Welcome to Game Design and Development

Sebastian Grygorczuk - STEM Institute at CCNY

This Week's Agenda

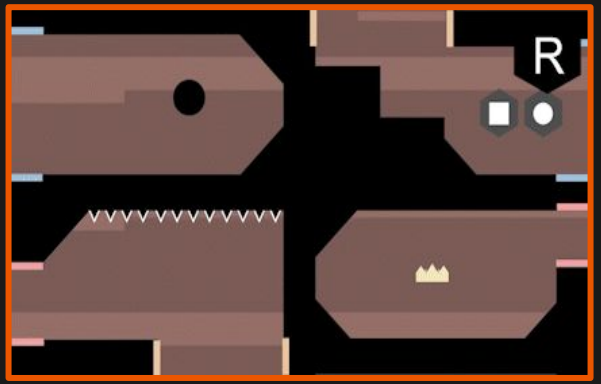
Getting Prepared for the Semester

- 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.

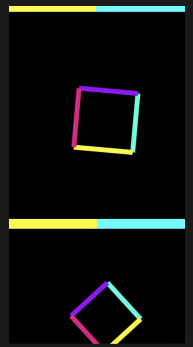
My interests outside of gaming are robotics, hiking and history.

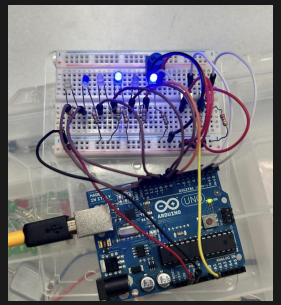


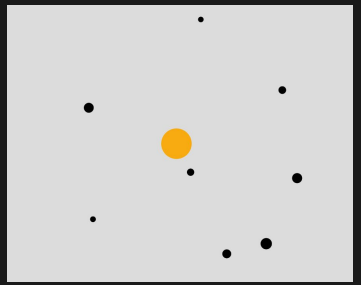
Who are we?

Adita Das

- Currently studying Mechanical Engineering at Columbia University SEAS
- First time as a TA
- I've made a version of color switch on p5js! (degraded lol)
- Been dancing since I was 2! Professional dancer for five years!
- I love anime and I caught up to One Piece after 1 and half of years of consistently watching





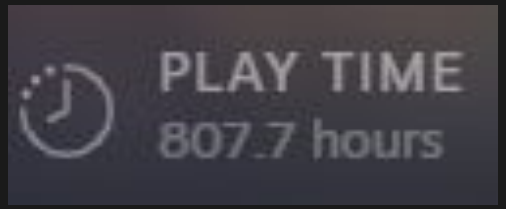


Sebastian Grygorczuk - STEM Institute at CCNY

Who are we?

Carlos Rodriguez

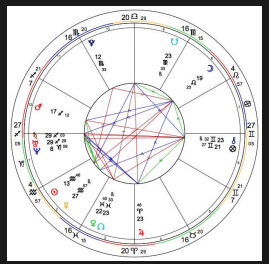
- Currently studying computer science at CCNY
- First time as a TA
- Wrote an entire history book on the game I plan on developing once I refine my skills
- Used to compete professionally in Super Smash Bros. Melee and Ultimate
- Really bad at pixel art



Who are we?

Katerine Arias

- I studied Multimedia Journalism and Theatre at Lehman College
- I am not too familiar with gaming but I'm excited to learn more about it, here
- My favorite game growing up was Mario Bros
- I'm a big fan of astrology and know how to read birth charts
- I love writing and going on road trips



Sebastian Grygorczuk - STEM Institute at CCNY

Introductions

Sebastian's Favorite Game

Carlos' Favorite Game

Adita's Favorite Game

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.

