to work in a large team or have personal freedom? Or do you want move to one of the few locations with a game development studio? What are you finances looking like as an independent developer? And many more questions like that.

Where Game Studios Exist

Game Development Studios are abound in many place however, New York City is not one of them.

If you'd like work at a large studio you will have to move to one of these cities. [Sources] Remote work is opening up as a possibility but on location is the preference for larger companies.

1. London
2. San Francisco
3. Tokyo
4. Paris
5. Austin
6. Los Angeles
7. Seattle
8. Montreal
9. Vancouver
10. Toronto

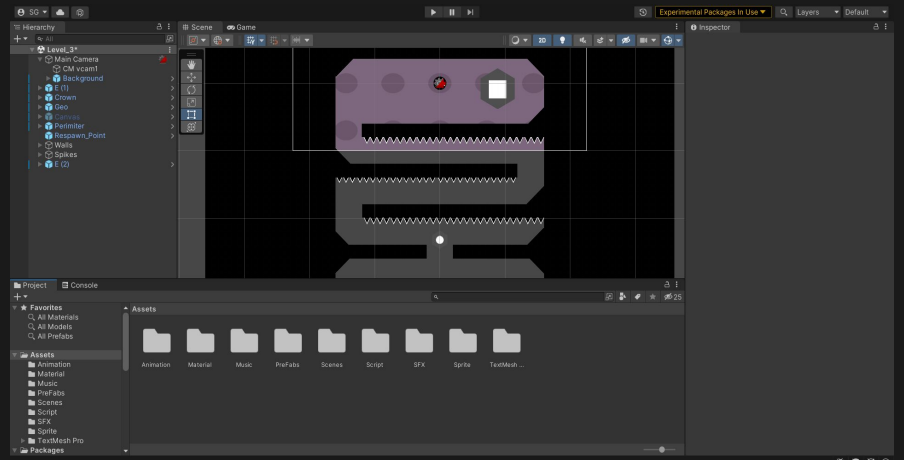


Unity

What is a Game Engine?

- It's a software that already has built in library and features that help you create games.

Things like physics, image processing, sound are already set in place and you have to make the connections between the features to create the game rather than having to program everything from scratch.



Sebastian Grygorczuk - STEM Institute at CCNY

Perks of Unity

- Port it to any platform; Unity allows you to create executable files for your game to pretty much any platform that's currently available. This includes: Windows, Mac, Linux, Web Browser, Android, iOS, AR, VR, Playstation, Xbox, and Switch consoles.
- Is free software for personal use as long as your projects don't exceed $100,000 revenu in a year.
- Is connected to the Unity Asset Store which host a treasure trove of free and purchasable assets.

Games Created With Unity

Unity is a multi-tool that allows whoever wields it to create anything:

Of course primarily we think of games when it comes to Unity and there are plenty of great examples such as “Cuphead”, “Hearthstone”, “Rust”, “City Skylines”, and many more.

All of these games cover wildly different styles of gameplay and complexity.

If you’re interested in checking the comprehensive list of games made with Unity.

Animation Created With Unity

Unity holds a incredibly powerful animation system which we will discuss during our third week of classes where we will animate our own cutscene.

Beyond just in-game cutscene Unity can create full on animated shows and movies such as “Mr. Carton” by Michaël Bolufer. Furthermore allowing companies such as Disney to film in VR environments using Unity creating movies like “The Lion King [2019]”.

If you’re interested in checking out some other animations made with Unity. “Lion King” Article.

Non Game Related Software With Unity

With the Unity engine being so versatile other industries have taken it for their own uses.

Main two uses are for visualization and simulations, allowing engineerings, architects, medical professionals see what it is they are constructing or working on while simulating the physics of their prototype.

If you're interested in checking out software made for automobiles, engineering or aerospace using Unity.

Other Game Engines

Unreal: Industry Standard uses 2D/3D game engine using C++ is free to use. Witcher 4 is being developed in it.

GoDot: Develing into creating 2D and 3D Environments and Levels, allows you to Programming with C# and C++ and Visual Scripting. Free to Use.

Construct 3: Web based 2D game engine with Visual Scripting, you have to licence it for all tools.

There dozens of game engines all different from each other however, many use similar standardized tools so that once you are familiar in one you can transfer your skills to another easily.

Additional Resources: Most used Engines The Best Game Engines of 2021

Installing Unity

To access Unity you will first have to download the Unity Hub.

Unity Hub is center for you to manage your projects and versions of Unity. As time goes on Unity releases newer versions that hold new or improved features.

To start your download head over to https://unity.com/download#how-get-started.

If you scroll down a bit you will see a box that allows you to download Unity Hub for all the different platforms.

Unity Hub

Once you've gotten the Unity Hub installed you should be greeted with this window.

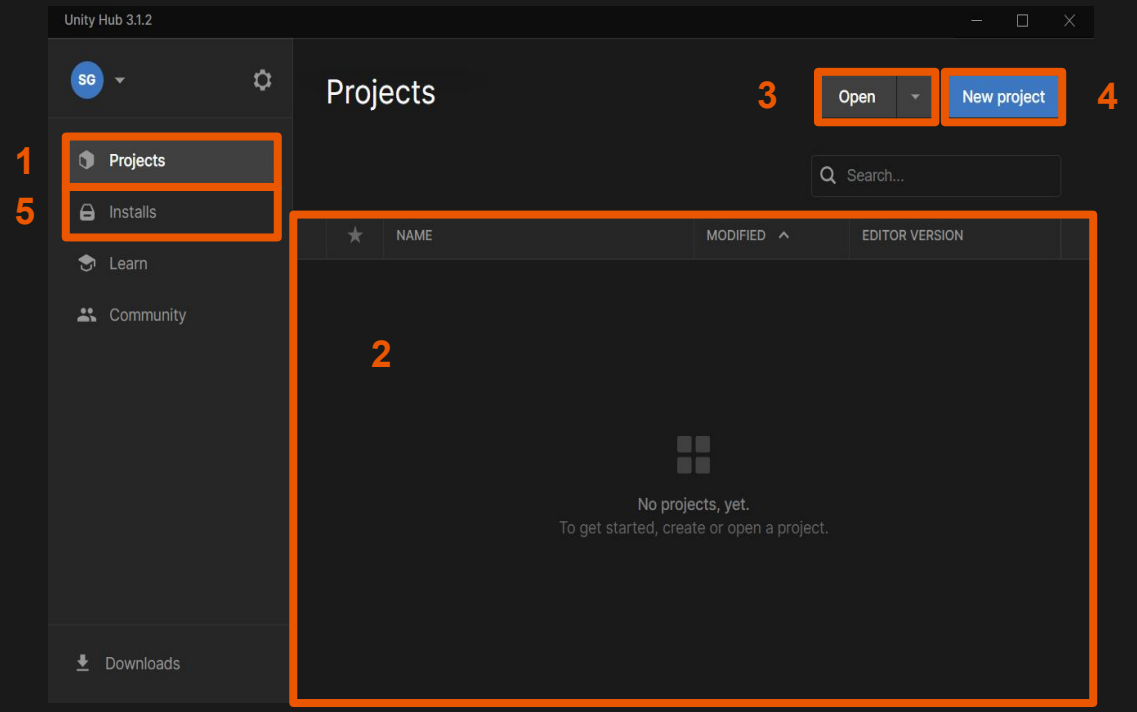
[1] The Projects Tab allows you to view all of the project you've recently worked on.

[2] This is where all the project will selected from.

[3] This allows you to open pre-existing projects that aren't displayed in [2].

[4] Allows you to create a brand new project.

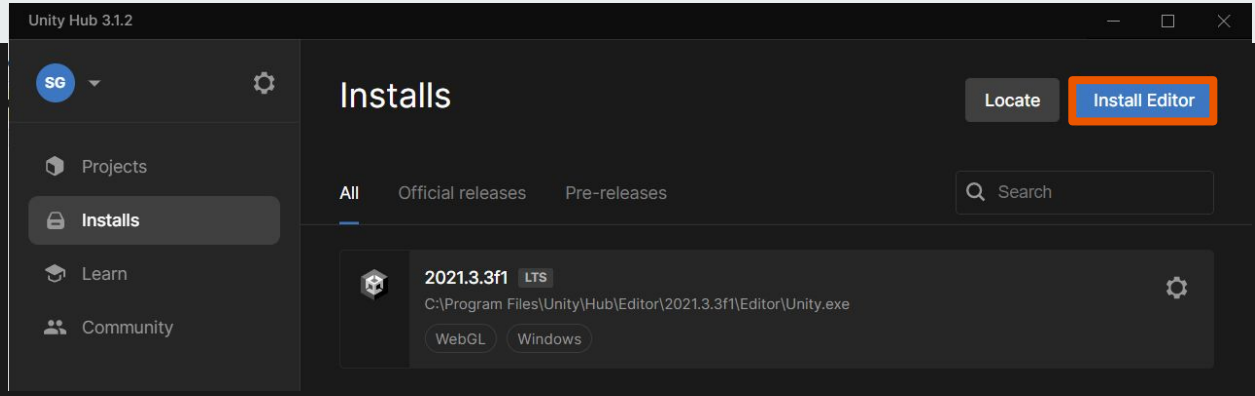
[5] Install tab allows us to manage the versions of Unity that are currently installed and available for us to use.

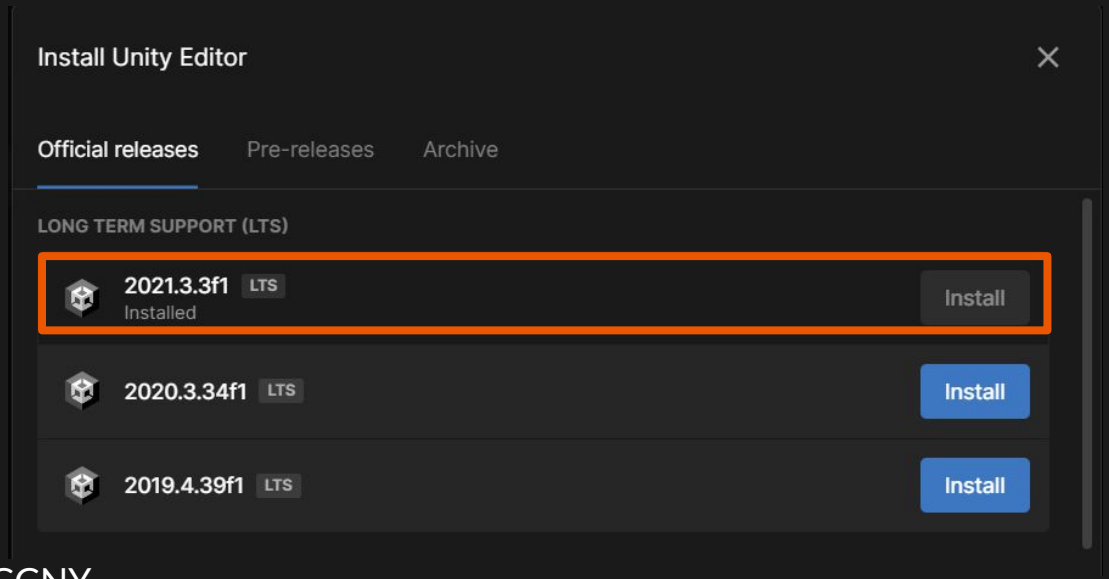


Unity LTS

LTS or Long Term Support is a version of Unity released once a year, it is stable, and it will be continuously supported for two years after launch. Each new year has new features so download the 2021.X.XXX which is the one we'll be using for this class.

This will take a fair bit of time and space, so we will comeback to Unity in our next lesson once everyone has it installed.



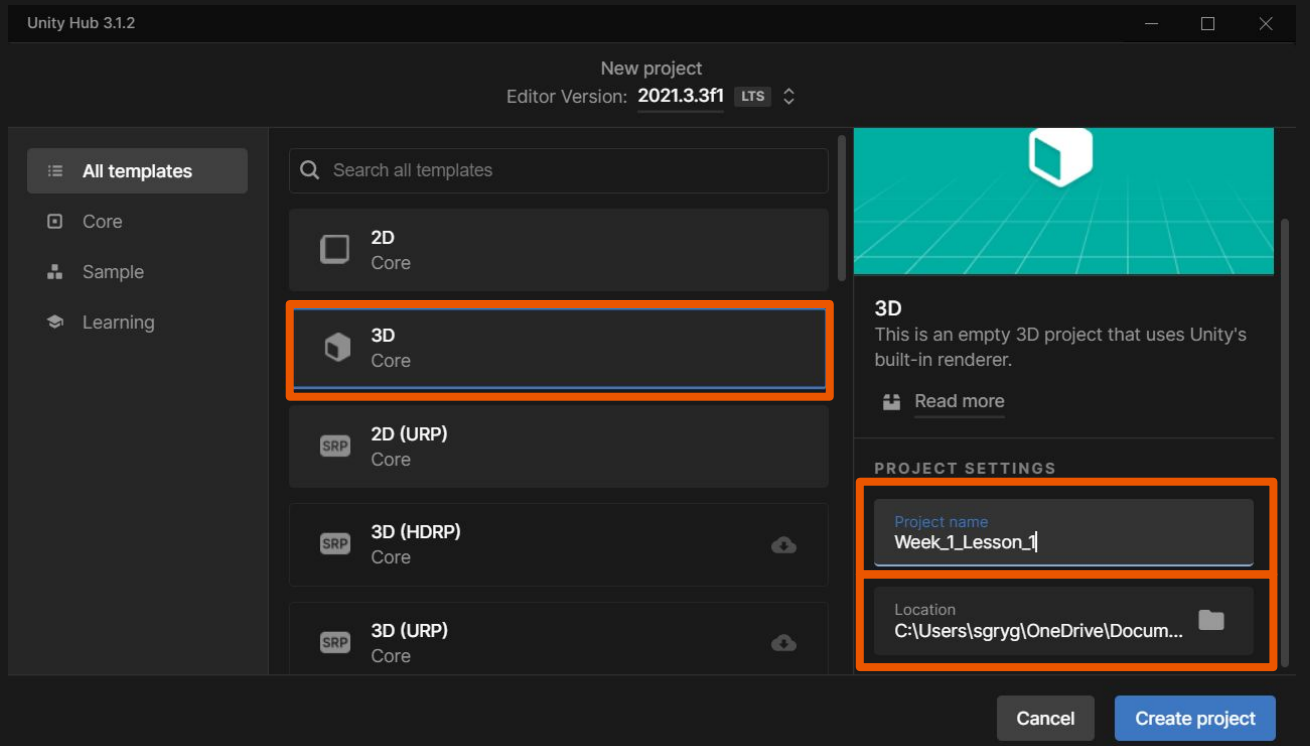


Opening up a new Project

When LTS is downloaded you will be able to head back to the Projects Tab and click, ‘New Project’ and should be met with a similar window to this.

Keep the template to 3D and make sure you choose a name and location of the project.

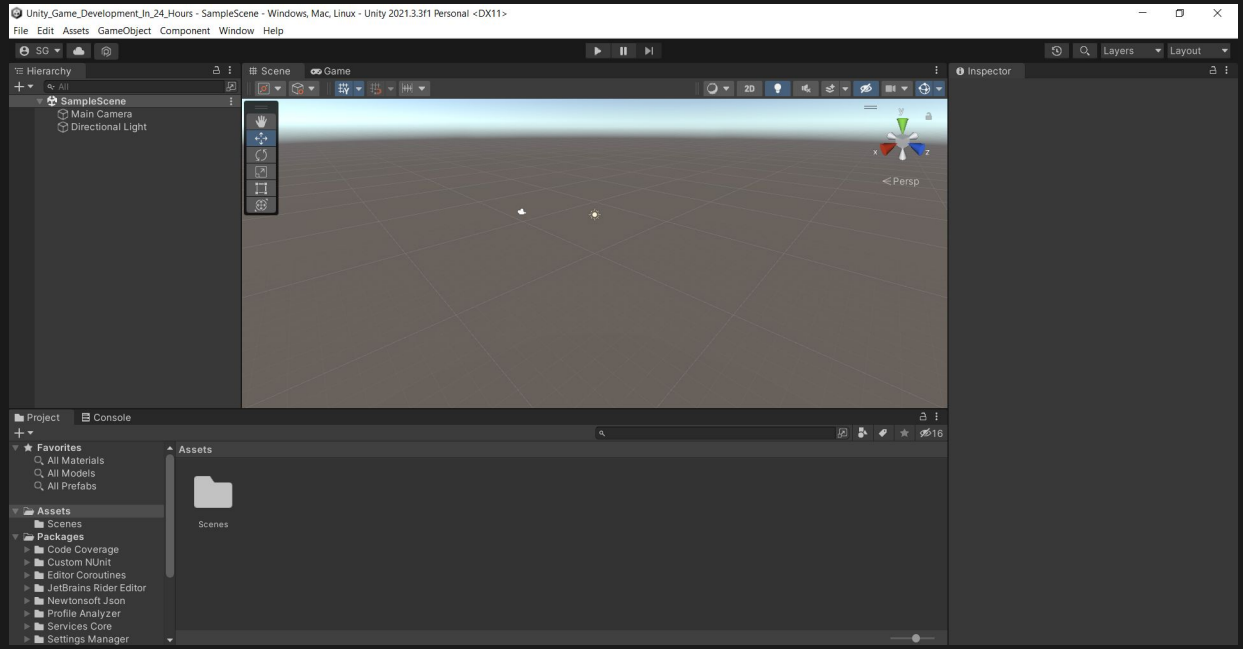
Once all of that is set, feel free to click Create Project.



Unity Once Loaded Up

Creating a new 3D Project will greet you with the Unity Game Engine.

We will go in depth on what you're seeing in our next lesson.



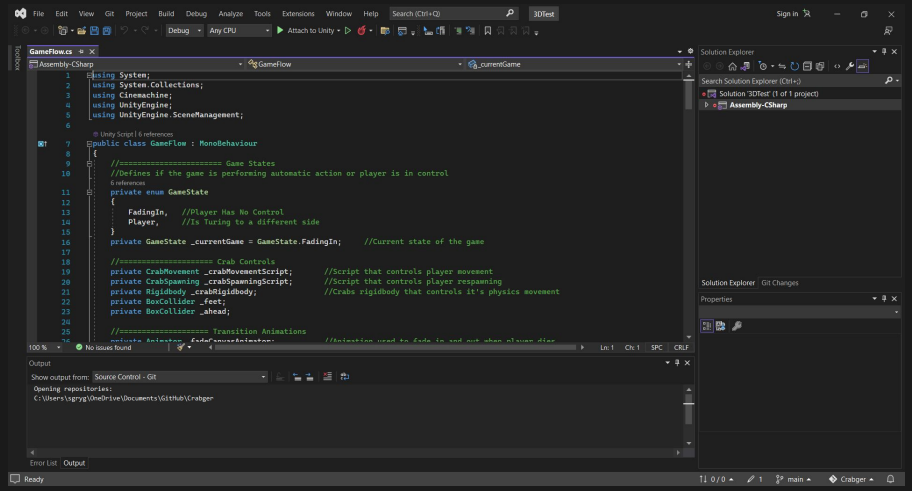
Visual Studio

What is an Integrated Development Environment (IDE)?

- Is a software that allows programmers to write, edit and debug code. In simplest form it's a supercharged Text Editor.

With Visual Studio you will be able to write C# scripts that bring life to all of the Game Objects you will create.

Visual Studio Code is developed by Microsoft.



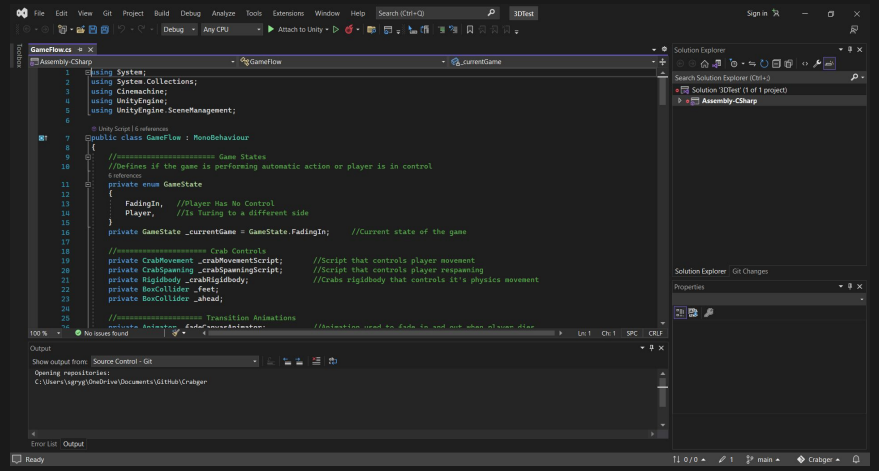
IDE vs Text Editor

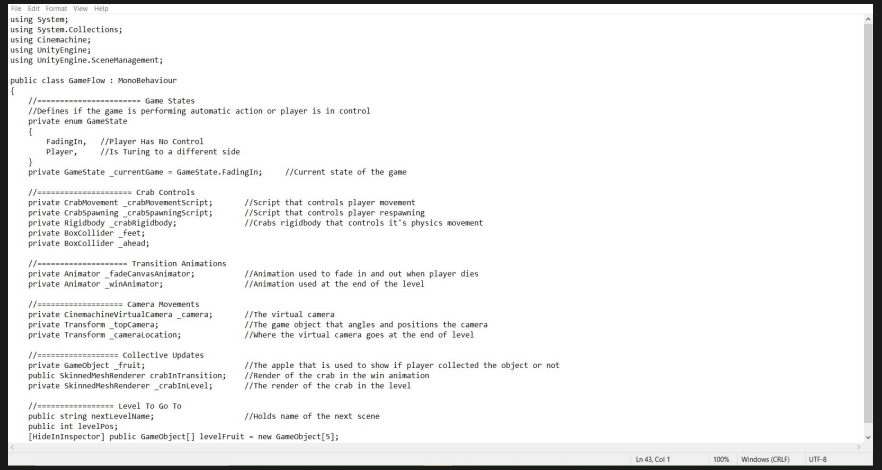
Here is the code shown in Visual Studio versus Windows Notepad. They more or less fulfill the same purpose of allowing you to write code.

However you can immediately tell that Visual Studio Code, color code your work by what the word means. Give you a line counter which is necessary when debugging your program.

Beyond that Visual Studio has useful features such as autocompleting the word or method you are currently writing.