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/STEM-At-CCNY-Fall-2023-Semester

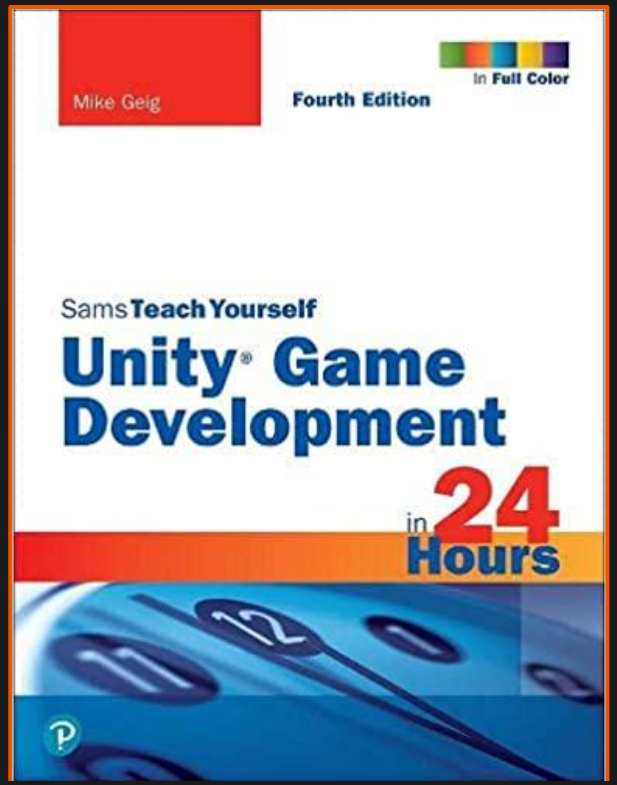
Computer Requirements

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

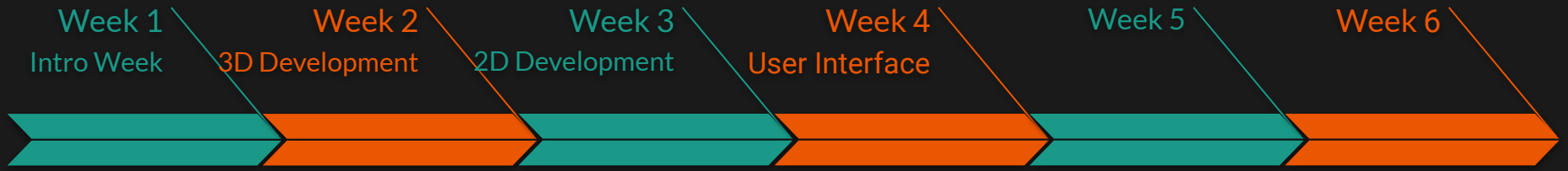
Unity Game Development in 24 Hours 4th Edition is an ideal textbook for this class because it's an introduction to Unity and many of its designer focused tools with programming included, but not programmer focused.

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

3D

Learning how to use Unity to create 3D games

2D

We'll have a 3D Lab and start learning how to create 2D games

User Interface

We'll learn how to create 2D user interfaces and have a 2D Lab

C# Fundamentals

Learning about C# programming language

C# Continued

Learn how to use C# to make game objects behave in the way you want them to and a programming Lab

Class Road Map

Animation, Lighting and Audio

We learn how to make the game feels responsives and have a Lab

Industry Guest

No class for Election Day and a Industry Guest Speaker

Flappy Bird

We build a Flappy Bird from beginning to end shows how to put together a fully functional game.

Game Project

No Class for Thanksgiving and starting on creating a Game Design Document.

Game Project

Working on the final project

Game Project

Finish the project for the class.

Week 13 - Final Week

December 12th we have a showcase of everyone's games and choose team(s) that will present at the closing ceremony. We will also talk about how you could apply the skills you learned past this class and what steps you can take if you wish to make games as a career.

December 14th we have a pizza party to celebrate your successes.

December 16th is the closing ceremony for the semester.

Attendance

Attendance

It is very important that students attend every class session to ensure that you are up to date with the topics covered in class. Attendance will be taken immediately at the beginning of each class, so please advise the STEM office of an unexpected absence.

As this is an after school class, lateness will be leanty. We are aware you are traveling a long distance.

Withdrawal

If you decide any of the classes you are taking aren't something you are interested in you have until August 27th to withdraw without it affecting your grade.

Withdrawing after August 27th, having 3 consecutive unexcused absences, or having a large sum of unexcused absences will result in a NCW, No Credit Withdrawn, which will affect your grades negatively.

Grading

STEM Institute works mostly on a Pass/Fail grading system, as long as you do the HWs and Project you will be fine. You're grade will be either

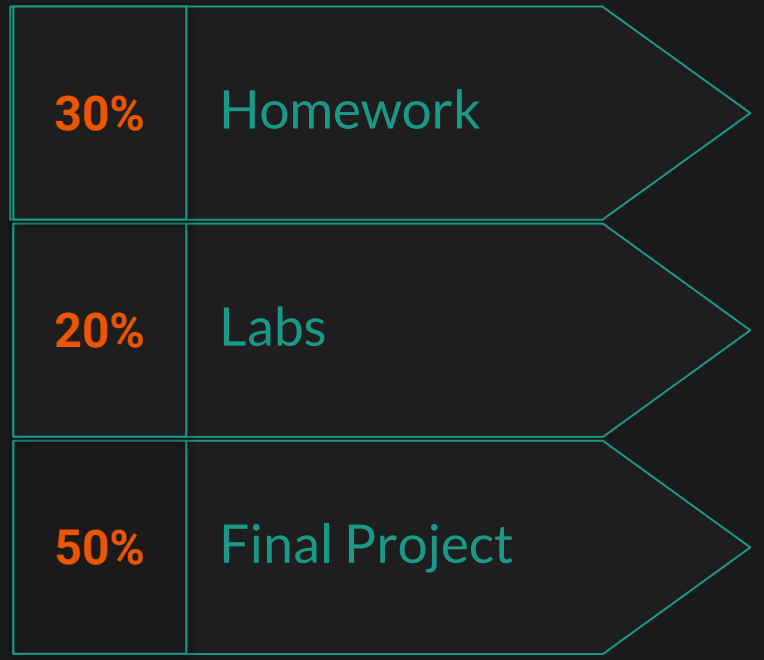
Fail - Below 65%

Pass - 65% - 89%

Numerical - 90% or above

If you rather not use the numerical grade let me know.

All Grades will be submitted on December 9th so if you have any problems with your grade let me know before then.

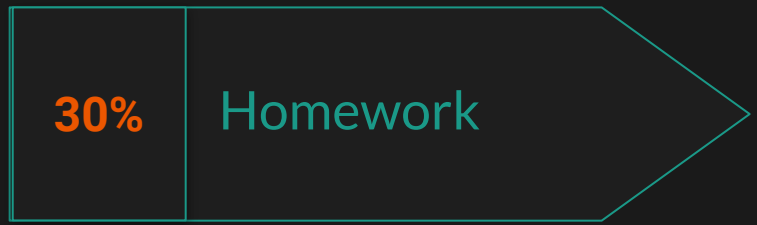


Homeworks

There will be one homework a week, broken down into two problems that we cover that week.

The homeworks will be practice
This class officially ends at 6PM with extension to 6:30PM for helps, you are free to work on the homework with our assistance.

We will have a total of seven homework assignments. After that we will focus on getting ready for the creation of the final project.



Lab

After finishing a topic we will have a lab that will test what you've learned by creating a game surrounding the concept we discussed.

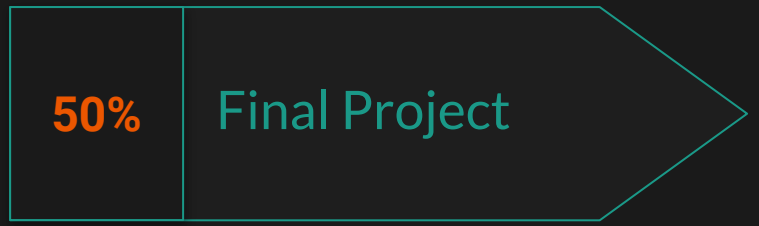
There will be a total of five labs, with the goal that you will have made five different practice games before the final project.