We are using Visual Studio because it works directly with C# and is easily integrated into Unity, different IDEs will be predisposed to working with different programming languages.





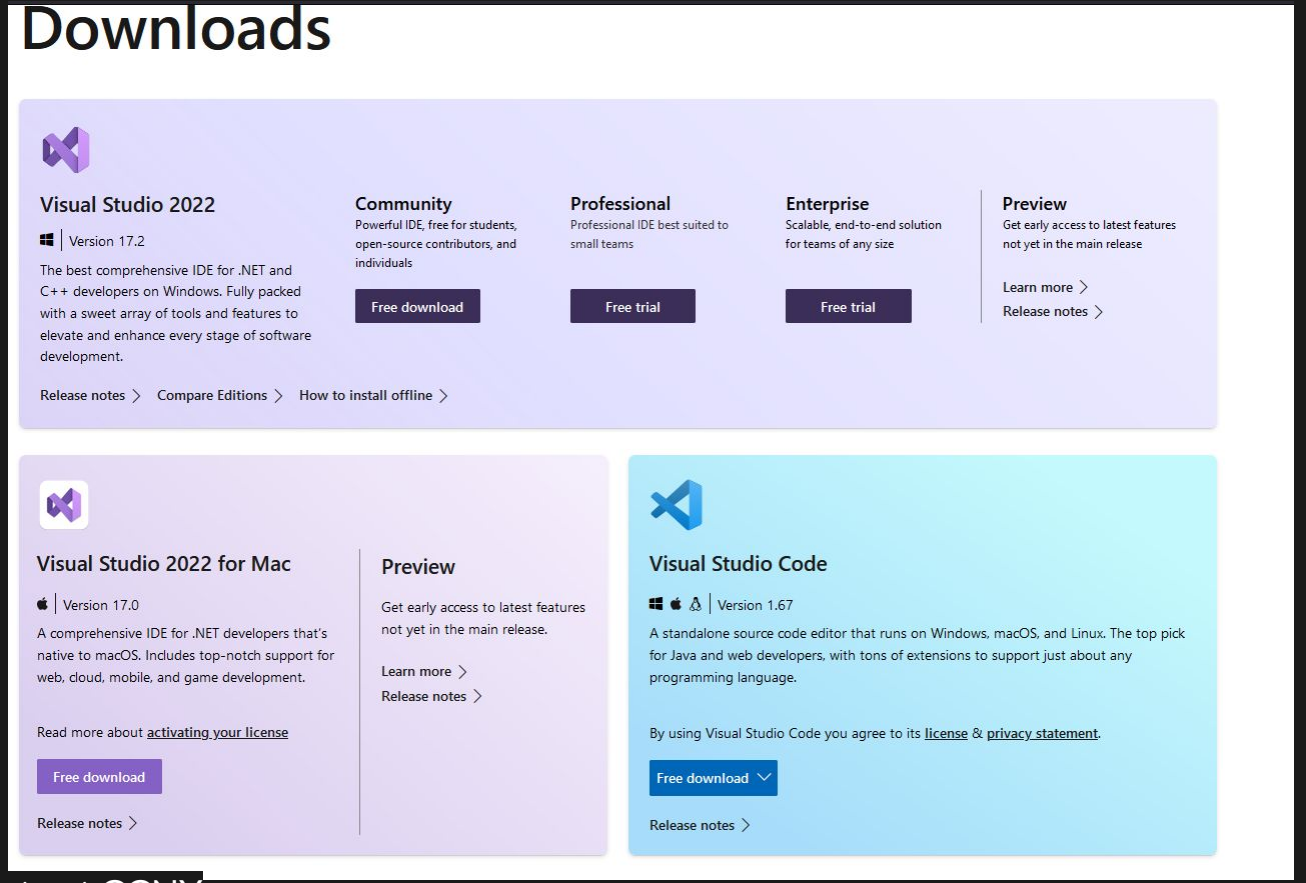
Sebastian Grygorczuk - STEM Institute at CCNY

Downloading Visual Studio

Head over to download: https://visualstudio.microsoft.com/downloads/

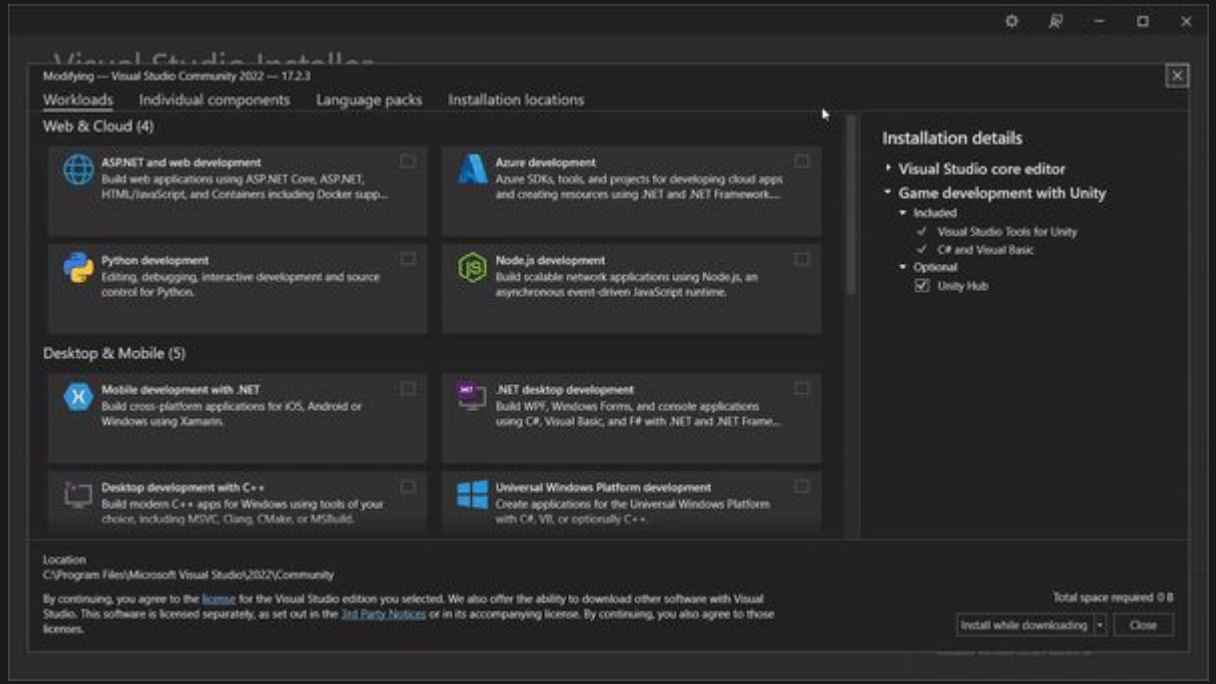
From here select the Community download for Windows and Free Download for Mac.

This will install Visual Studio Installer, a Hub that will allow you to download libraries for whatever programming work you intend to do.



Visual Studio Hub - Workload Installs

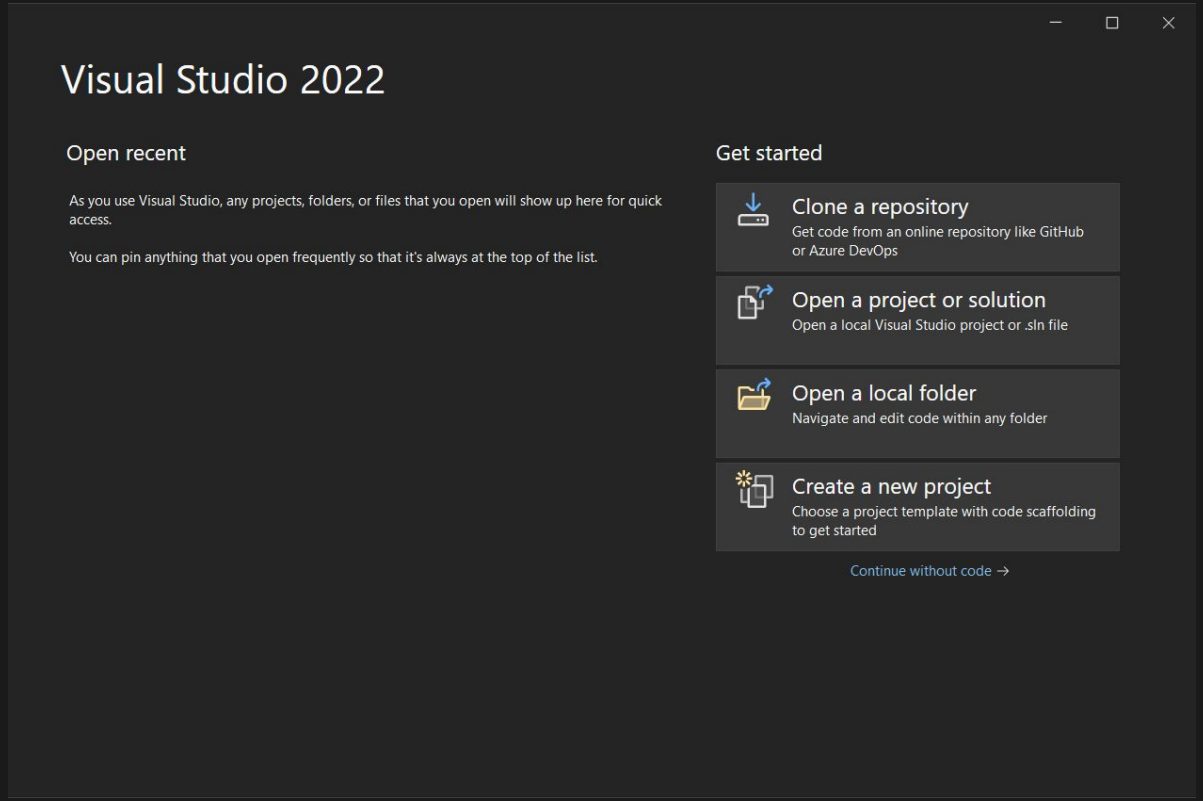
During the installation it will ask you if you want to download any Workloads, scroll down and check the "Game Development with Unity" that's the only thing you'll need for this course.



Visual Studio

Once you finish your installation you will be met with this screen allowing you to select a Script.

For now this is where we'll stop as we will become much more familiar with it once we get into Week 2 of the class.



Diving Into Unity

Sebastian Grygorczuk - STEM Institute at CCNY

Today's Agenda

We're going to be going over Chapter 1: Introduction to Unity, Chapter 2: Game Objects, and Chapter 3: Models, Materials, and Textures

- We're going to get familiar with the layout out all the Unity Views.
- Learn how to navigate through a scene and interact with Game Objects.
- Devel into different Components such as Mesh, and 3D Renderers.

The Unity View

The Unity Engine is made up of a multitude of windows that will be essential to creating your game. The windows we will currently look at are:

[1] Scene View

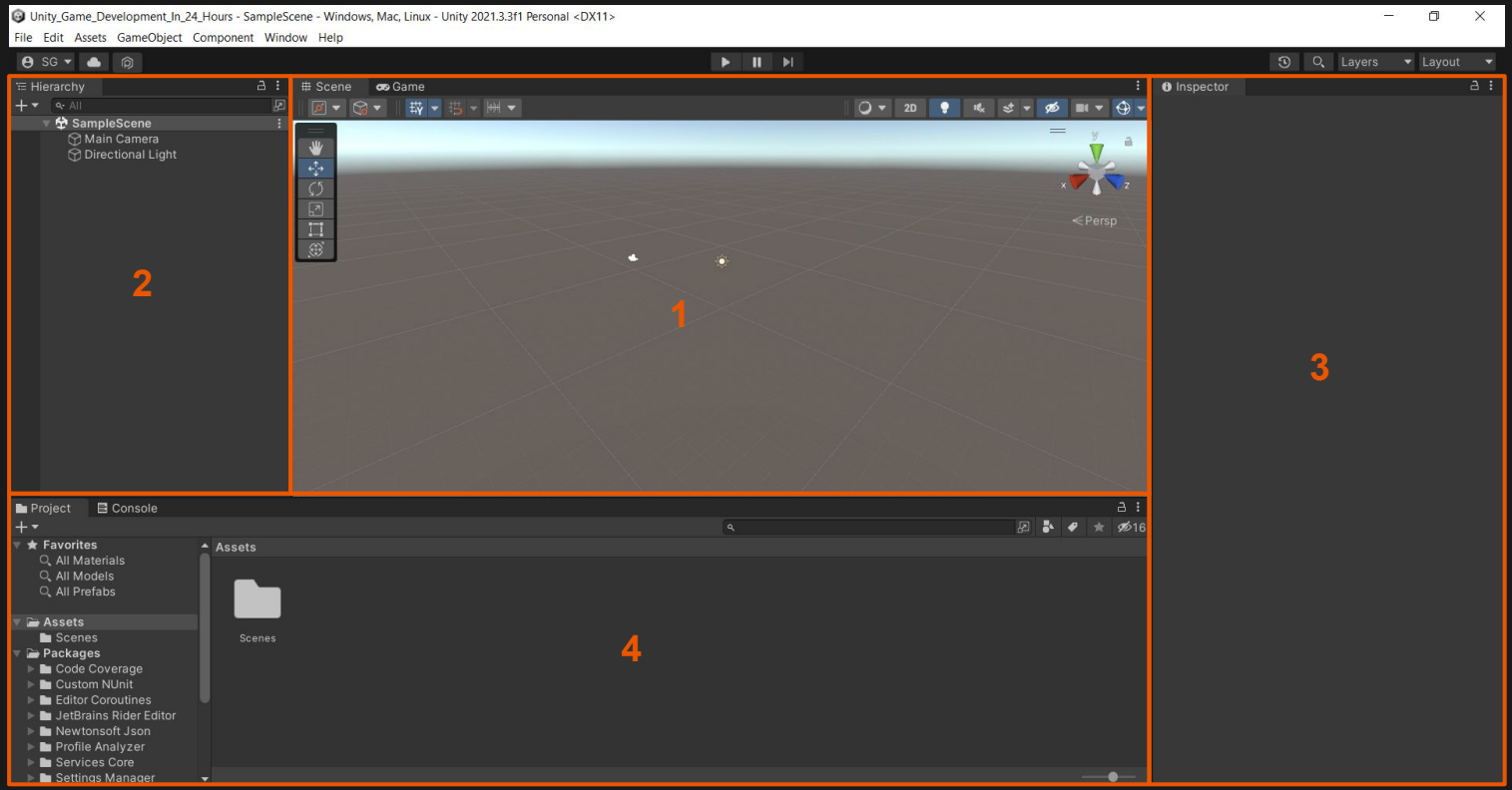
[2] Hierarchy

[3] Inspector

[1] Game View

[4] Project View

[4] Console



Game Object

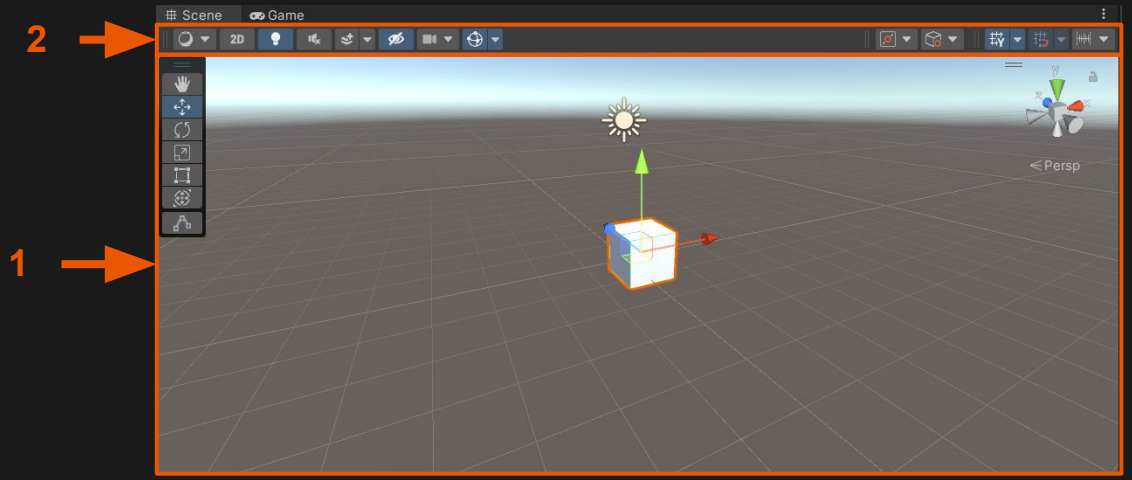
Before we get into the different views I would like to first define what a Game Object is as the two words will come up a lot.

Game Object is a foundational component of anything that will exist in a game. Think of it as an empty container that you can add any property to turing it from nothing into a car, person or anything else you can imagine.

As we make our ways through all of the Views you will be very familiar with Game Objects.

Additional Resources: Page 23 of the Textbook, Game Objects and Components - Unity Official Tutorials

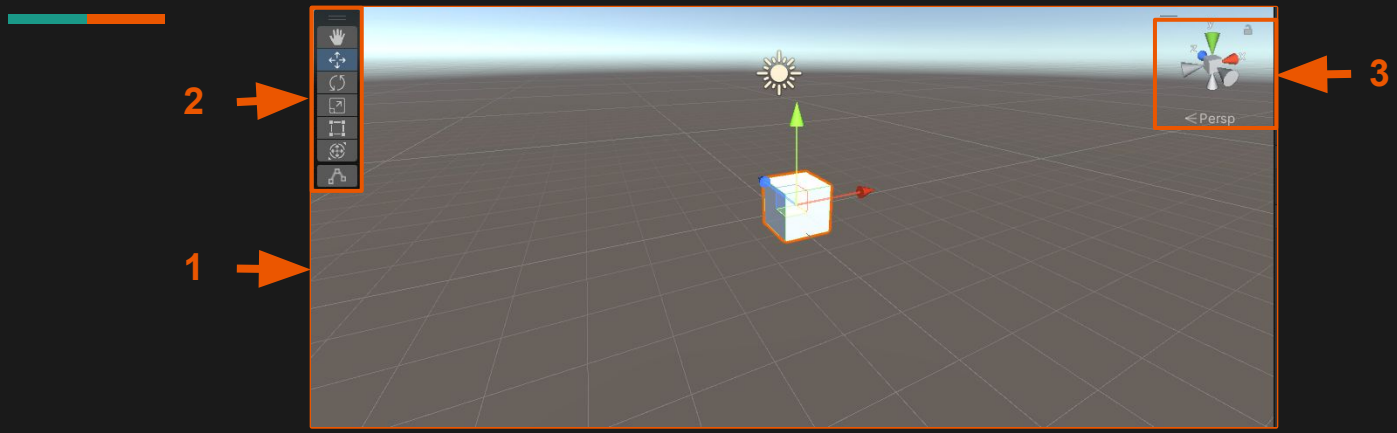
Scene View



The [1] Scene View allows you to visual see the game you are building, and the [2] menu allows you to manually edit the Game Objects in the scene. In this scene we can see two Game Objects, the Cube and the Light Source.

Additional Resources: Pages 14-15 of the Textbook, The Scene View - Unity Official Tutorials

Elements inside the Scene View

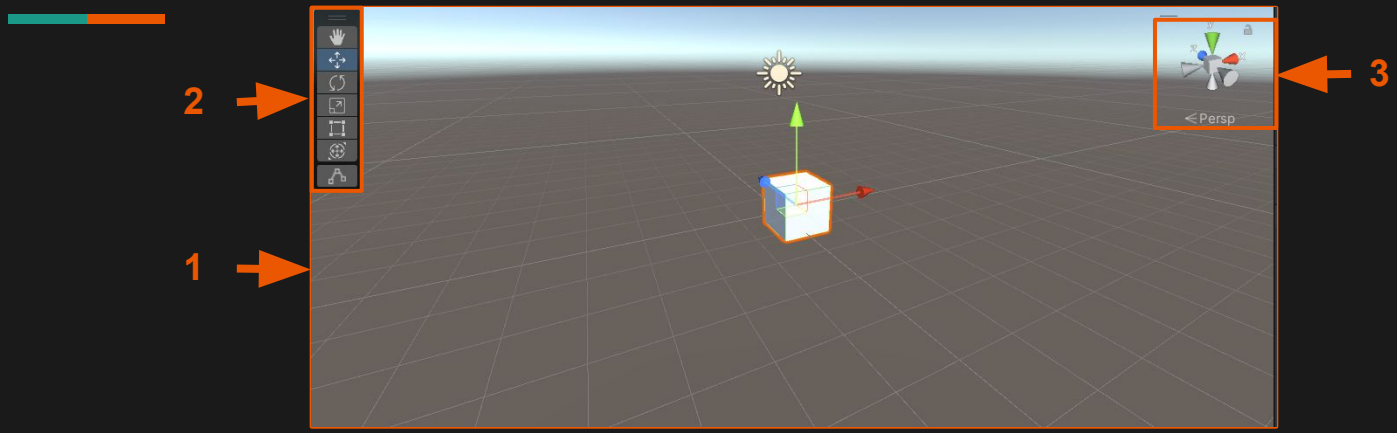


[1] The Scene View, here you can fly/move around the scene or level and edit the position, rotation, and scale of Game Objects using [2] Transformation Tools Bar.

[2] Transform Tools Bar, allow you to manipulate the Game Object that are within the Scene View.

[3] Scene Gizmo, allows you to snap to different planes to see the game at different angles.

Scene View



[2] Transform Tools Bar, this collects different tools that will help you navigate the scene and interact with Game Objects. These tools are are the Hand Tool, the Move Tool, the Rotate Tool, Scale Tool, and Rect Tool.

Hand Tool/Scene Navigation 

When you have the Hand Tool Selected, you can freely explore your scene. There are many button combinations that will allow you to change the way you are navigating.

Holding the Left Mouse Button will let you move along the current 2D plane, allowing you to move left,right, up, and down.

Holding the Right Mouse Button will enter you into Fly Mode which will allow you to use WASD to move freely around the world.

Holding Left Alt locks you in your current position and using the Left Mouse Button you can now rotate around that point or using the Right Mouse Button you can zoom in or out.

You can also zoom in or out using the Mouse Wheel at any point.