Final Project

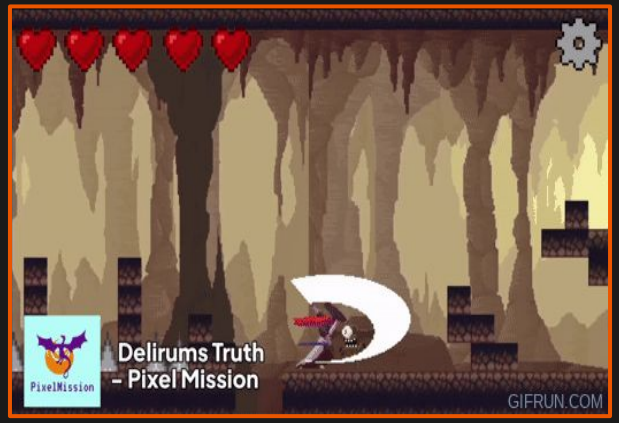
Final project will consist of you putting together all of the skills you've learned through the first four weeks and applying them to making your own game.

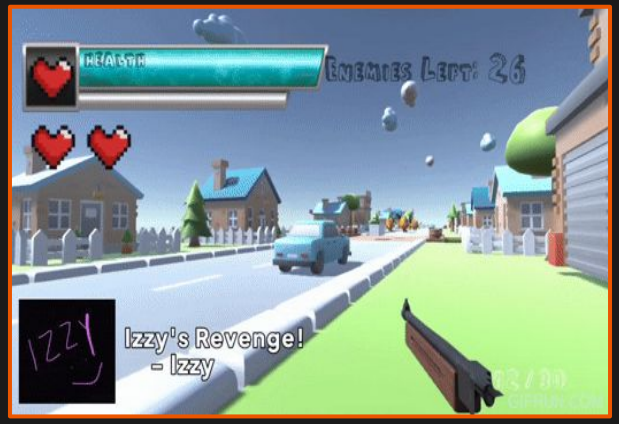
The project will be done in groups of 2-3 people as Game Development is a highly collaborative space.

The working game will require you to create a Start Screen that invites the player, one or more levels with a clear win and fail states that uses visual and sound effects to provide player feedback, and a Credits Screen that shows all the assets you've used and what each person worked on.



Last Classes Projects

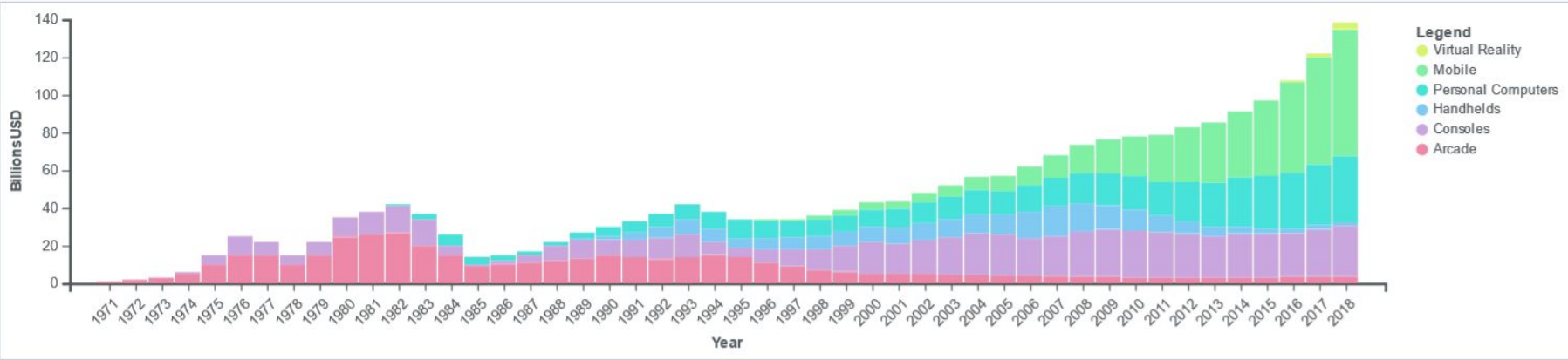




What is Game Development

"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

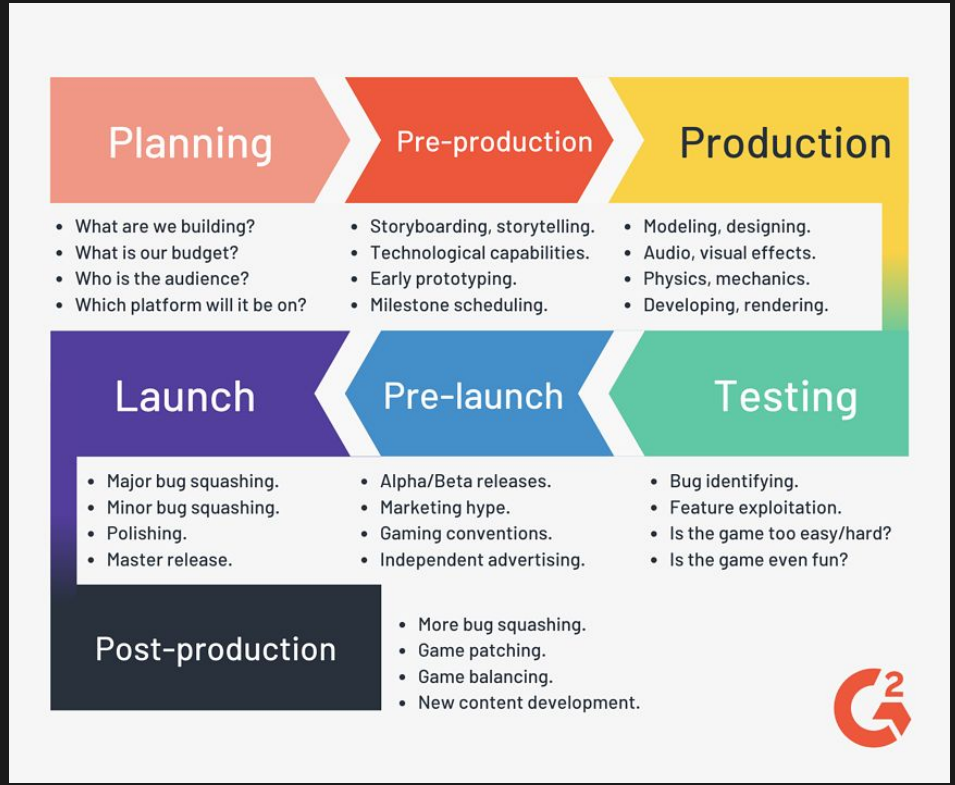
As we enter Week 4 and onwards of this class we will be following these stages of Game Development. We will mostly focus on the first four but we will talk about how to show off your game once it's done.

[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.

Challenge: Design

Think of any nebulous game ideas that you would like to implement in your game:

What is the theme you want to follow?

Is it a 3D game, if so is it over the shoulder, FPS, Isometric or something else? Or is it a 2D game, if so is it top down or side scrolling?

Does the game have a story? Are some plot points you'd like to include?

Feel free to compare it to games that exist.

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:

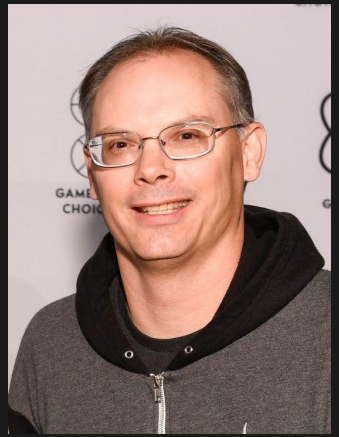
- Gameplay Programmer role
- Backend / Server / Online Programmer role
- AI Programmer role Graphics Programmer role
- UI Programmer role
- Animation Programmer role
- Physics Programmer role
- VFX Programmer role
- Audio Programmer role
- Tools Programmer role

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.

Challenge: Mechanics

Now it's time to get a bit more technical, think about the mechanics you want to include in your game.