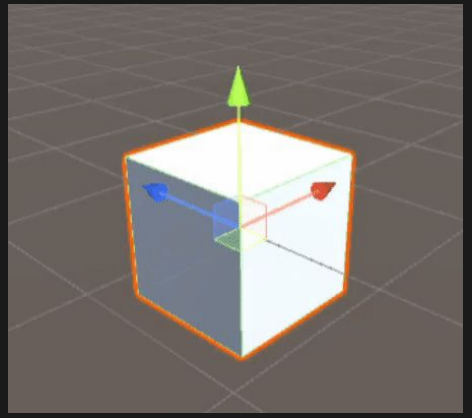
Additional Resources: Pages 18-20 of Textbook

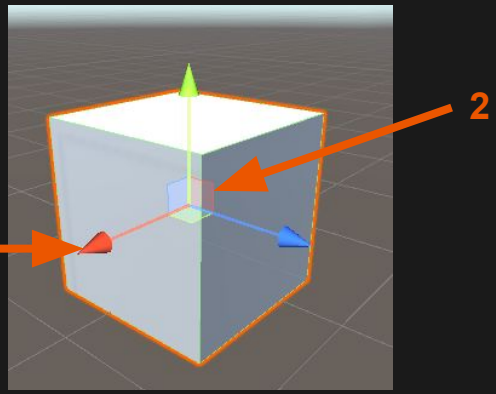
Transform Tools - Move Tool

The move tool allows you to change the position of the Game Object you are currently looking at. You will know what Game Object you are currently view as it will show the handles.

[1] The arrows allow you to move the object along individual axis, X,Y, Z and red, blue, green respectively. This will correspond to the colors on the Scene Gizmo.

[2] The planes will allow you to move the Game Object along the three different plains. XY, XZ, and YZ, and blue, green, and red respectively.





Additional Resources: Pages 32-33 of Textbook

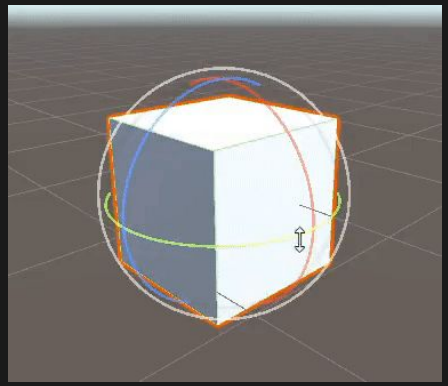
Transform Tools - Rotate Tool

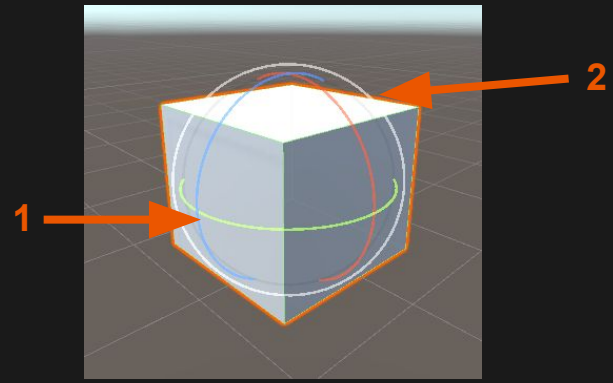
The rotation tool allows you to rotate the object you are working with.

[1] The three rings are broken down into individual colors of red, green and blue, and allow you to individually adjust the rotation of X, Y, and Z respectively.

You will notice a yellow arc show up as you are rotating, it shows you the amount of degrees you rotated from your original rotation.

[2] The White Ring will rotate the around the Scene View Z Axis





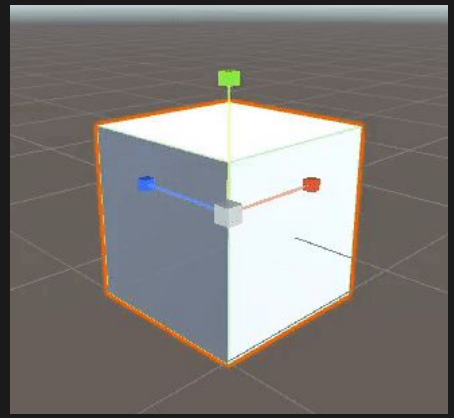
Additional Resources: Pages 33-34 of Textbook

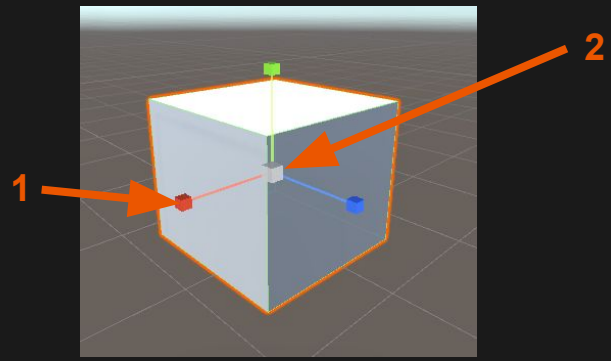
Transform Tools - Scale Tool

The scale tool allows you to modify the size of each side of the Game Object that you are currently looking at.

[1] The Arrows allow you to modify each of the axis individually X, Y, Z with red, green and blue representing each respectively.

[2] The Middle Notch will allow you to scale all three axis at the same rate.





Additional Resources: Pages 34-35 of Textbook

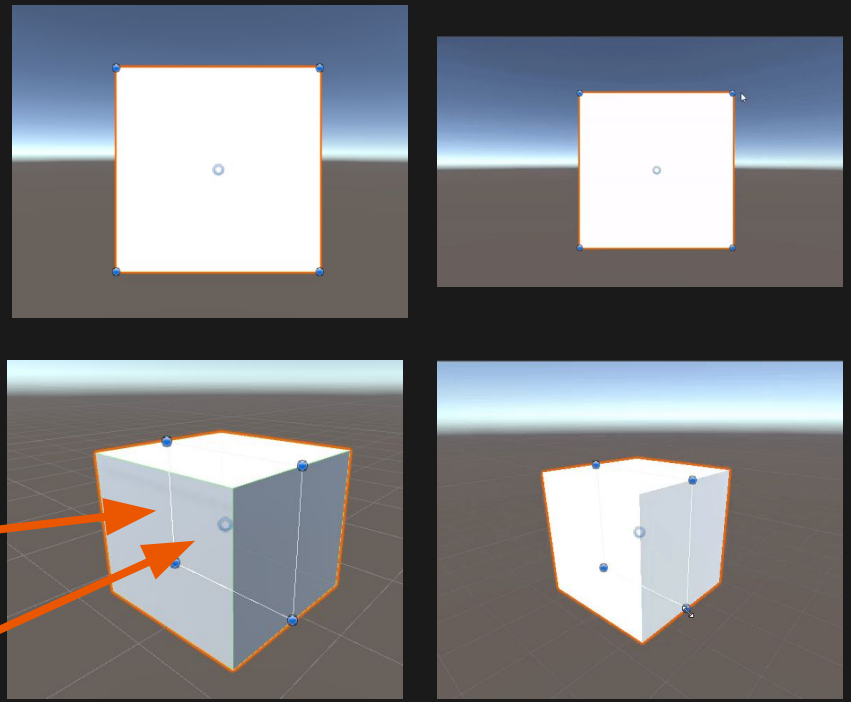
Sebastian Grygorczuk - STEM Institute at CCNY

Transform Tools - Rect Tool

The Rect Tool combines all three of the previous tools into one, allowing you to translate, rotate and scale the object all at once.

[1] As you can see however the Rect Tool works on a plane, so you will only be modifying the object in XY, XZ, or ZY planes therefore it's relegated for use on 2D elements such as 2D game object or UI elements.

[2] Pivot Point is the center of the object and is the point the object rotates around.

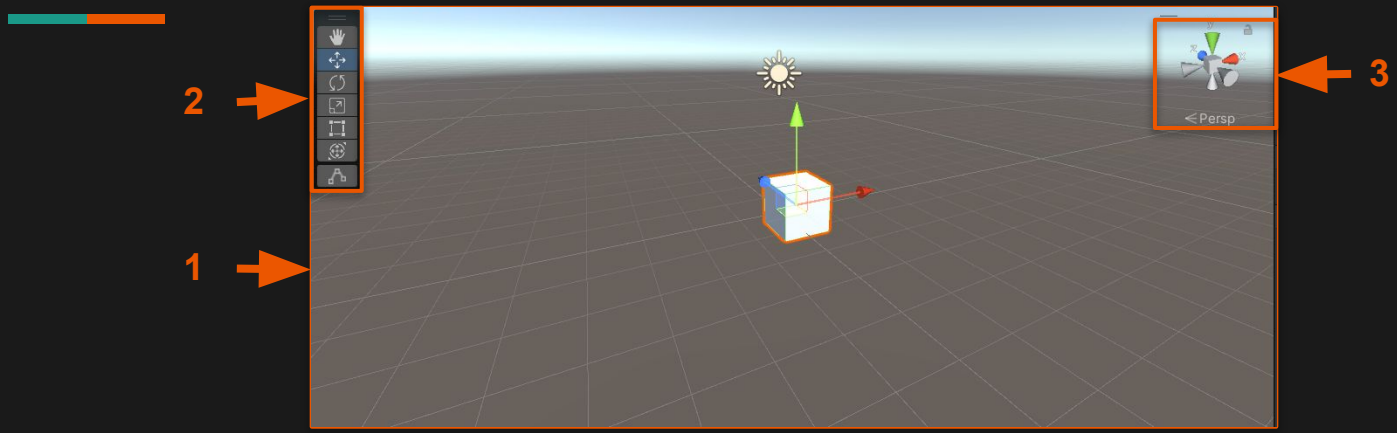


Tool Hotkeys

All of these functions are mapped to Hotkeys. If you would rather quickly swap between settings instead of selecting from the menu, here's a helpful chart.

Tool	Hotkey	Function
Hand	Q	Navigate in the Scene
Move	W	Translate selected object
Rotate	E	Rotate selected object
Scale	R	Resize selected object
Rect	T	Manipulate 2D object

Scene View



[3] Scene Gizmo allows you to quickly snap to different planes XY, XZ, ZY, -XY, -XZ, and -ZY in addition to changing the camera view from perspective to isometric and likewise. This can be incredibly helpful in transforming Game Objects as manipulating them in 3D space

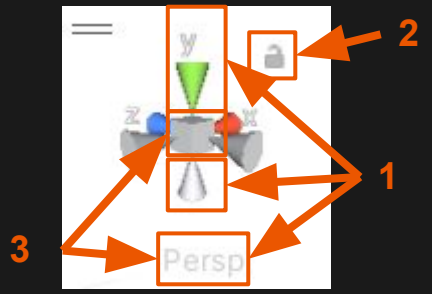
Scene Gizmo - Six Planes

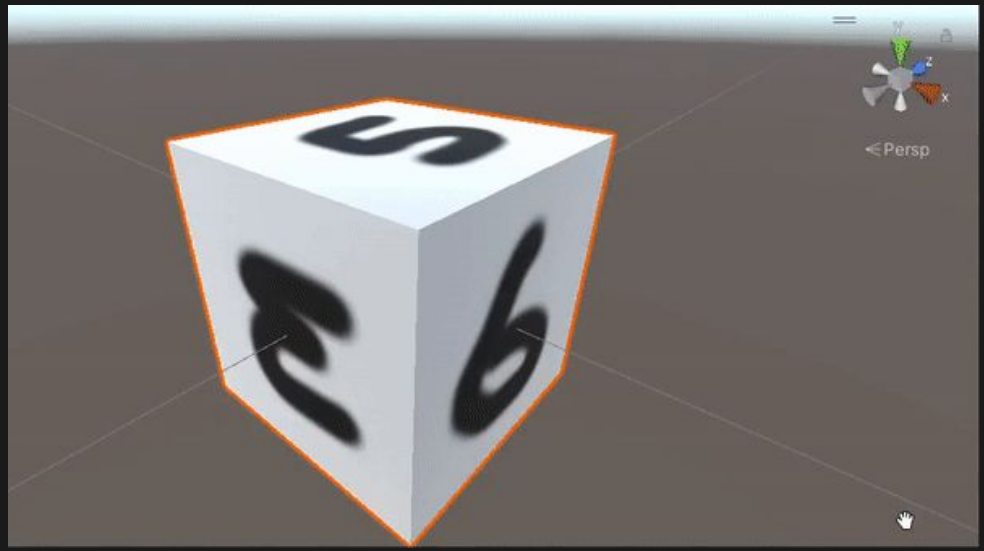
Transforming Game Objects in 3D space can be very hard to orient and translate. Thus snapping to a 2D axis could allow us to much easier move the object to desired position, rotation or size.

[1] There are six cones, ones for the positive X, Y, and Z axes and ones for the negative X, Y, and Z axes. Clicking on the Y cone would snap you to the XZ plane.

There is a text indicator that helps you figure out the orientation, while performing the snapping you will go through Top (ZX), Front (-XY), Back(XY), Left (-ZY), Right (ZY) and Bottom (-ZX).

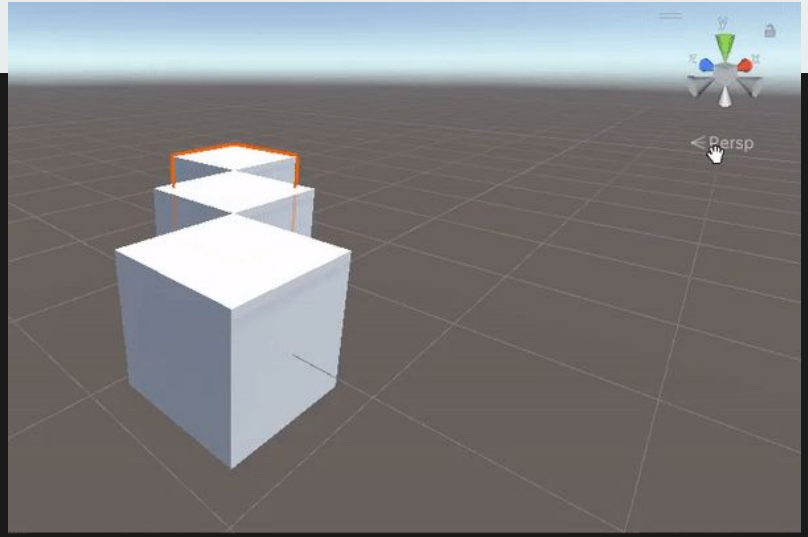
[2] You can lock the view to the current rotation in case you don't want to accidentally move the scene camera.





Sebastian Grygorczuk - STEM Institute at CCNY

Isometric Vs Perspective

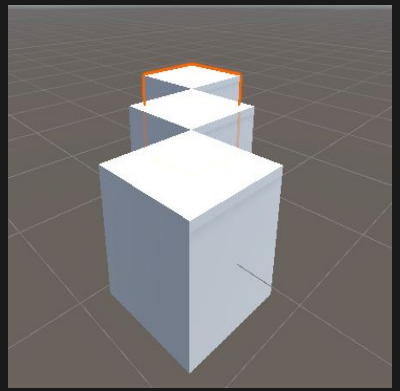


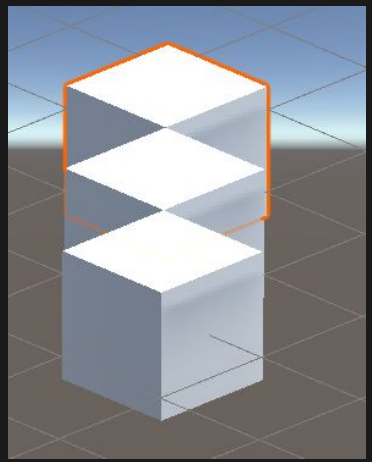
[3] You may have noticed that before we snapped to a single plane the Scene Gizmo indicated that we were in Persp, or Perspective mode.

Perspective is the way we view the world, the further a object is from us the more foreshortened are it's dimension are in relation to us.

Isometric on the other hand does away with that distortion by keeping the projects on all three axis equal. This is helpful when we want to have a detailed drawing but we lose Depth in the process.

To switch between these you can tap on the Cube in the middle of the Scene Gizmo or on the text.

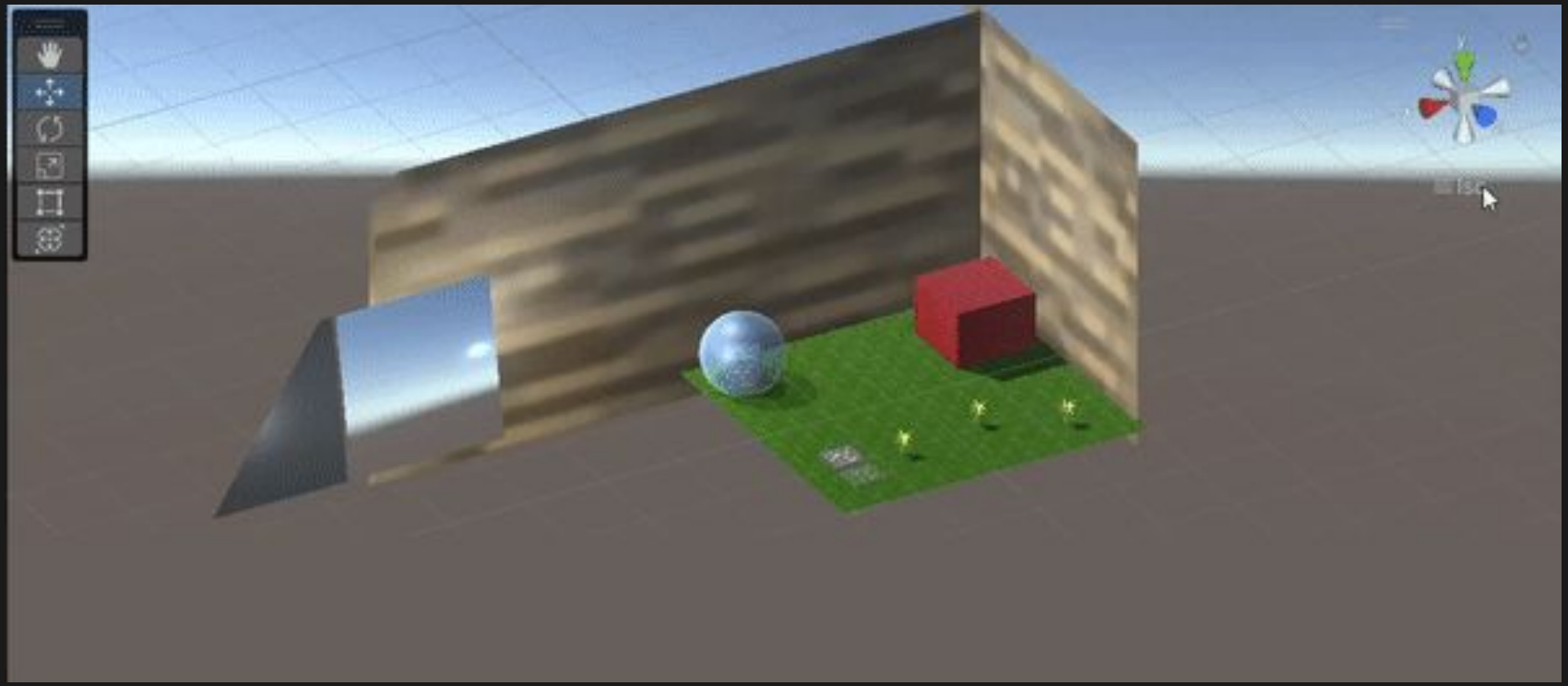




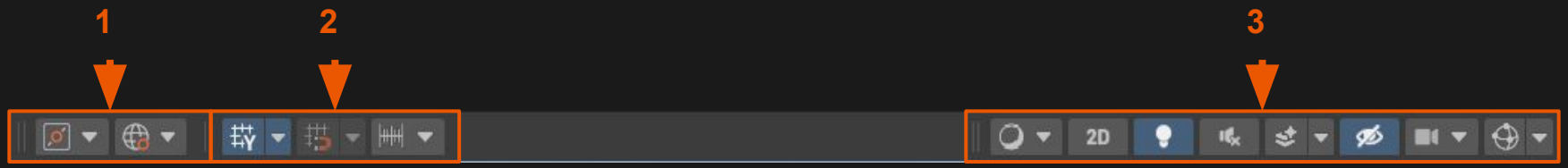
Additional Resources: **Isometric vs Perspective**

Sebastian Grygorczuk - STEM Institute at CCNY

Isometric Vs Perspective Example



Scene View Control Bar



The Scene View Control Bar breaks down into three tools sets, these will further help us manipulate the object to our desire and display the scene that is best at the moment.

[1] Tool Settings, Modifies how the tools work

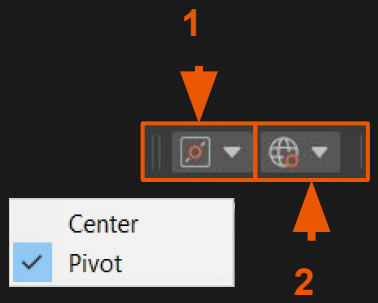
[2] Grid and Snap, Controls the Grid

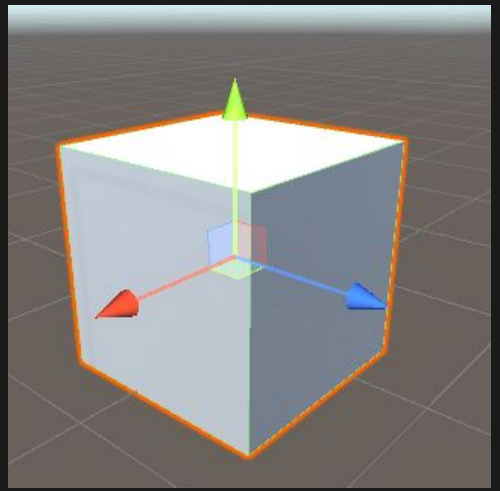
[3] View Options, Controls how the Scene is show to you

Tool Settings: Pivot

[1] Handle Position Toggle lets you switch where you'd like the Transform Gizmo to show up on the Game Object. So far we have seen it always show up in the center of the Game Object, this act as the origin point for the Game Object.

However, if you import an item from a 3D modeling software it may have a pivot point that's not it's center. Unity allows you to use either one as your preferred origin point.