Are you focused on platforming, puzzle solving, defeating enemies or something else?

What ways does the player have to interact with the world?

Are you keeping score, giving player level ups, or something else to reward them for playing well?

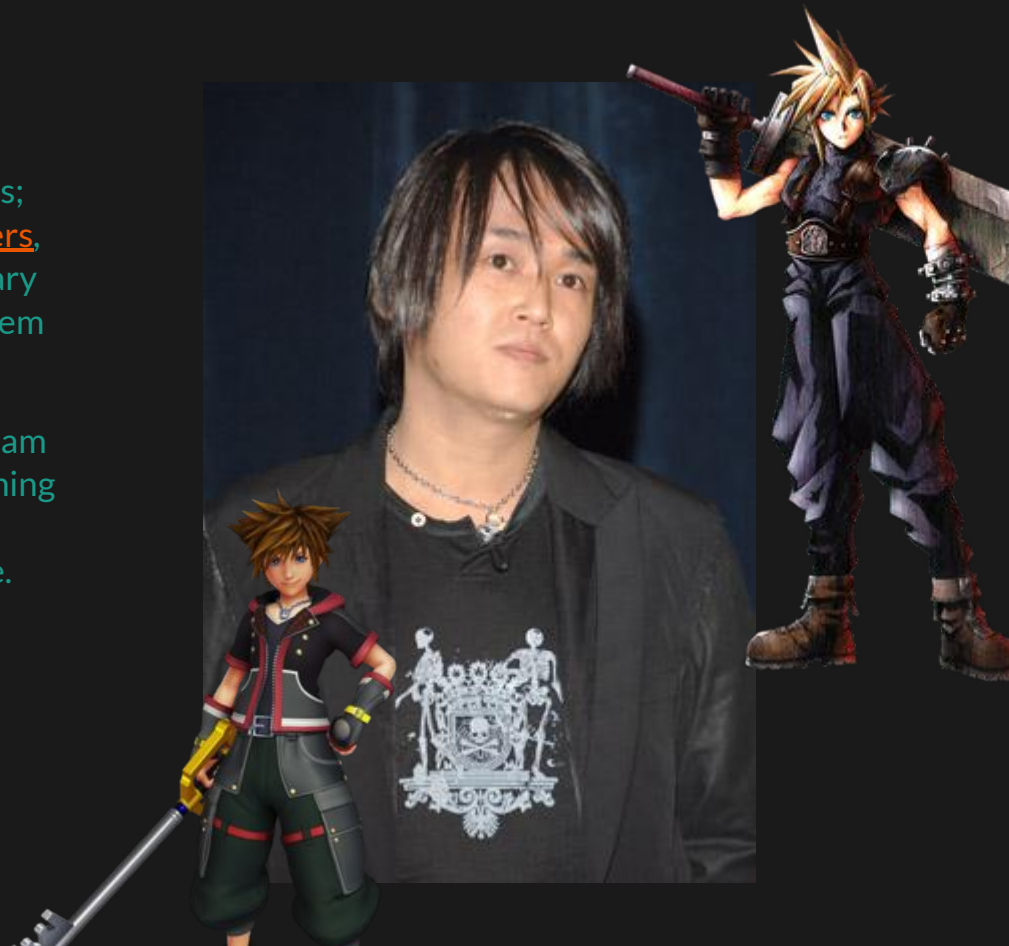
Feel free to compare it to mechanics in games that exist.

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]

Challenge: Art

Now start thinking about how the game would look?

Are you going to try to give the game a more cartoony or realistic look?

What kind of setting is your game going to take place in?

What do the character looks like?

Do enemies if you have any have visual indicators of how they should be defeated, or are the platforms marked a certain way to show the player where to go, ect.

Feel free to compare it to art in games that exist.

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.

Challenge: Sound

Now start thinking about how the game would sound?

What kind of music do you want to play?

What atmosphere do you want to set for you player?

Do you want over the top or more subtle sound effects?

Would you have voice acting?

Feel free to compare it to music and sound effects that exist.

Challenge: Share

Now that you have your game ideas set up form into groups and share what you've come up with.