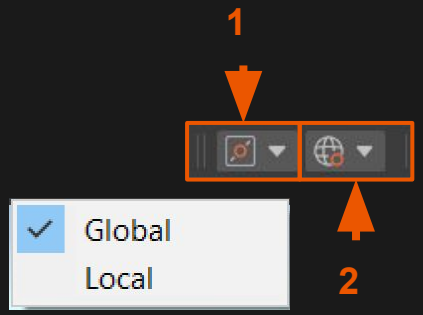


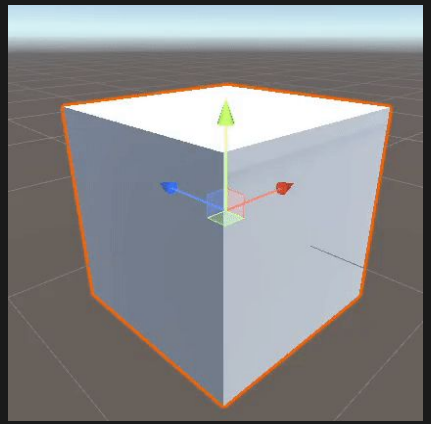


Tool Settings: World vs Local Coordinate

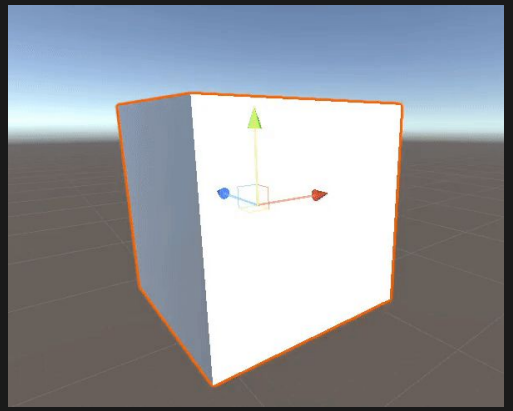
[2] As we have been transforming our Game Objects they always referred to their Origin Point as (0,0,0). This means they are abiding by the World Coordinate System. Each object also has their own coordinate they keep track of, mainly in respect to its rotation.

This is a very useful feature for character control. Rather than having to adjust which way you're moving in accordance to the world you can just program to move forward and using the Local Coordinates you just move in the axis you deemed as forward.





World Coordinates



Local Coordinates

Additional Resources: Pages 28-29 of Textbook, Unity Tutorial - Local Space Vs World Space in Unity

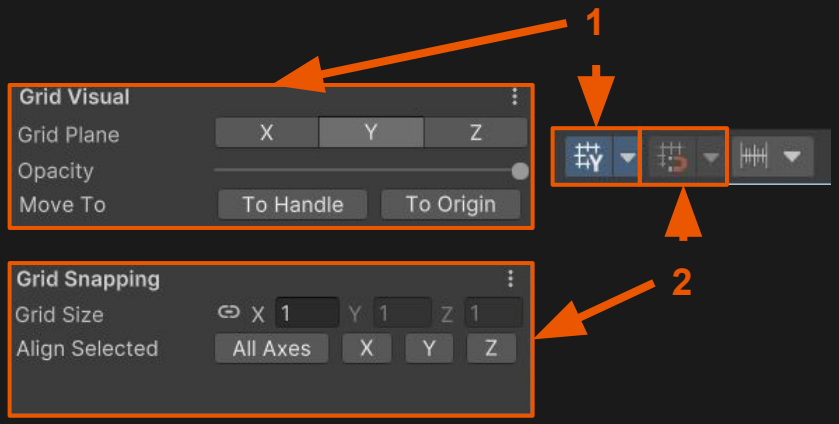
Grid and Snap

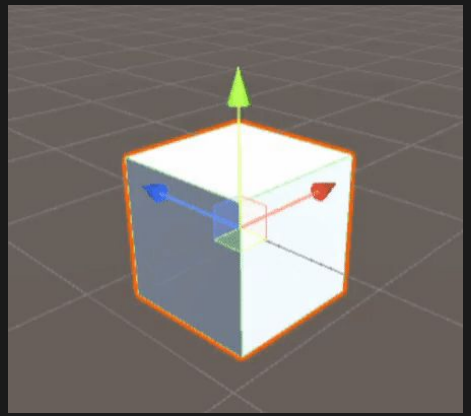
[1] The Grid Visual allows you to toggle the Grid On and Off, set the axis on which you'd like it to show up on and adjust the opacity [transparency] of the grid.

[2] Grid Snapping allows you to toggle on and off if you'd like your Move Tool to snap to the points along the grid. You can change the distance between each point.

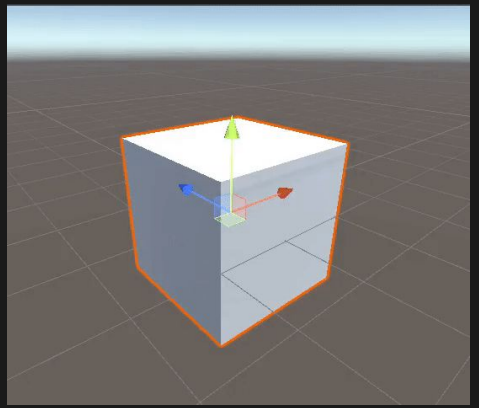
Grid Snapping is only available while you are translating the Game Object in World Coordinates.

Using the grid you can have better time placing items precisely.





Snapping Off



Snapping On

Sebastian Grygorczuk - STEM Institute at CCNY

View Options

The View Options give you full control over how and what is displayed in the scene.

[1] Draw Mode picks how the objects should be rendered.

[2] Light, Audio and Other Visual Effects Toggles checks how the scene works.

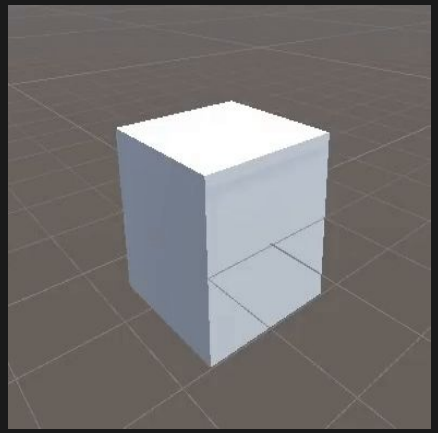
[3] 2D Toggle flips between using the 3D space or using 2D space.

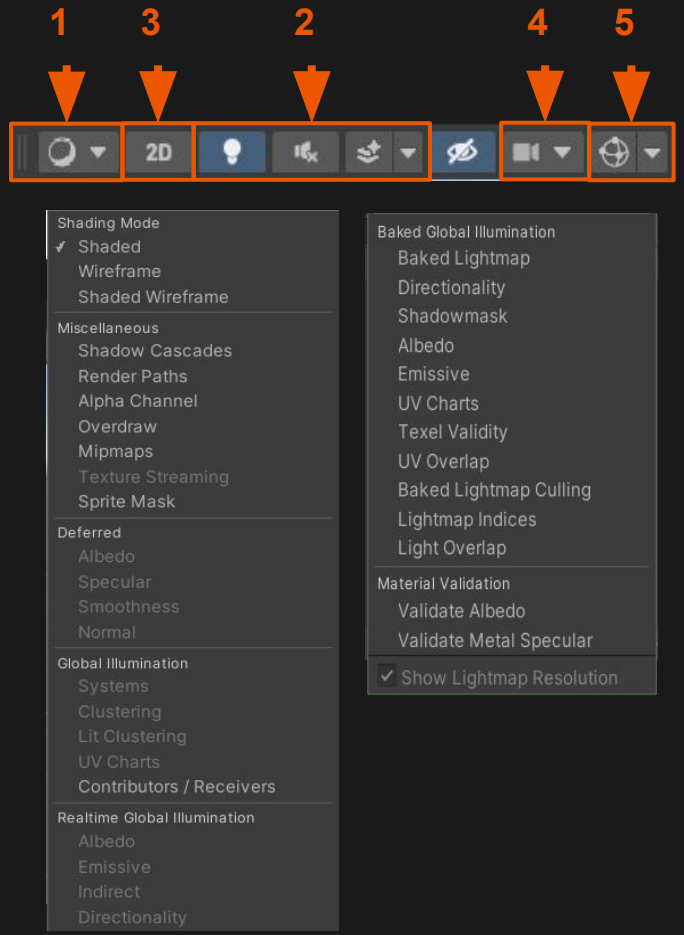
[4] View Scene Camera Options are camera settings that affect how we move around in the Scene View.

[5] Gizmo Visibility Toggle controls Icon display

View Options - Draw Mode

[1] Draw Mode allows you to switch between different version of rending the objects in the scene. For the most part we will stay in the default Shade Model option as all of these option pertain to advanced lighting ideas.





Additional Resources: Scene View Draw Modes For Lighting

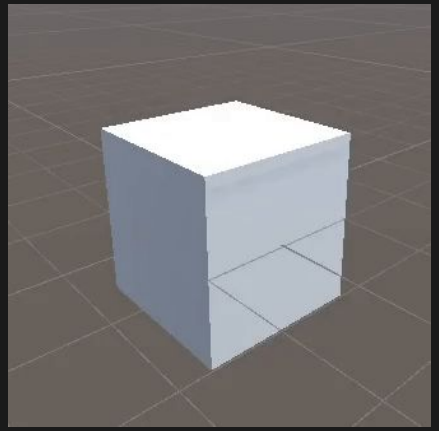
Sebastian Grygorczuk - STEM Institute at CCNY

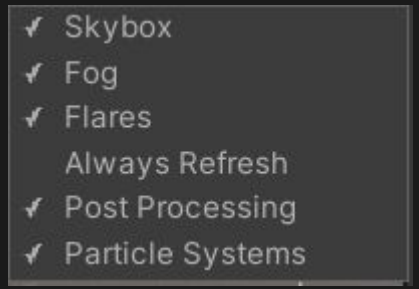
View Options - Lighting, Audio

[2] These three toggles are pretty simple, you can toggle on and off

- Lighting Effect
- Audio Effects
- Visual Effects

This can allow you to see what the Game Objects look like when unaffected by the environment.





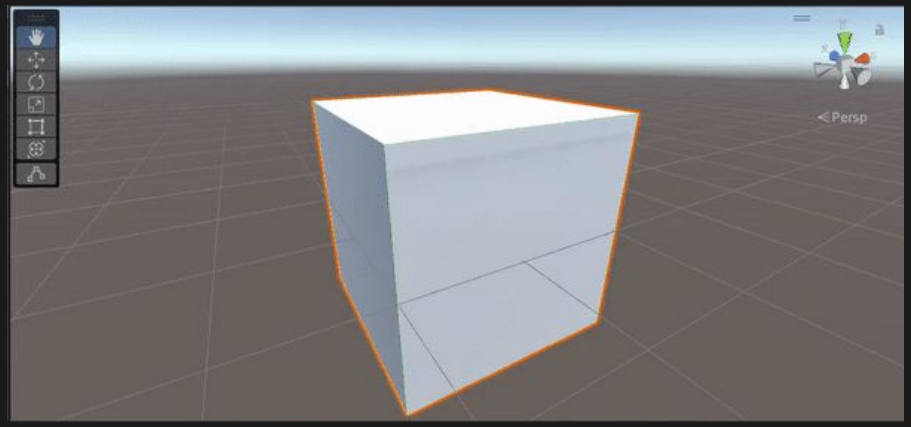
Turning Off/On Lighting and Visual Effect

View Options - 2D Mode

[3] 2D On/Off Toggle turning this on will flatten your view against the 2D XY plane.

This will remove your ability to use the Scene Gizmo to snap or go into Isometric View.

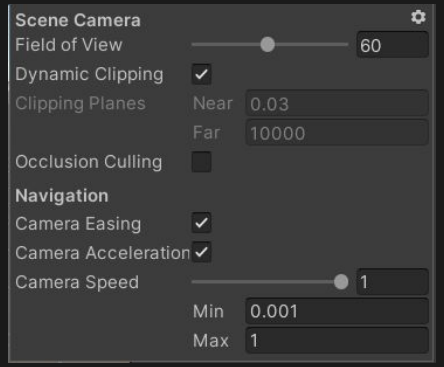
This option will be turned on whenever creating a 2D game or working on UI.

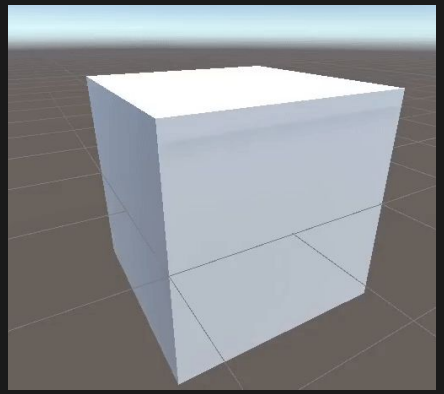


View Options - View Scene

[4] View Scene Camera, allows you to control the Field of View. This dictates how near or far the vanishing point is. The smaller the Angle the closer to Isometric view we are the further the angle the more distorted the object is. Default is 60 degrees.

Navigation settings are there to help you with Fly Mode, Easing and Acceleration make the movement feel natural and you might want to set the max and current speed to whatever is comfortable It for you.



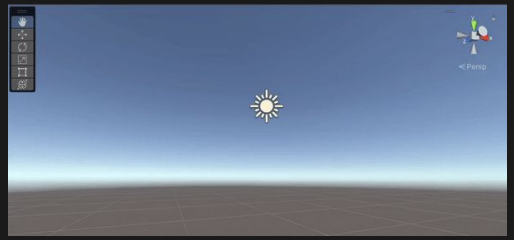


Additional Resources: Focal Length and Field Of View

View Options - Gizmo Toggle

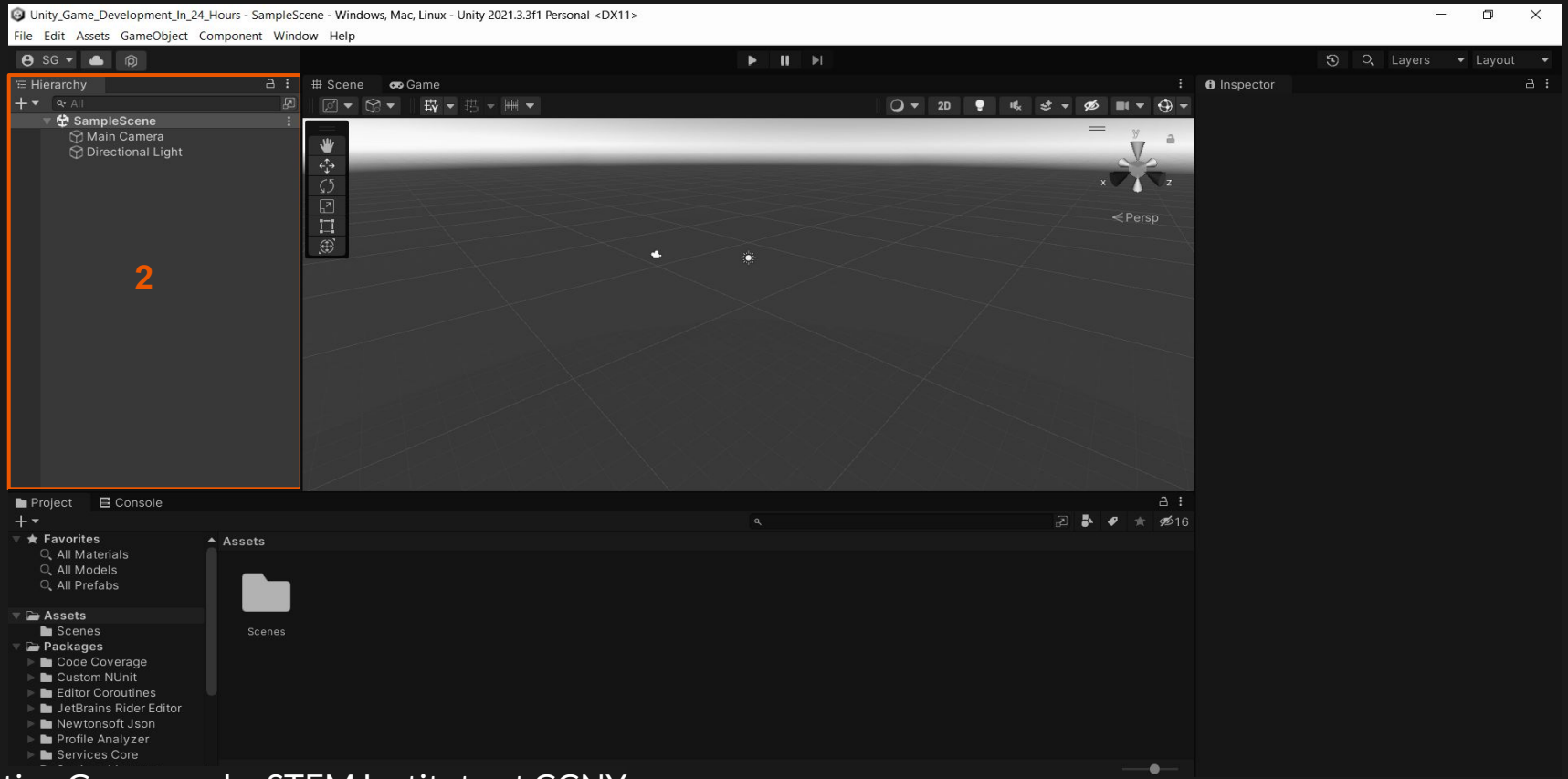
[5] Gizmo Toggle allows you to turn off and on visual for game object that may not have visual presence. You've already seen two of those, the camera and the light source both have icons that display their location in the scene.

As you can see there are many icons that could be on screen and you have full ability to change which are on and off.



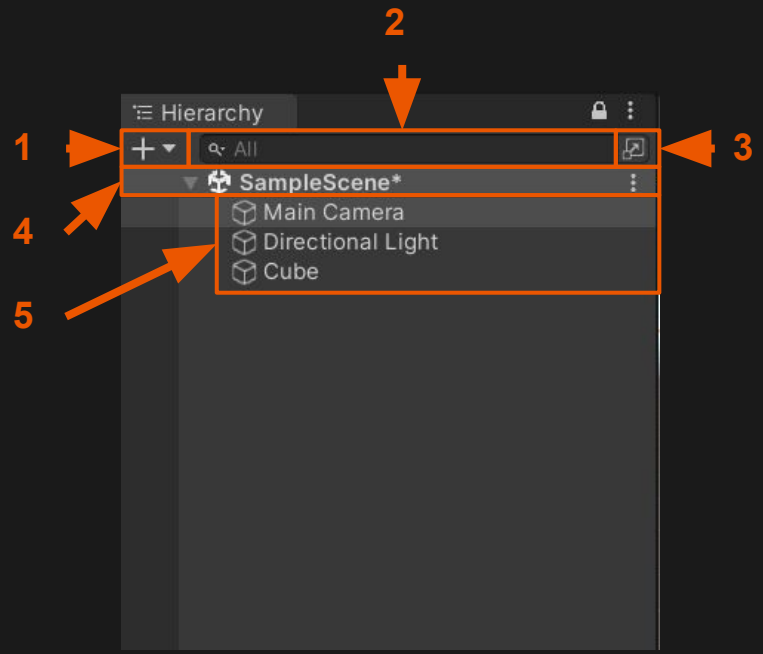
One setting you might want to be aware of is the 3D Icon, toggling it on and off will leave them ay a standard view no matter where you are or keeping their size dependent on your position to it. a

Hierarchy


Unity_Game_Development_In_24_Hours - SampleScene - Windows, Mac, Linux - Unity 2021.3.3f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
2
Scene
Game
Persp
Inspector
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Packages
Code Coverage
Custom NUnit
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core

Hierarchy View

Hierarchy View displays a list of every Game Object that's currently in the scene. This View is invaluable when looking for anything as searching for Game Object through the Scene View could be almost impossible once the amount of Game Objects in a scene is large enough.

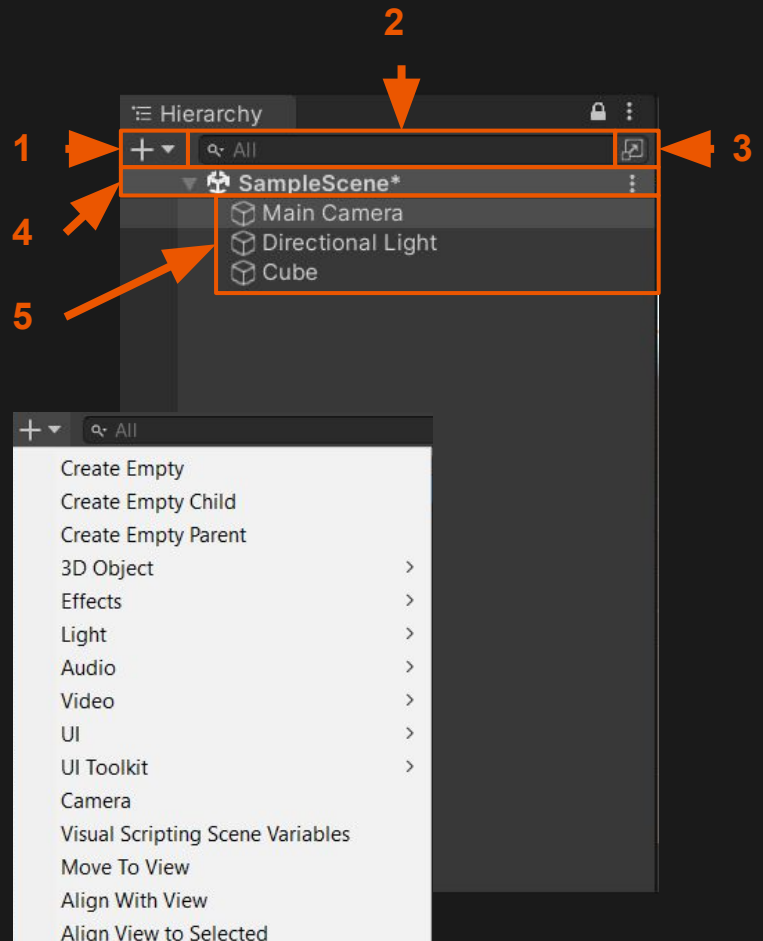


Additional Resources: Page 12 of the Textbook, The Hierarchy Window - Unity Official Tutorials

Hierarchy View

[1] Create, this drop down menu allows you to create a new Game Object. You can create an Empty one or a Unity Premade One with select components. We will create many of these throughout this course.

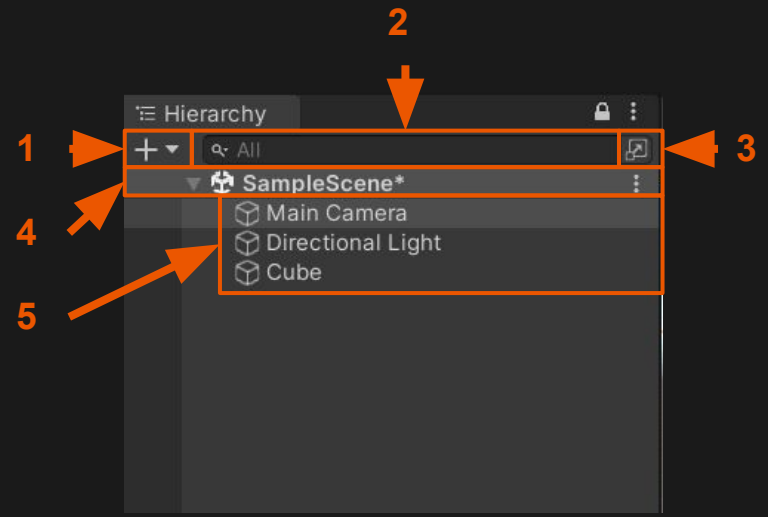
You can also create a Game Object by right clicking on the empty space in the Hierarchy or clicking the Component button in the File Menu Header.

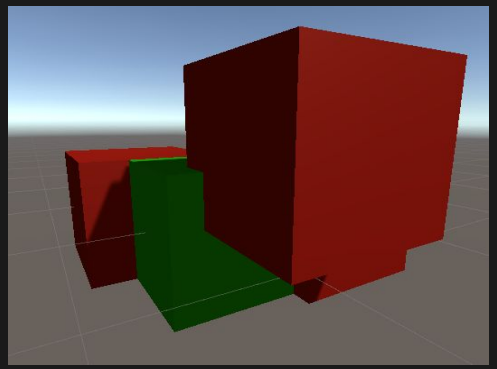


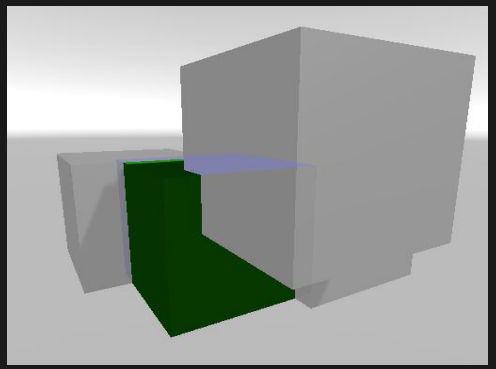
Hierarchy View

[2] Search Bar, will help you find the Game Object, help you find anything that contains what you type into it. Thus if you type in “Cube”, it will show you every Cube you have in the Scene. If you have a very specific Cube in mind, it’s best to give your Game Objects descriptive names.

A wonderful feature of the search bar is that it work in conjunctions with the Scene View and will grey out any object that doesn’t contain the word typed.



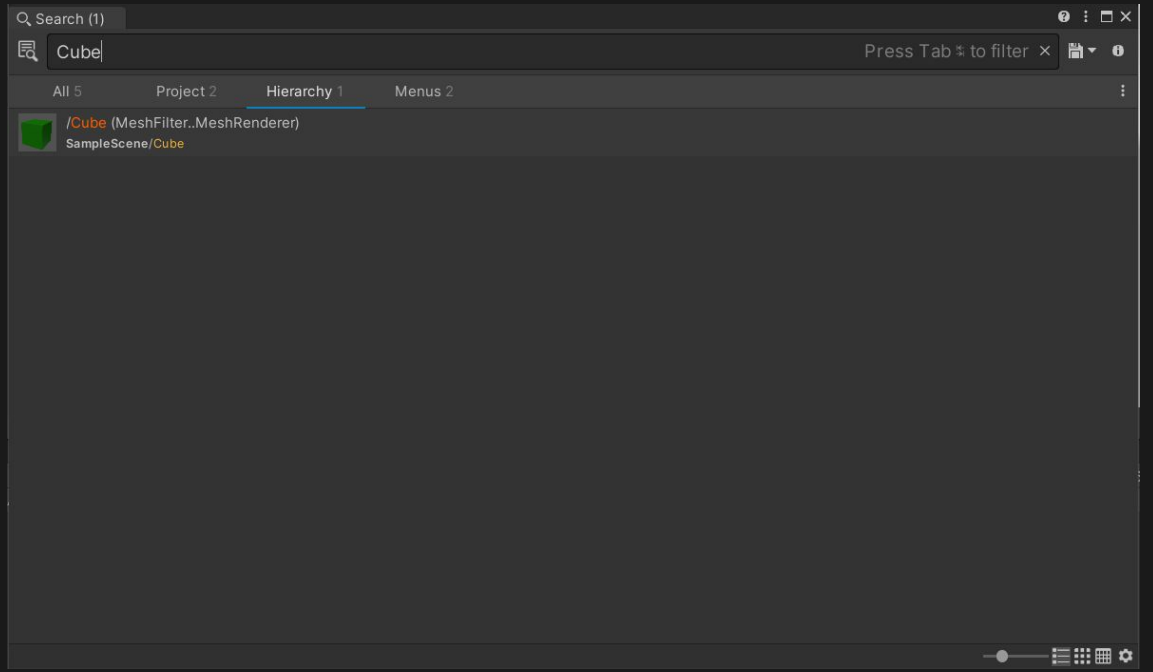




Hierarchy View

[3] Advanced Search, clicking the button next to the search bar will bring up the Advanced Search Window. Here you can look for the Object across Projects, Scene, Hierarchy, and more.

For projects on our scale we won't have to rely too much on this tool but it's good to know about it.



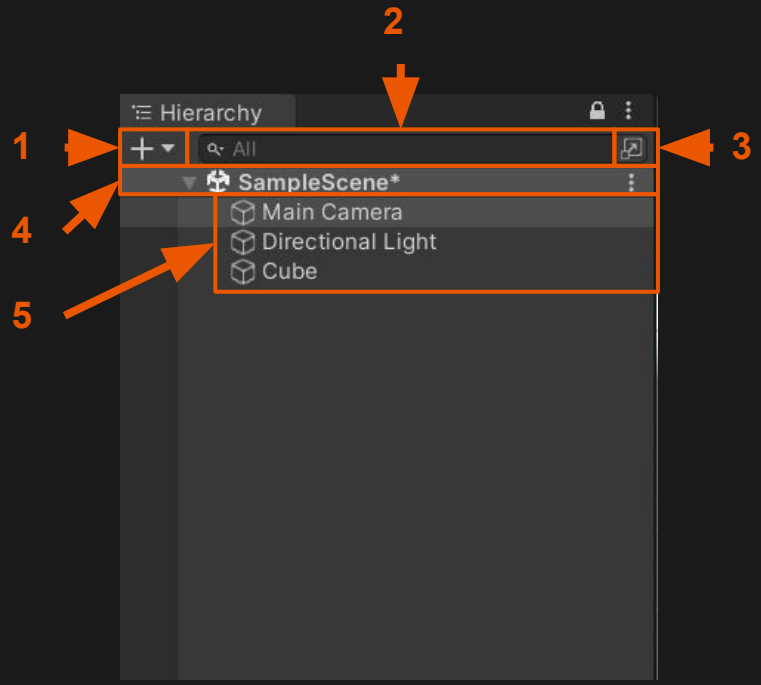
Hierarchy View

[4] Scene Header, this tells us which scene the objects are connected to. For the most part we will work with individual scenes, however in the future you may notice a new header popup when we work with data that's cross scene.

You may have noticed that there's a * next to the SampleScene, that means you've made changes that have not been saved.

Make sure to save your progress often as Unity doesn't have a built in auto-save function and if it crashes all your progress will be gone.

[5] Game Object List, shows you all of the Game Objects in the Scene.



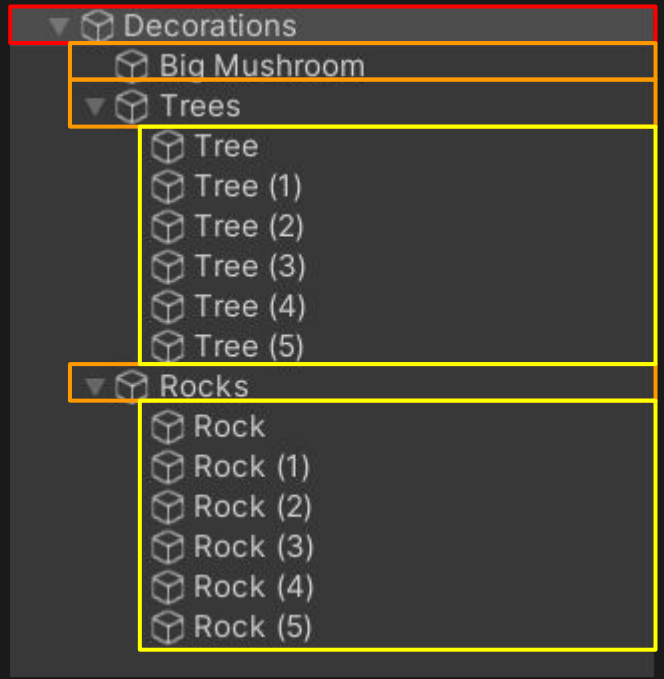
Nesting Game Objects

Nesting is the process of connecting Game Objects to other Game Objects. You can achieve this by dragging one object on top of another. You will know that the Object is nested as it will indent below it's parent.

The Game Object that we connected to is called the Parent and the Object that got connected is called the Child. So in this case Big Mushroom is a child of Decorations. Similarly Trees is a parent and has children Tree, Tree (1) … Tree (5).

Tree, Tree (1) … Tree (5), aren't direct children of the base of this Nesting, so they are called Descendants of Decorations.

Bring back World vs Local, when you nest a Game Object the parent object now becomes the world Origin point for it rather then the standard scene view one.



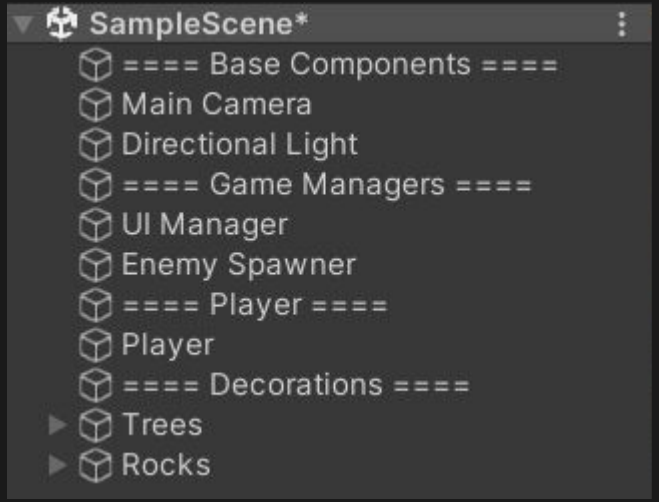
Keeping Hierarchy Organized

Beyond just just nesting a good practice is to put up dividers. These are just empty objects with some kind of symbol to make it easier for the eye to distinguish. It's not necessary for you to do it but your future self will thank you when you can find that one game object at a glance.

==== Divider ====

**** Divider ****

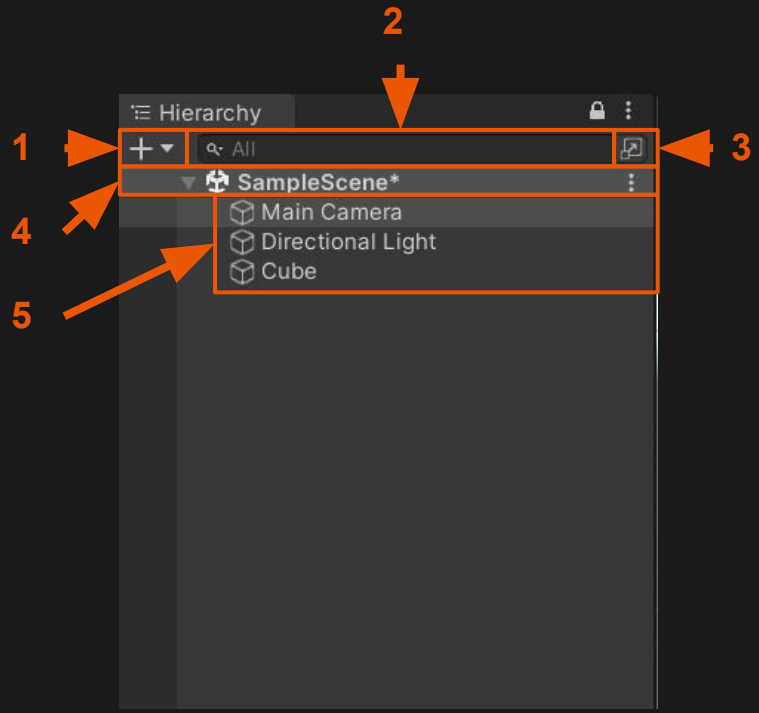
---- Divider ----



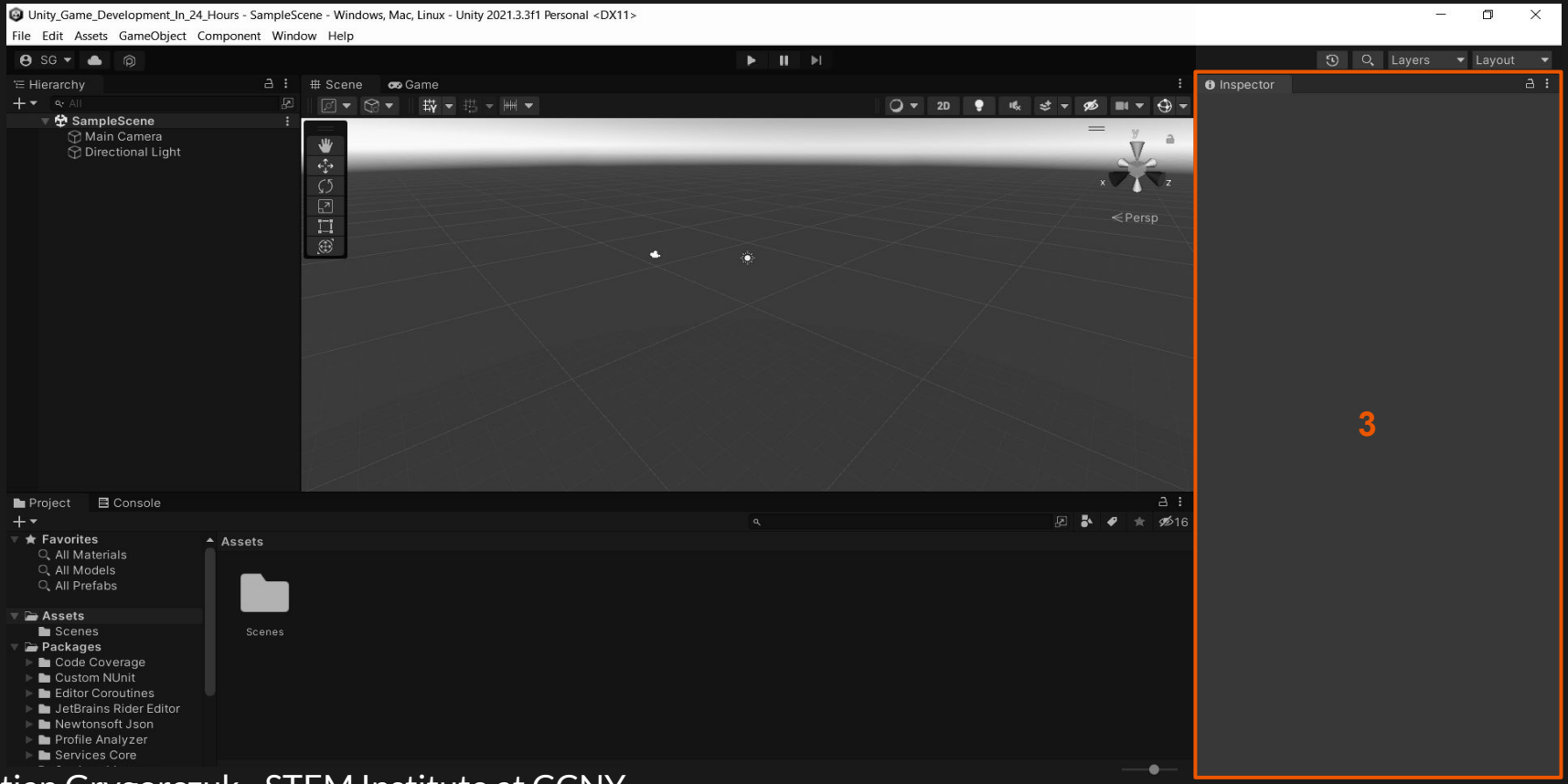
Finding Game Object in Scene View

By double clicking on any of the Game Object the Scene View will automatically zoom in to have to object in view.

Additionally, you can use the Hotkey F while hovering your mouse over the Scene View to zoom in on the Game Object.



Inspector


Unity_Game_Development_In_24_Hours - SampleScene - Windows, Mac, Linux - Unity 2021.3.3f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
Scene
Game
Persp
Inspector
3
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Packages
Code Coverage
Custom NUnit
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core

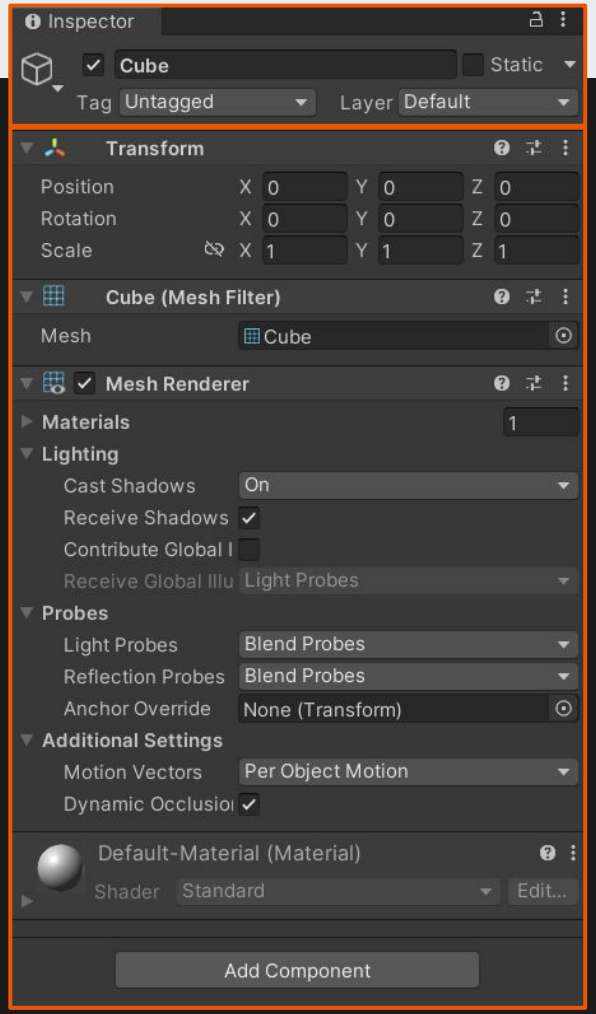
Inspector View

The Inspector View shows all of the properties of the selected object. In this instance it's a Cube.

With this we can see the the:

[1] Inspector View Overhead

[2] Game Objects Component List.

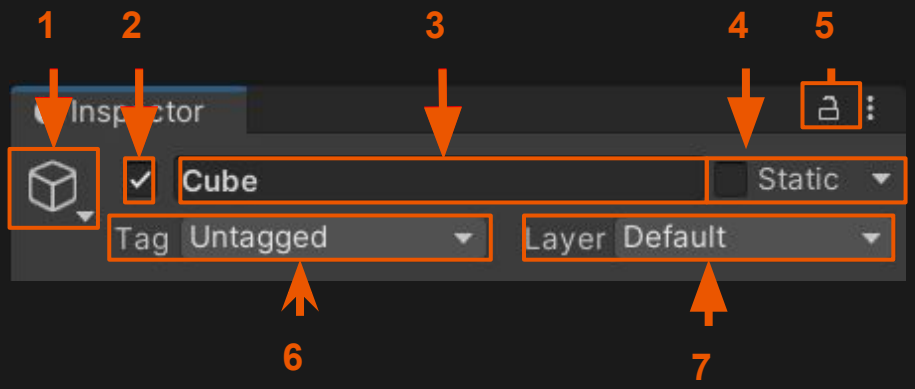


Additional Resources: Pages 13-14 of the Textbook, The Inspector Window - Unity Official Tutorials

Inspector View Header

There's a lot of thing here but don't worry most of them are very simple and very useful to us.

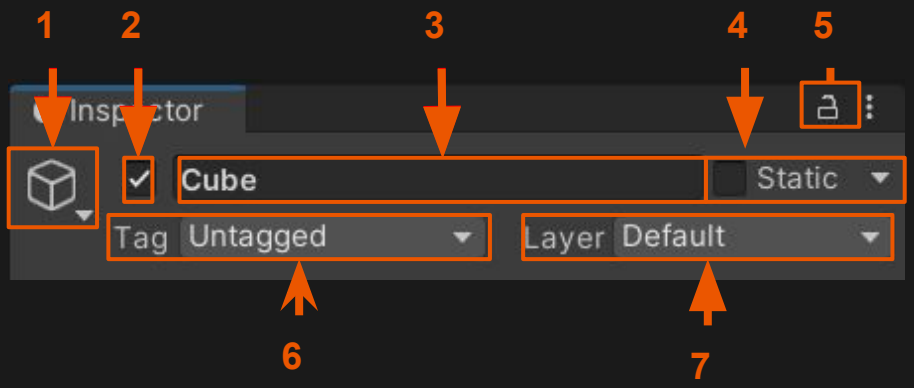
[1] Icon, all game objects can have an icon, but start without one; that's what the empty cube means. Given that game objects don't necessarily have a visual thing to represent them an Icon can be attached to find. These are mostly used for things like Game Managers and Spawn Points to help you visualize the scene better.



Inspector View Header

[2] Enable, the check mark represents that the Game Object is enabled, meaning it's active in the scene. If the toggle is uncheck that means the Game Object will not appear in the scene and interact with the level.