Sharing will help you delve deeper into the core of what your game has to be and will allow you to see it from different perspective maybe even find somethings to add or tweak.

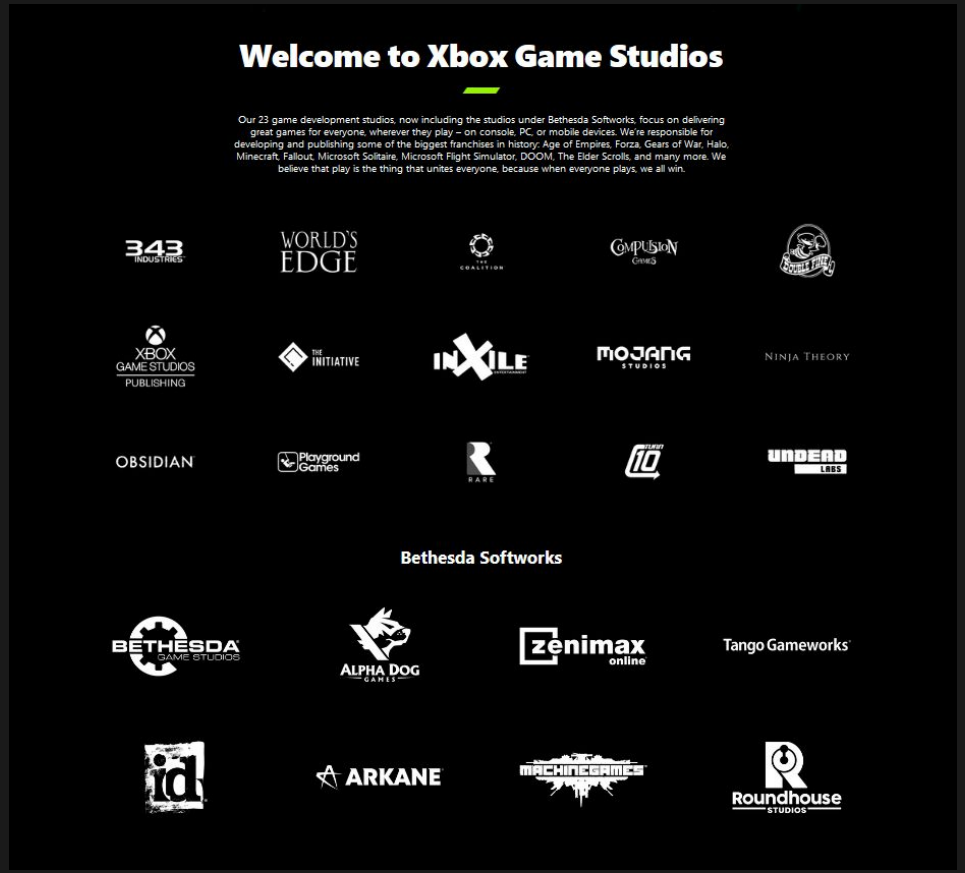
While you are doing that think about all the questions you have to answer as any one of these roles. Think about which one suits you the best.

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.



Crunch & Game Development Culture

Crunch refers to the rushed development of a game due to a close deadline, like the game's release date. If the game development timeline is poorly planned or managed developers are often pushed to work 80-100 per week which burns people out.

This tends to happen in the larger publishers and if you are ever faced with such a situation in any field step away. There are plenty of jobs out there and participating in an environment like this will only drain any passion you may have once had for the field.

Quality Assurance, unlike many of the other roles, isn't necessary for the majority of game production but only on its tail end. Therefore a lot of them are hired for contract work and once the game is done they all get laid off.

Because of the contract style of work they tend to not get benefits like other roles would and tend to receive lower pay and get overworked.

On a brighter note employees at Raven Studio are currently in the process of unionizing their Quality Assurance team for better working conditions. [Article]

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.