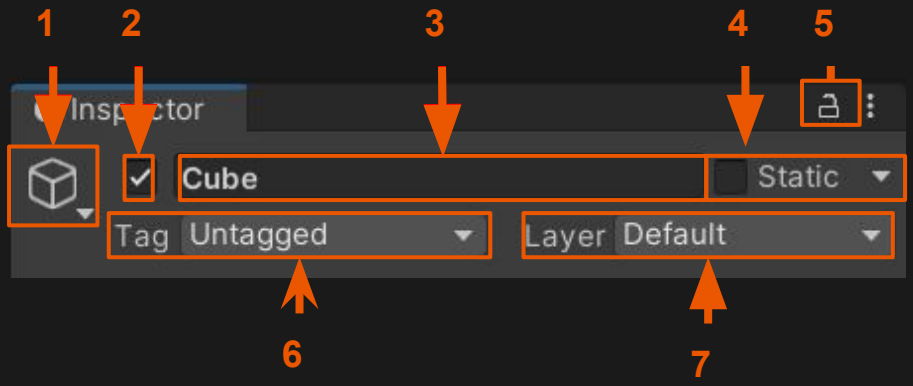
There are many reasons to turn of the Game Objects such a performance; if the physics and Rendering are turned off the computer will have an easier time calculating everything else happening in the game.



[3] Name, you can edit the name of object from here and it will be directly change in the Hierarchy View. Keep the names simple and direct to the point so that you can find it later when editing the Scene.

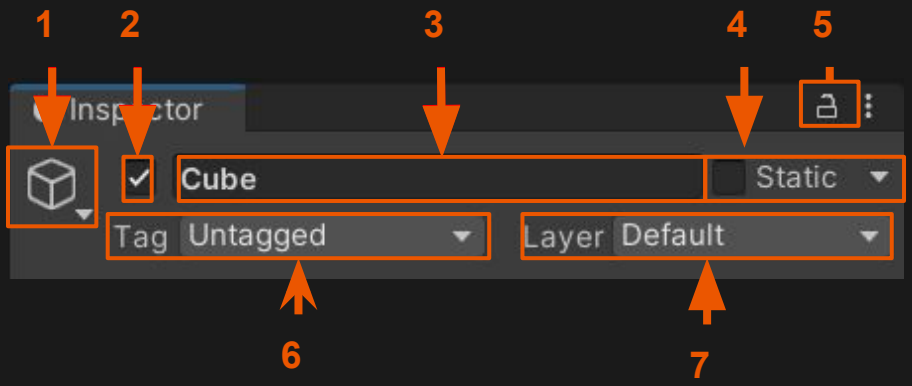
Inspector View Header

[4] Static, means the Game Object should be ignored by built in Unity systems such as lighting or navigation. You can set it to all or toggle specific systems you want to be ignore. We'll put a pause on this tool as it's not necessary for anything we'll be doing.



[5] Lock, this will lock the Inspector to the Game Object you are currently viewing. This is useful in situations where you might be working two or more objects and in so would end up having the Inspector switch between them but you want consistently edit the property on a light for example.

Inspector View Header



These two allow us to distinguish what the Object is and how it should interact with the given environment.

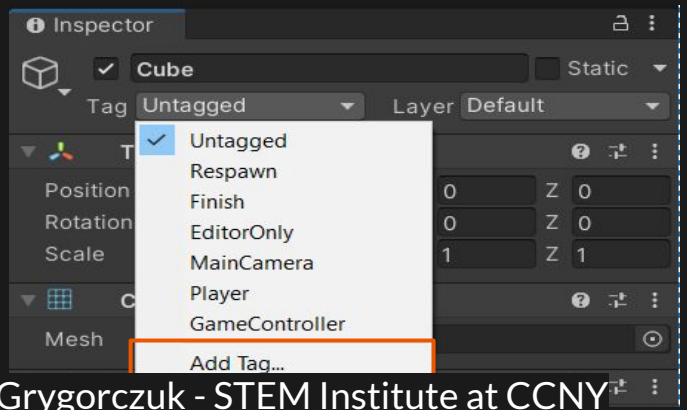
[6] Tag, using tags we can use our Scripts to identify game objects and apply wanted behaviors; very effective for checking collisions and applying behaviors if any are required.

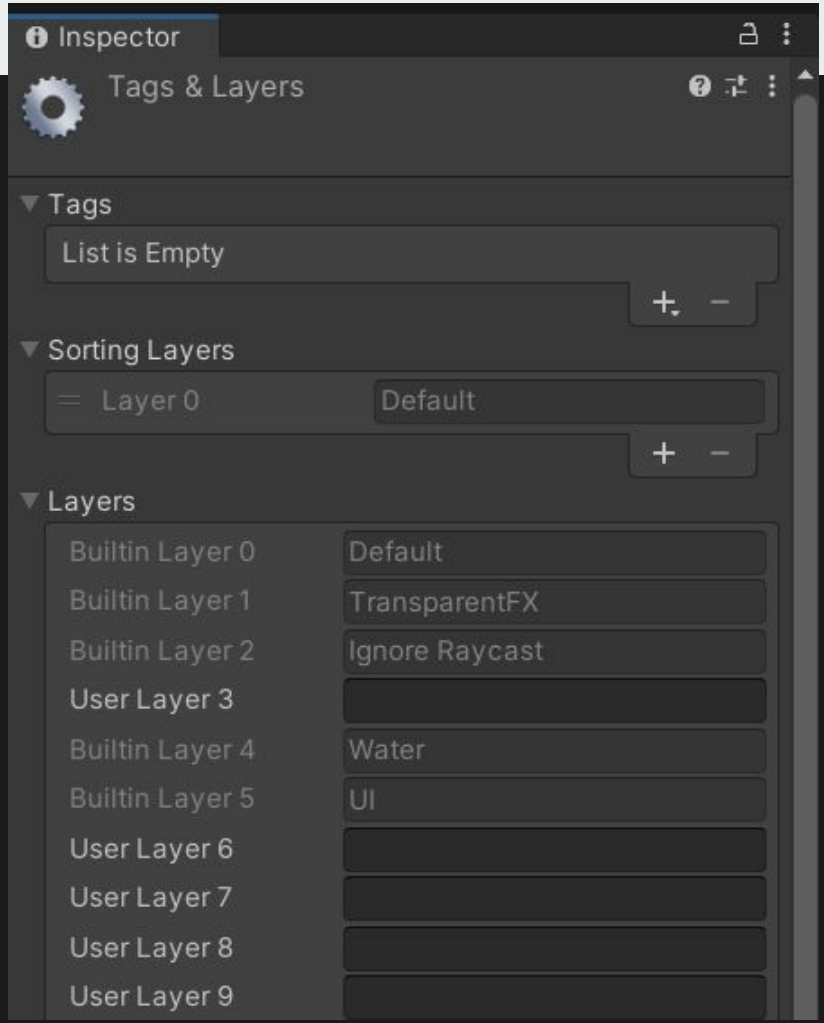
[7] Layer, are similar to Tags but rather than identifying Game Object for our script it's used for built in Unity tools such as lighting or navigation.

Inspector View Header

It’s important to note that even though both Tags and Layers come with a predefined lists you can add your own categories.

You can access this list by clicking the drop down menus on the Tag or Layer drop list and selecting “Add Tag” or “Add Layer".



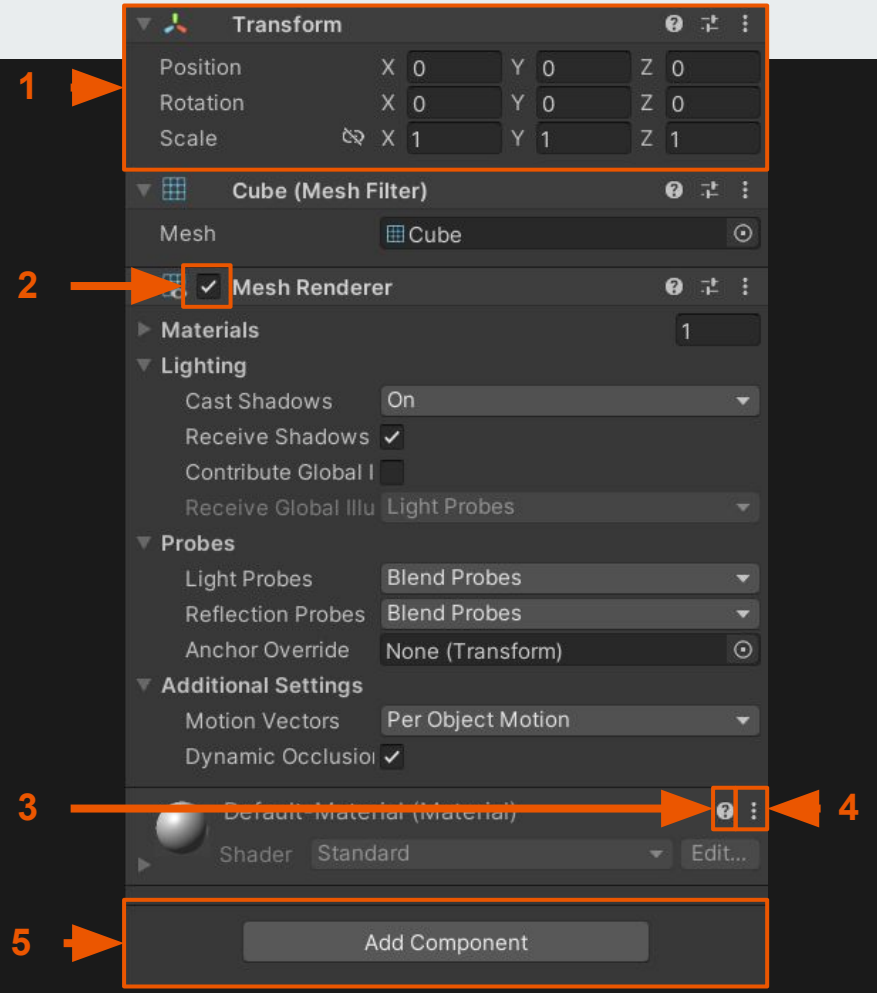


Components List

The Component List will show you everything that's attached to the Game Object, from Renders, to Lights, to your own Scripts.

As you see the Cube has three game objects. The Transform, Mesh Filter, and Mesh Rendered. It might look like The Material underneath Mesh Render is a Component but that's part of the Mesh Render.

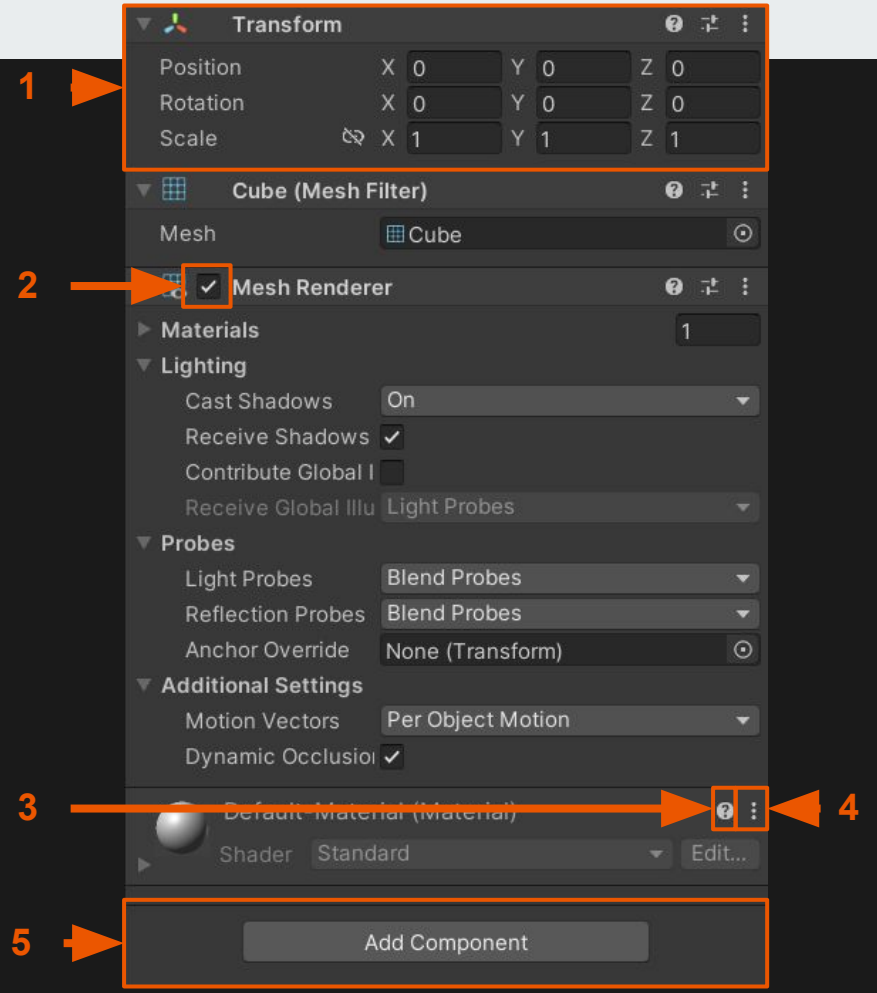
You can tell because all of the Components have a slight different color Header that holds a little arrow next to the name of the Component allowing you to collapse it if the properties aren't important to you at the moment.



Components List

[1] Transform, every Game Object will come with a transform. It defines where it is in the Scene, which way it's rotated and how large of a space it takes up. Every Transform is also 3D, even when building 2D games the Z axis will be available to edit.

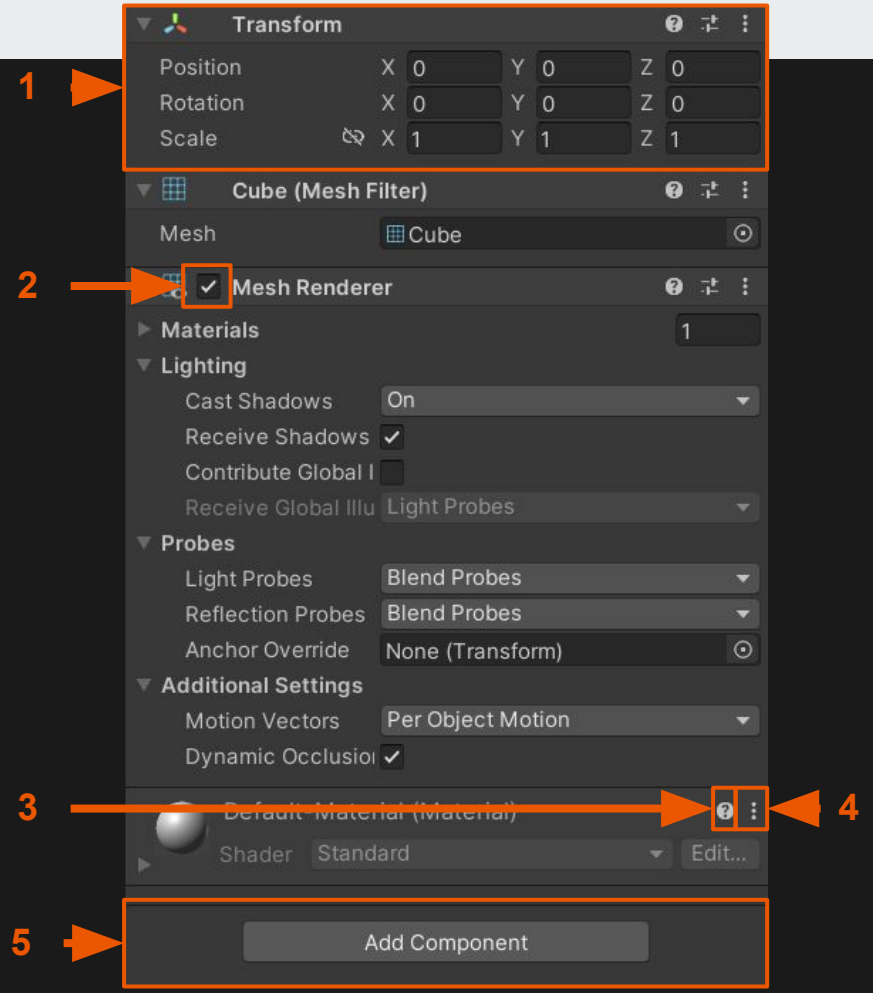
[2] Enable, notice that certain Components such as the Mesh Render have a similar Enable toggle like the Game Object has, this could allow you to keep the Render of a Game Object but turn of it's Collision, such that you can see it but nothing will touch it.



Components List

[3] Information Link, clicking this will bring you to the Web Unity Manual page for the given Component.

[4] Component Properties, allows you to Reset the Component to default state, allows you to move the Component up or down the list [Also possible by dragging the Component Header], Copy and Paste the values and altogether delete the Component from the Game Object.

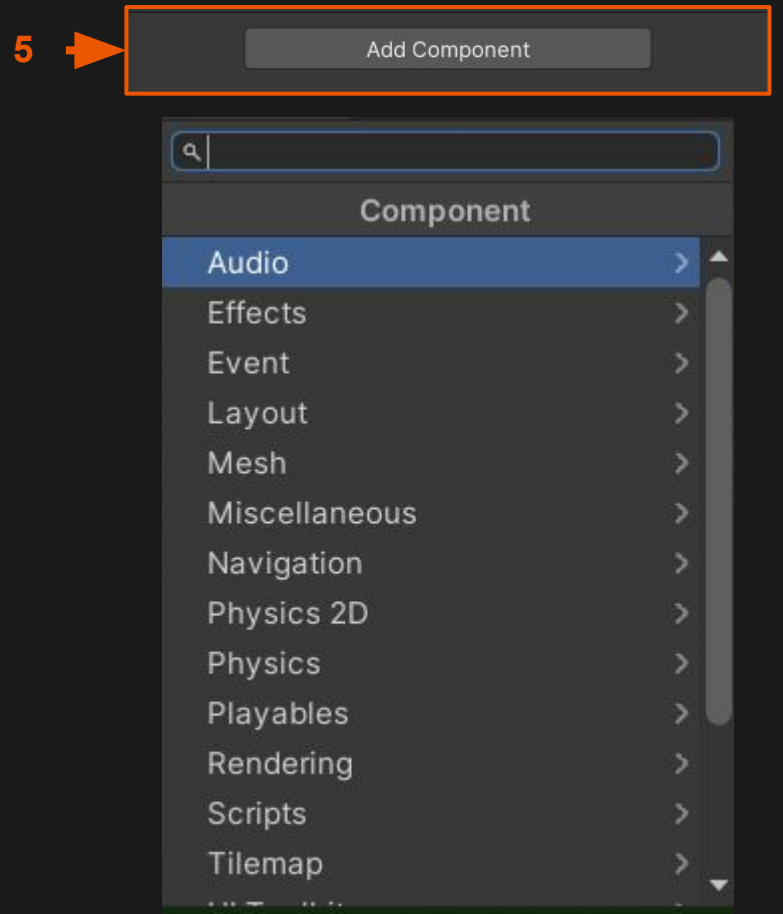


Components List

[5] Add Component, this will be a highly used feature. It allows you to bring in any Component that is pre built by Unity or any custom ones you may create such as Scripts. You can also add Components by dragging them from the Project View into the Component List or using the drop down menu from the File Menu Header named Component.

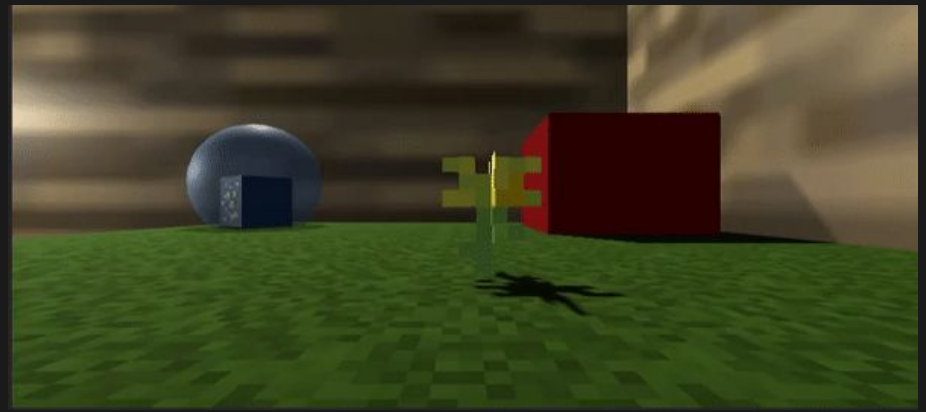
As you can see there are many categories, we will go over many of the Components held by them by not all.

Feel free to use the Information Link, to look up any we may miss through the course of this class.

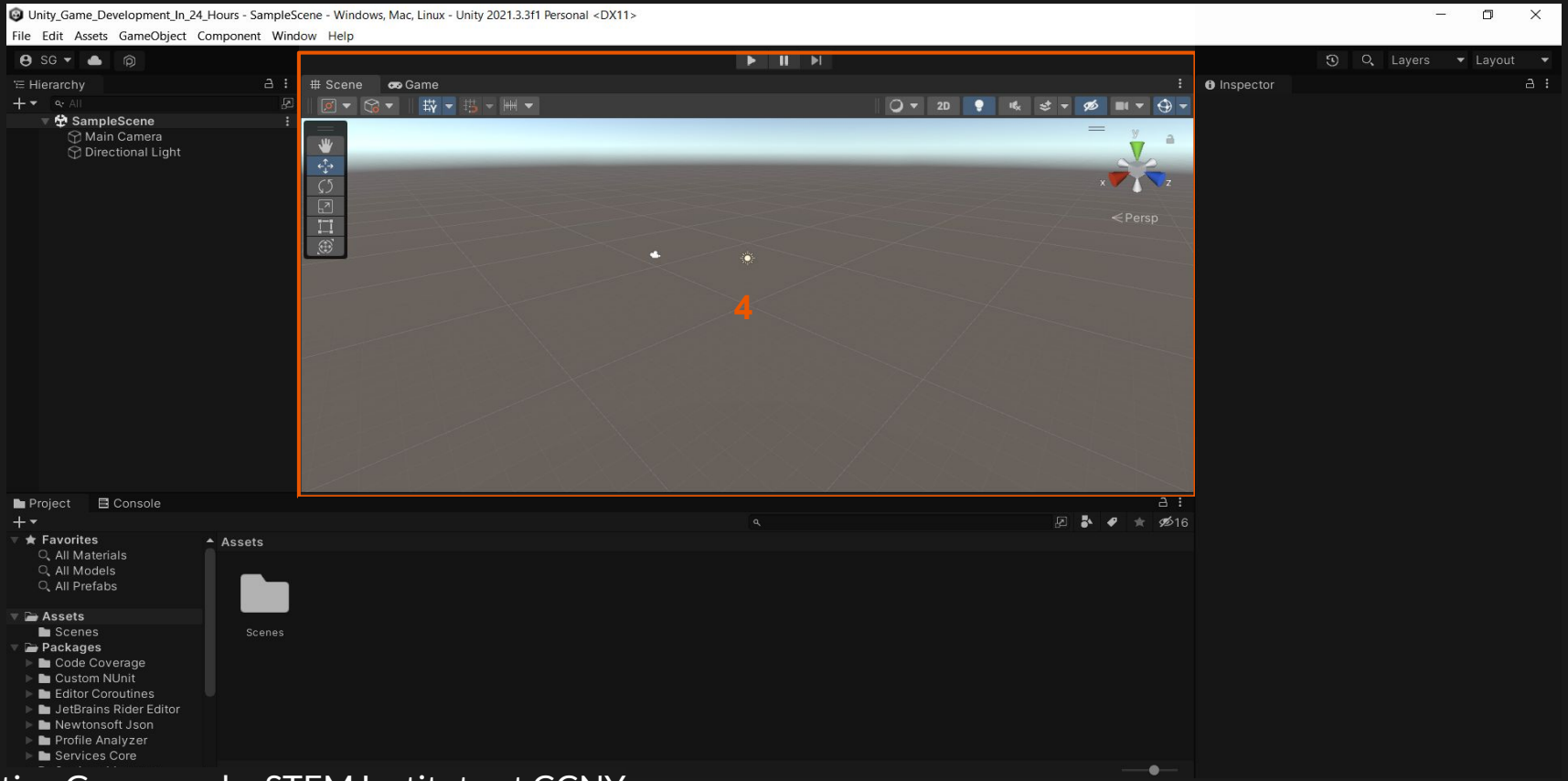


Game Object Challenge

1. Using the Hierarchy View, right click and create a new Cube Game Object
2. Notice that the Game Object has a Collision Component already on it. That allows it to touch other object that have Collision Components.
3. Using the Inspector View add the Rigidbody Component, [not Rigidbody 2D]. This gives the Game Object Physics.
4. Place the new Game Object over the Floor and click the Play Button on the top of the screen.
5. You should see your new made block fall



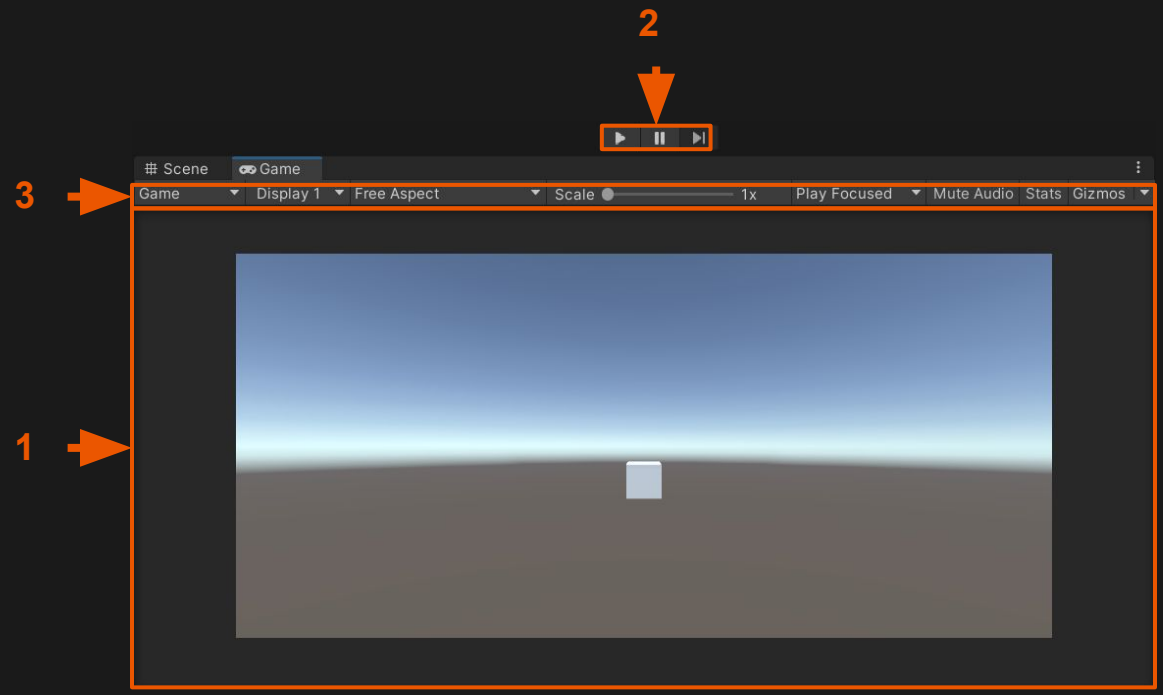
Game View


Unity_Game_Development_In_24_Hours - SampleScene - Windows, Mac, Linux - Unity 2021.3.3f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
Scene
Game
2D
Persp
4
Inspector
Layers
Layout
Project
Console
Favorites
All Materials
All Models
All Prefabs
Assets
Scenes
Packages
Code Coverage
Custom NUnit
Editor Coroutines
JetBrains Rider Editor
Newtonsoft Json
Profile Analyzer
Services Core

Game View

Game View shows you the final product, here you can play test the game and all of the mechanics that have been built into it.

[1] The Game View, here you see the game through the Camera Game object that was created alongside the project or a new one you may have predefined.



Additional Resources: Pages 16-17 of the Textbook, The Game View - Unity Official Tutorials

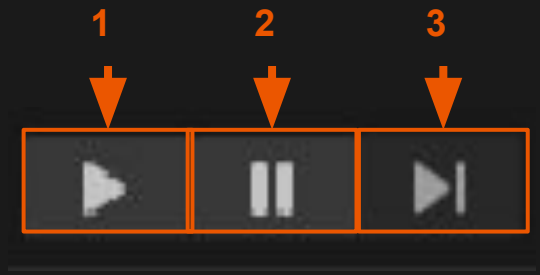
Play Bar

The Play Bar is always available to you, it consists of three buttons

[1] Toggles Play/Off, clicking the grayed out version the the play button will assemble the game and have it available for you to play. This can also be called Running or Executing the game.

Toggling the Play button Off, will reset everything to the way it was before you played. Any enemies you may have spawned will be gone and any progress you may have made in the level will be reset to the start.

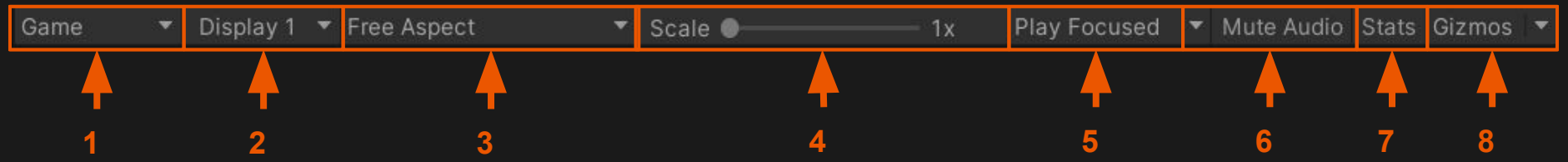
Also while in Play Mode you are free to change and move any of the Game Obejct through the Scene View or the Inspector, however be caution that the changes will revert to their original state once Play Mode is off.



[2] Pause, stops the execution of the game but keeps it at the progress you've made in the level. This is very useful when something isn't acting the way you intended so you can execute the game have it run for a bit, pause when the problem occurs and move to the Scene View to examine the game objects that are cause the problem.

[3] Step, similarly to pause is a great way to debug your game, each click progresses the game by a step so you can see what occurs every step or tick of time that passes in the game.

Game View Toolbar



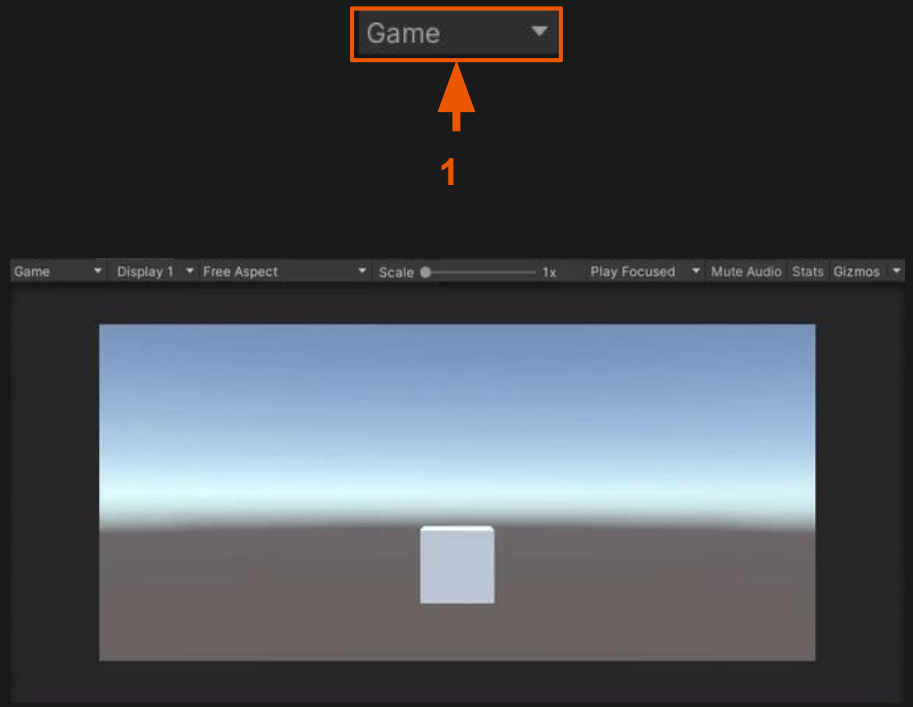
The Game View Toolbar will allow you to change the game is presented to you, allowing you to see what it would look like on different screen and resolutions.

The way you game is seen by you and the player is important as it will dictate how much and what you can show on the screen at once. Creating hard Game Design Decisions.

Game Drop Down

[1] The Game-Simulator Dropdown, will allow you to switch between a game view that is built for Desktop/Console vies to those available on the different mobile devices.

This also changes the available settings on the Toolbar, however we won't be covering Mobile Games in this class so it's good to know but not necessary for rest of the class.



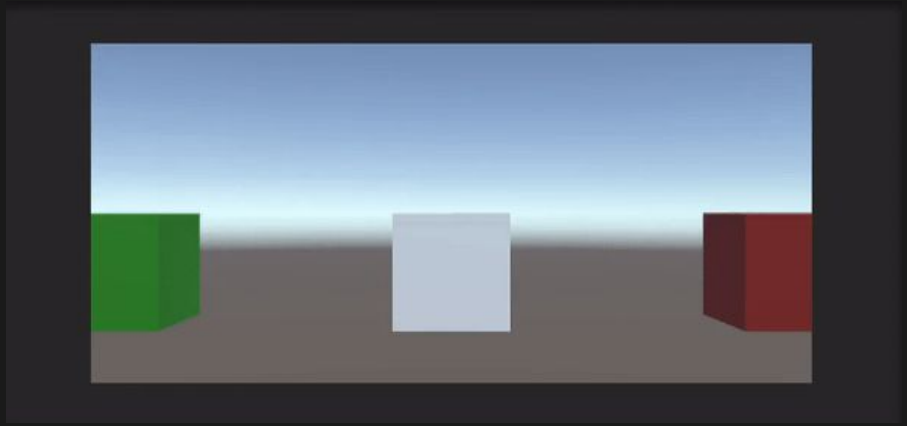
Display DropDown

[2] The Display Drop Down, allow you to set up to eight Displays or monitor that the game would run on.

Not to be confused by Split Screen or camera that would be positioned at a different character or location the game you can switch to.

This is for the games being connected to two or more monitors, for example the racing games at Arcades that have two to four player seats and are all connected each monitor following their own player.

We won't be working with feature.



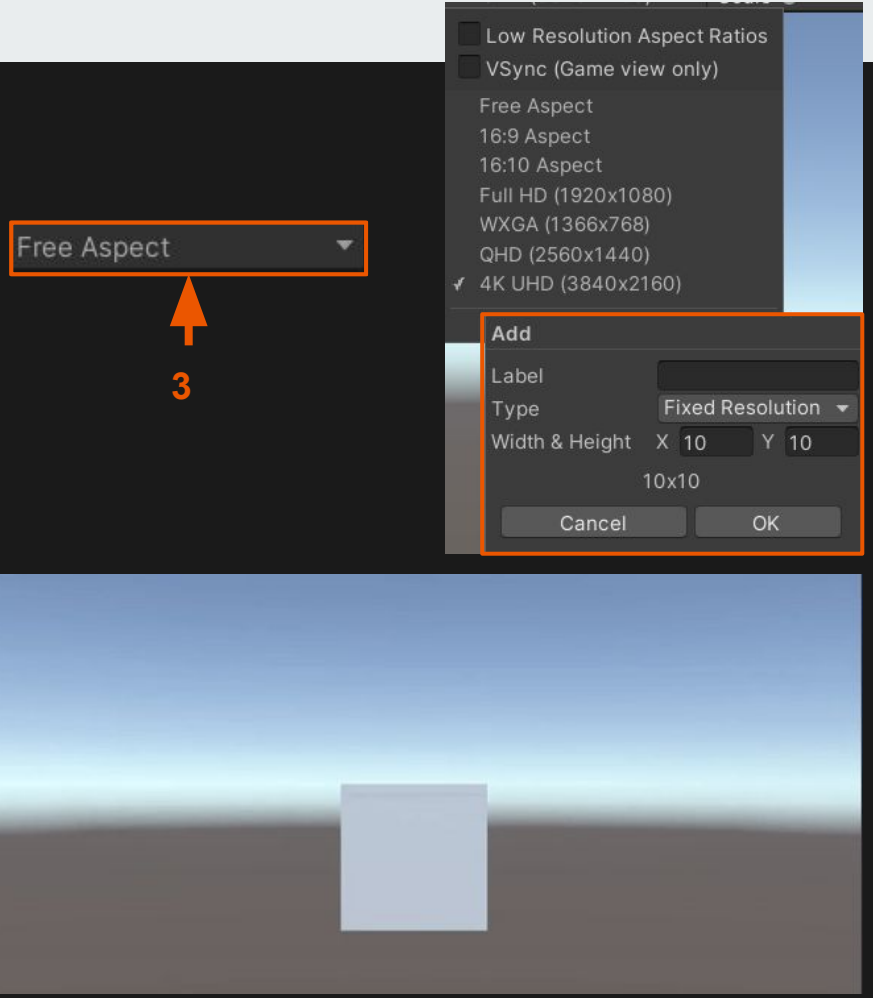
Aspect Dropdown

[3] Aspect Dropdown, allows you to set the Aspect Ratio at which you will be viewing the game through the Game View.

This does not set the Aspect at which the game will be exported out of Unity only how you are currently viewing it.

If you find the options lacking you have the ability create your own aspect ratio.

Depending on what type of game you want to make, you might want to use less of the screen setting up this aspect ratio will let you design your level in accordance with that.

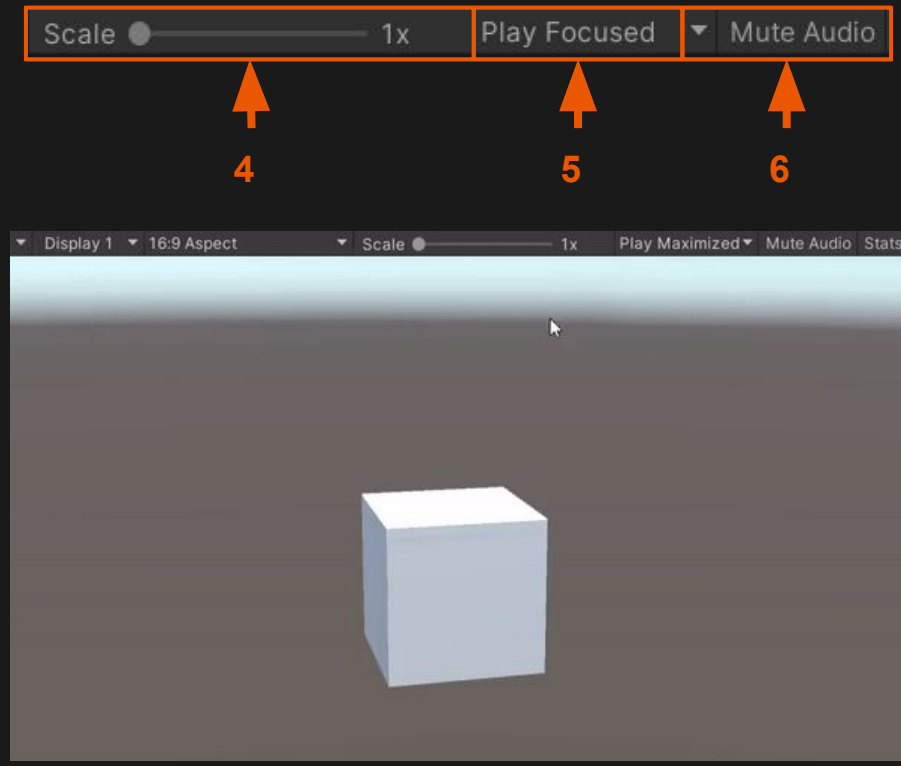


Scale, Play Focused / Maximized, Mute

[4] Scale, allows you to zoom in and out on the game screen. This isn't like navigating close to the Game Object in the Scene view but zooming in on the pixels of the current scene.

[5] Play Focused / Maximized, allows you to set the play setting to take up as much of the screen as possible while keeping the aspect ratio you've selected.

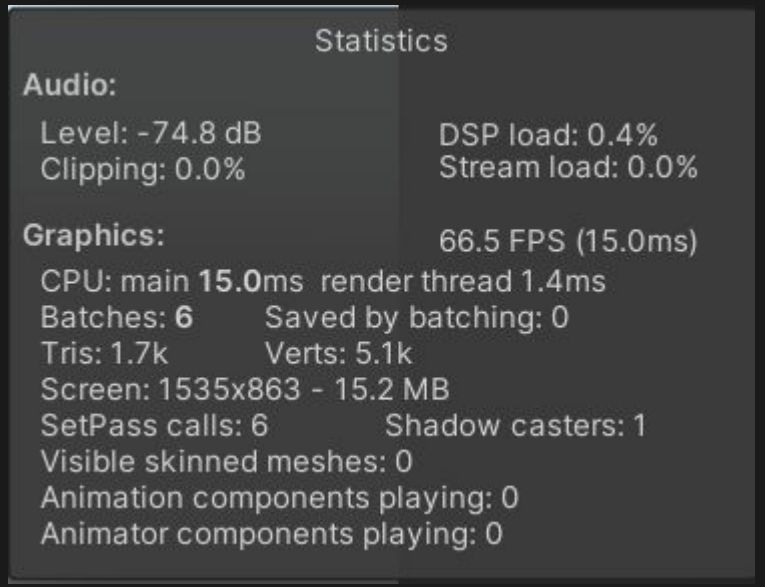
[6] Mute, pretty self explanatory, once you start testing your games for bugs you will be hearing the same sound effects and music over and over and it's good to turn it off when not necessary.



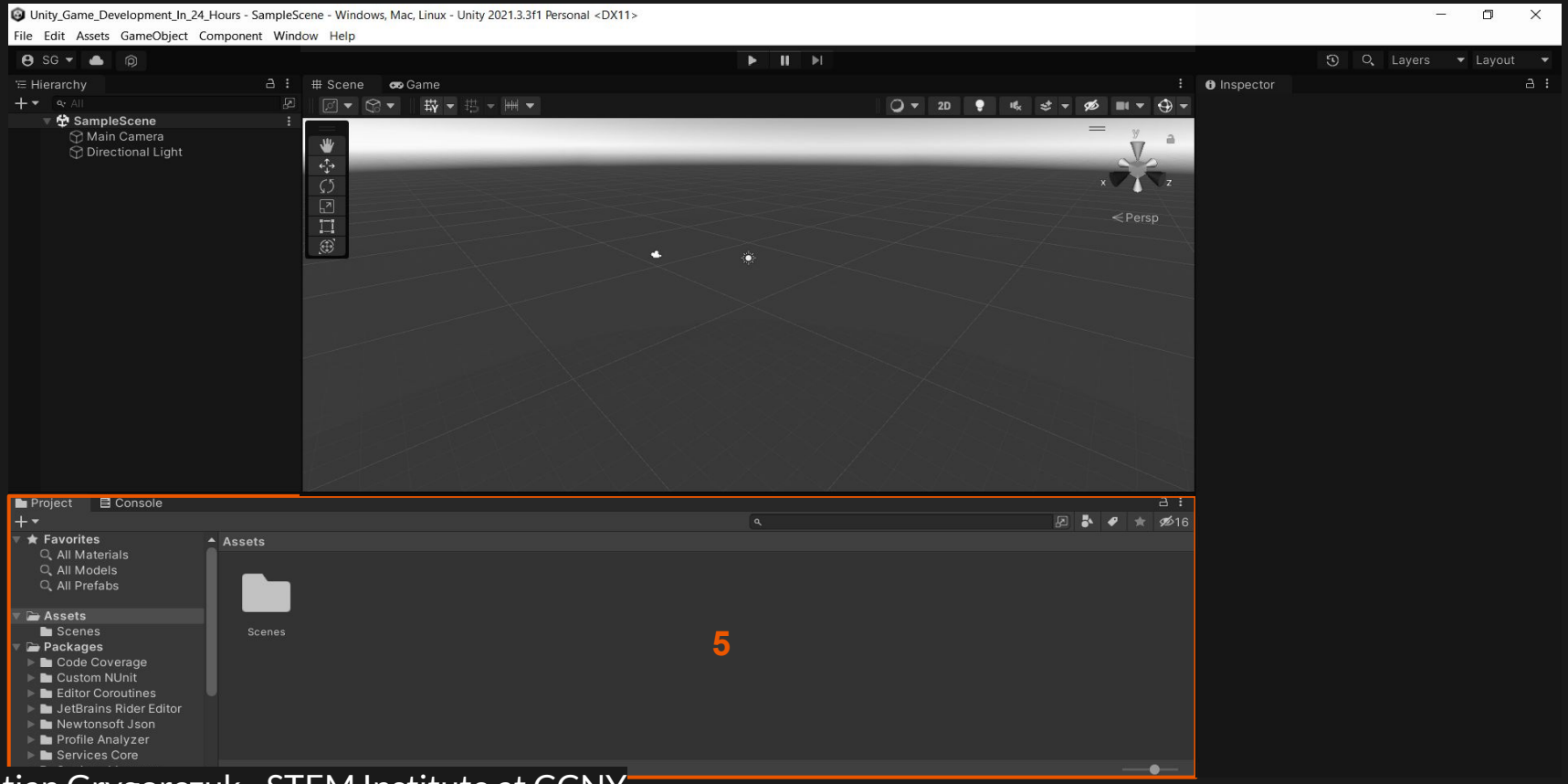
Stats and Gizmos

[7] Stats, a very useful tool telling us how the game is running. It's broken down into Audio and Graphical strain on your computer. With the scale of our projects you won't really have to look at this menu unless you've really messed up.

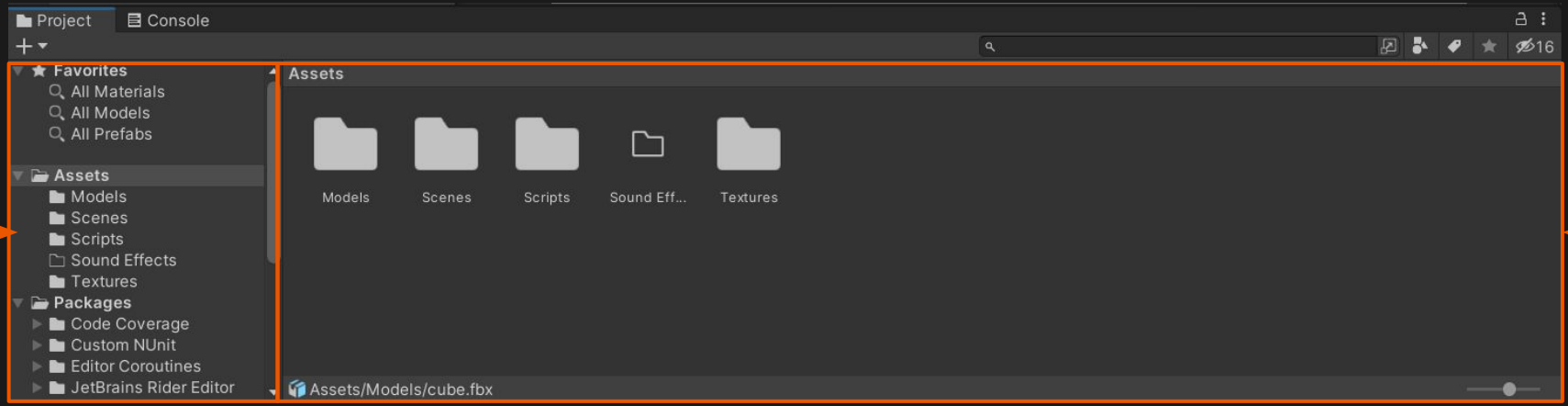
[8] Gizmos, work pretty much exactly like their counterpart in Scene View allowing you to see all Gizmos that are toggled on.



Project View



Project View



The Project View will display all of the Game Assets that are available to you, [1] shows you a list of all of the folders available to you, and your favorite searches. [2] Shows the current folder you are looking at.

Additional Resources: Pages 9-11 of the Textbook, The Project Window - Unity Official Tutorial

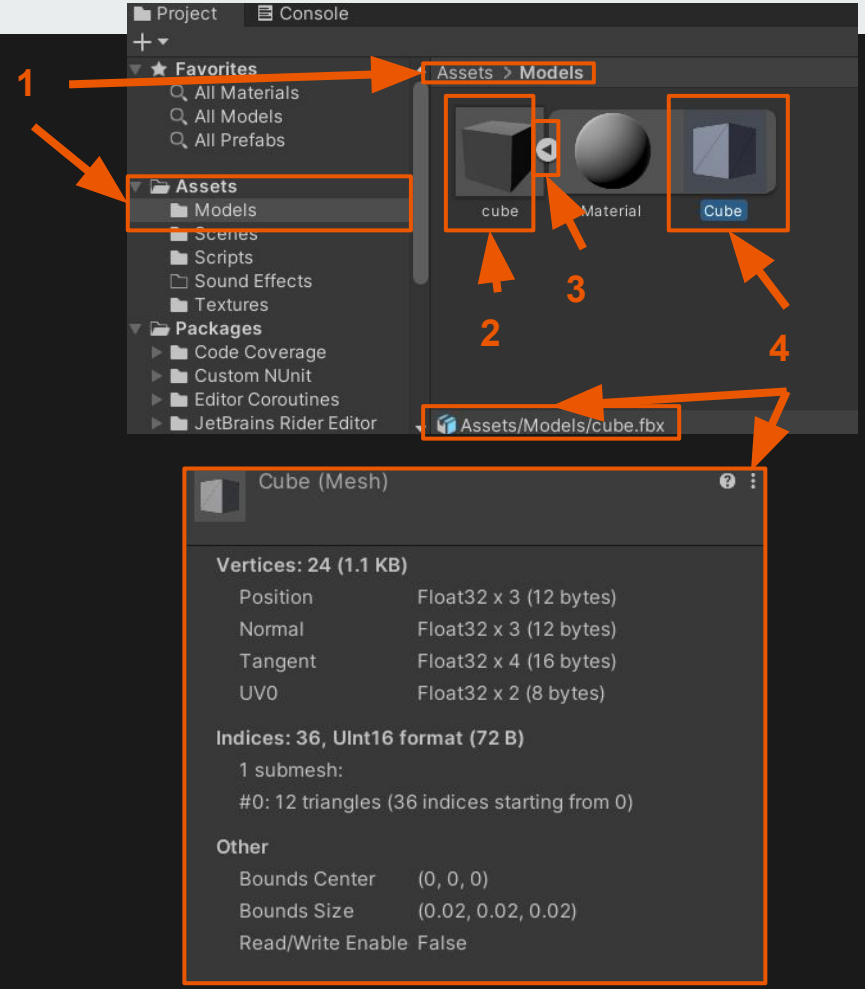
Game Asset Vs Game Object

A Game Asset is anything that has the potential to be Instantiated or created while Game Object is an instance of a Game Asset.

You can instantiate a Game Asset into a Game Object by dragging it from the Project View to the Scene View. Notice that instantly the Game Object is added to the Hierarchy and the Inspect View is filled with its Components.

Later on we will instantiate Game Objects using code.

Project View



When looking at Assets in the Project View you will have [1] The Breadcrumb Trail which is the path to the folder we're currently viewing. This will be also displayed in the Folder Column by highing the folder currently accessed.

Next let's look at a [2] Game Asset, here I have a 3D imported cube. More complex Assets such as 3D Models will have hierarchy of sub assets, you can click the [3] Arrow to expand or collapse them.

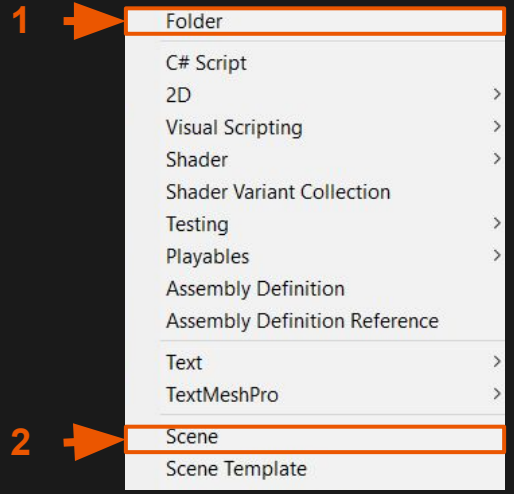
Finally you can see one of the sub assets for the cube which is it's [4] 3D model, when the asset or sub asset is selected you will see the Path to the asset on the bottom of the Project View and extra details will be shown in the Inspector View.

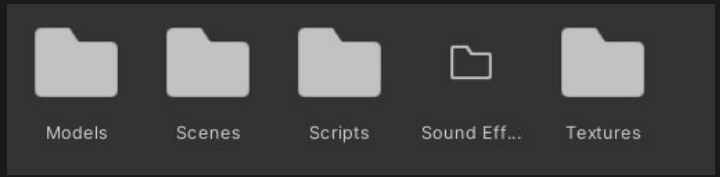
Creating Asset and Organization

There are many Game Assets that you can create directly from the Project View by right clicking on the empty space and selecting Create. This will provide you with a massive list but the main two we are going to focus on for a moment are Folder and Scene.

[1] Folder allows you to create new Folders in the Project View. It's not exciting but just like keeping the Hierarchy View organized is necessary for good game development same goes for the Project View.

[2] Scene Asset allows you to create a new scene or level. Everything we've been doing has been within one scene with this you'll be able to make many level with many scene and later combine them to make a game world.





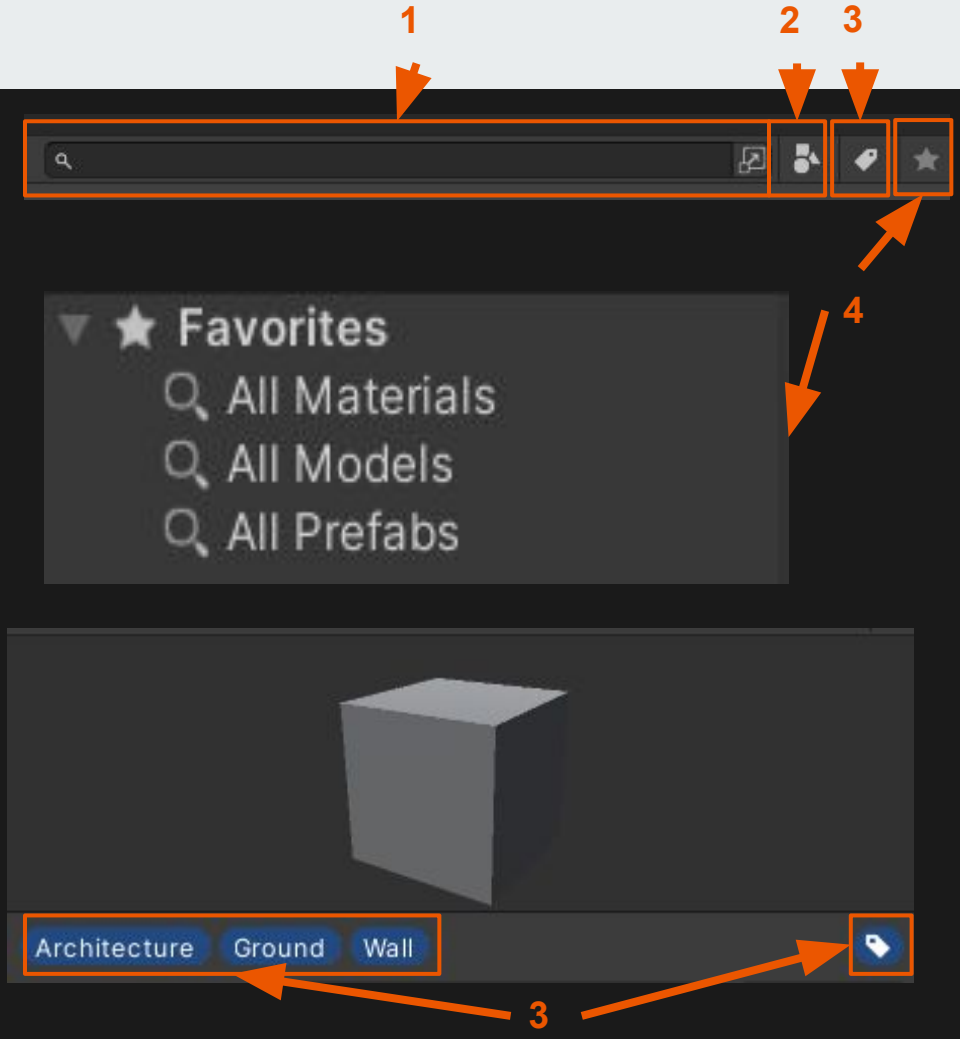
Search & Organization

Finding Assets quickly can shave hours off work. Thus we can use the [1] Search Bar and the Advanced Search just like the one in Hierarchy View. With the next three buttons working as filters for the inputted search.

[2] Allows you to view all items that fit into selected category, Models, Scripts etc. The Project view will look for items that have the correct extensions.

[3] Tags are things you can connect to specific Game Assets through the Inspector View, on the very bottom there will be a tag symbol with a list of descriptors, you can set as many as you'd like and then search by them.

[4] Favorite Search, when you search for a name in Search Bar you can favorite that search and it will pop up as a saved search in the Folder Column.



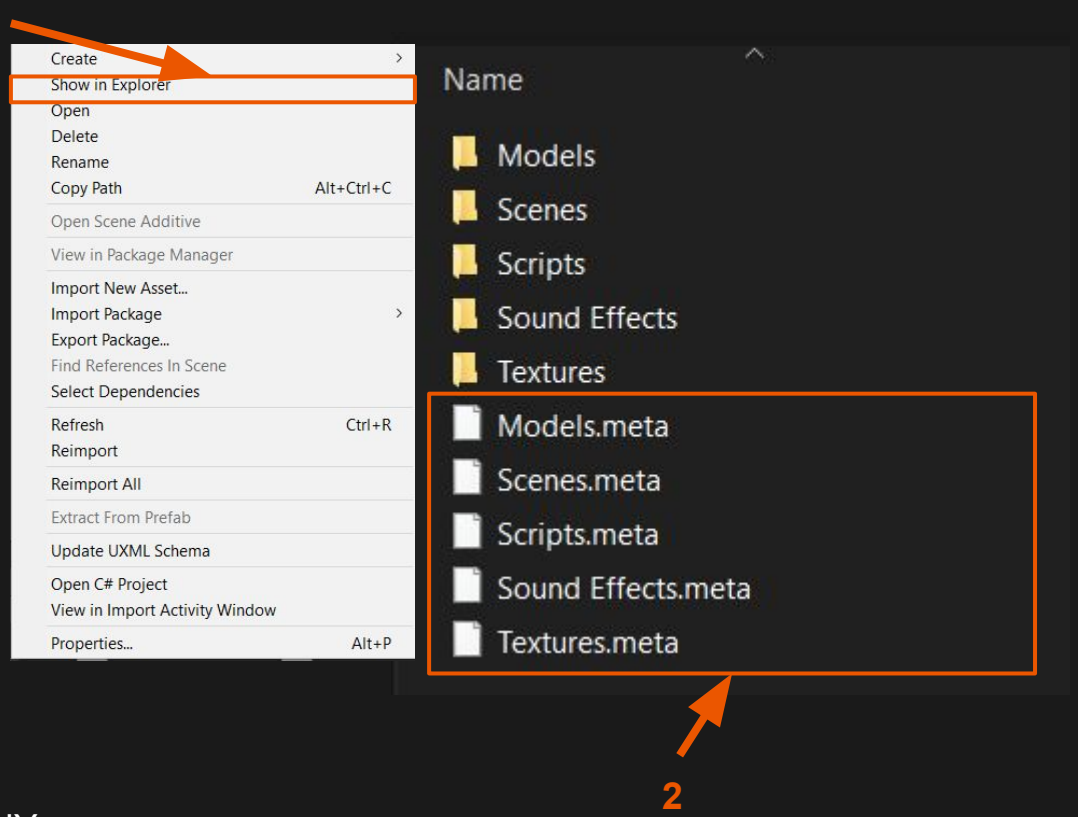
Sebastian Grygorczuk - STEM Institute at CCNY

Files Connect Directly to Project Folder

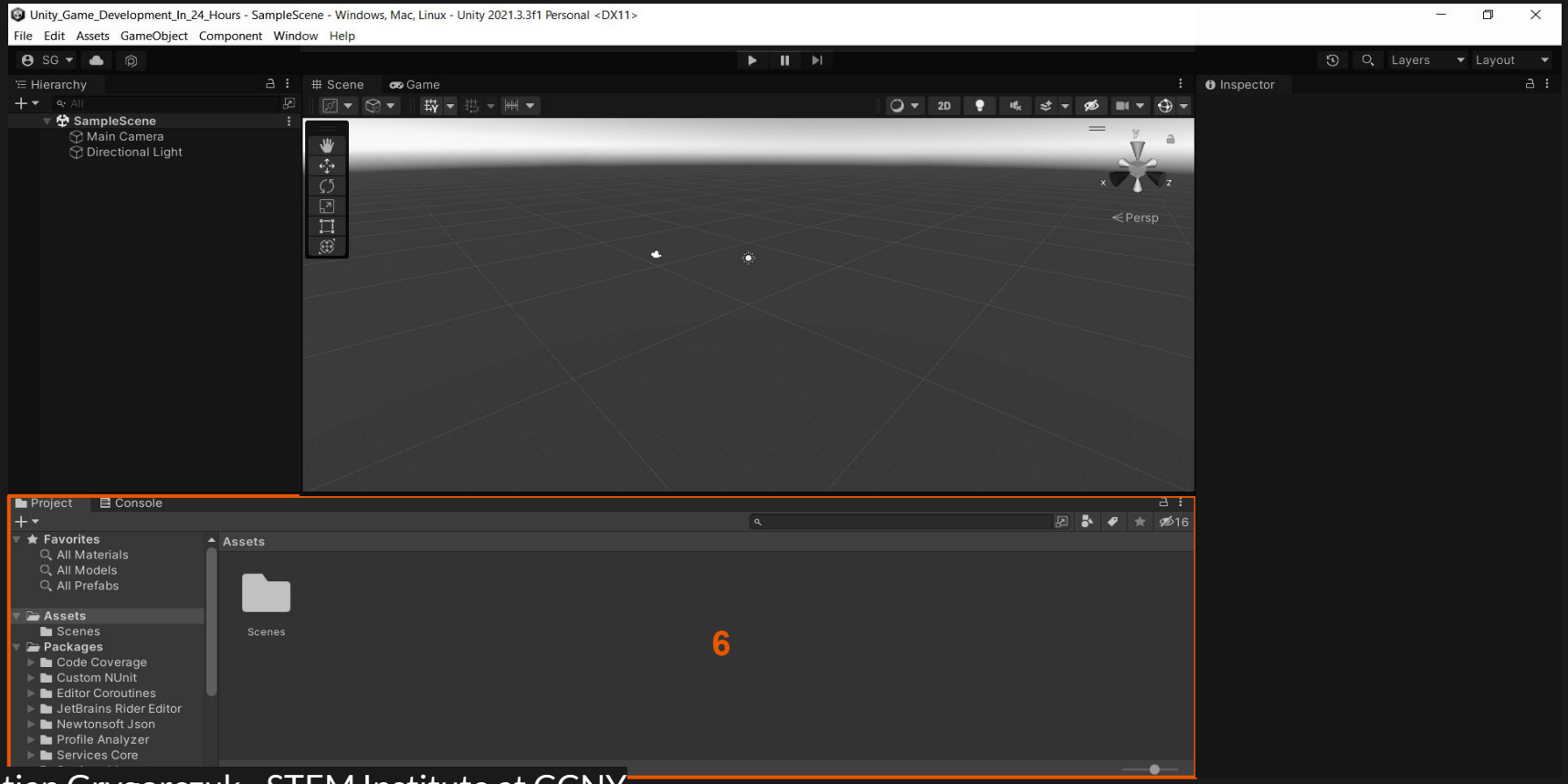
Each folder that's displayed in the Project View is directly connected to a folder in your project files. You can right click any folder and select [1] Show In Explore to open that Window Directly.

You may notice that inside the folder there are [2] meta files. These are files created by Unity that connect the assets to all it's functions in the game. Unity will automatically update them if you move or edit the file in any way.

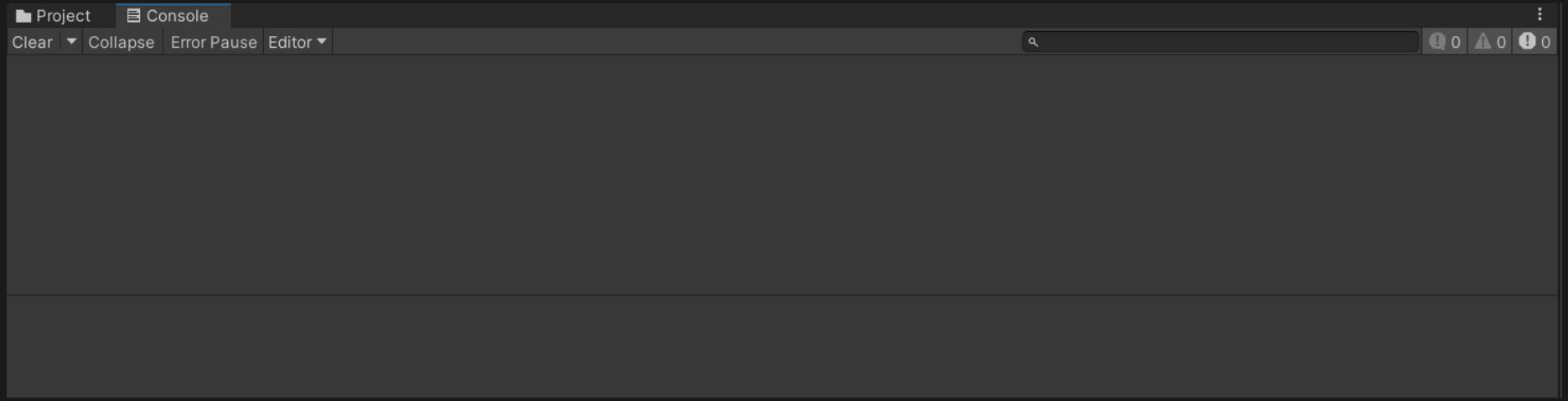
Only edit your files in the Project View as Unity may not update whatever changes you've done to a asset in the widow explore.



Console View



Console



Lastly there's the console, this will be heavily used once we start programming our game. From here you'll be able to print out states of object and look for bugs. We will return to this window in Week 2 once we being programing.

Additional Resources: Page 126

Layouts

One last thing that's important to mention is that every one of the views we've looked at can be moved and placed anywhere in the Unity program. There are many pre made but you also have option to create your own and save them for future use.

As we continue in the class we will introduce more views and depending on the task a new layout may be in order.