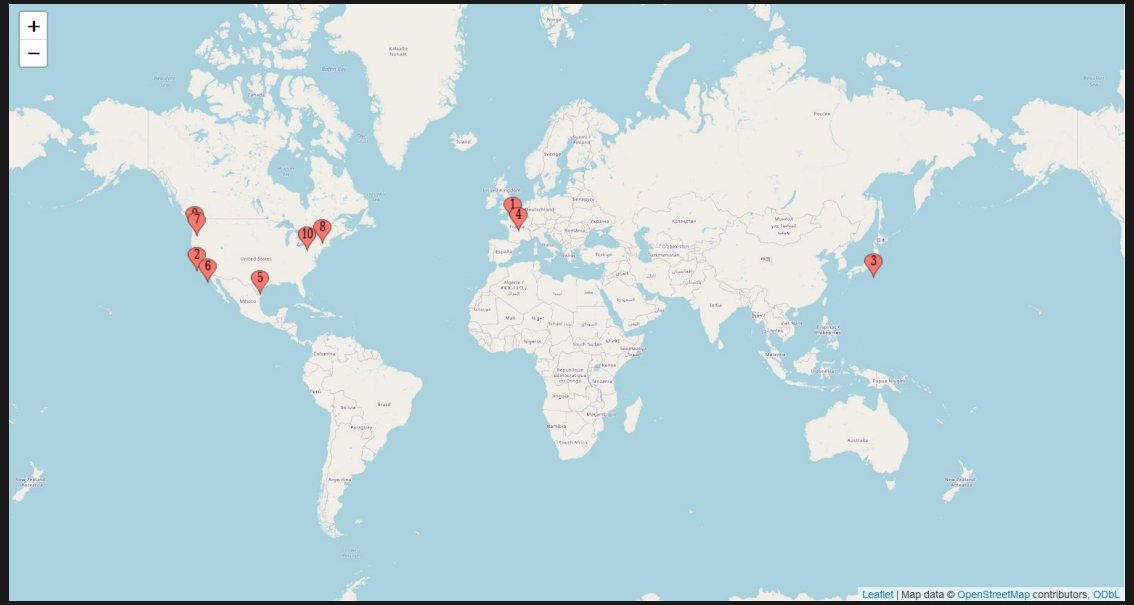
Do want to work in a large team or have personal freedom? Or do you want move to one of the few locations with a game development studio? What are you finances looking like as an independent developer? And many more questions like that.

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



Challenge: Developer Research

Go to https://www.mobygames.com/, this website host all of the credit information for games old and new.

Look up your favorite game and see who it is that made the game and what other titles they have worked on?

What Software Are We Using

Unity: Our Game Engine.

Visual Studio: The IDE we will be using

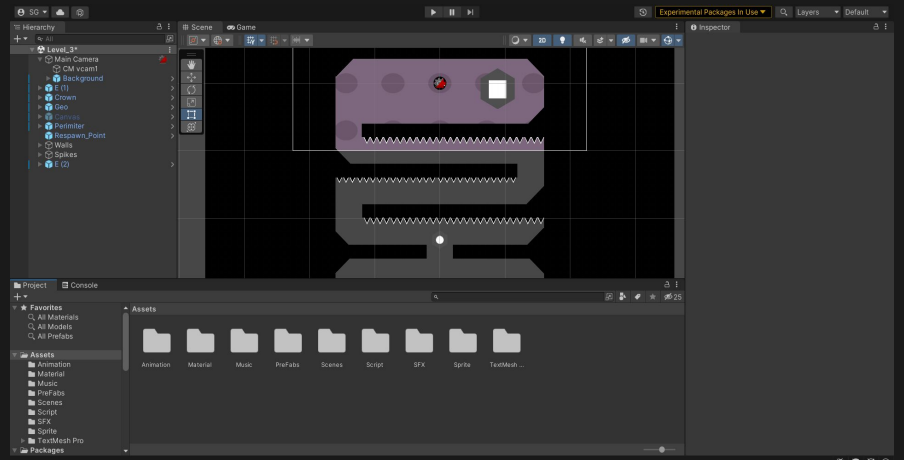
GitHub Desktop: The way we'll connect to the use of Version Control System.

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

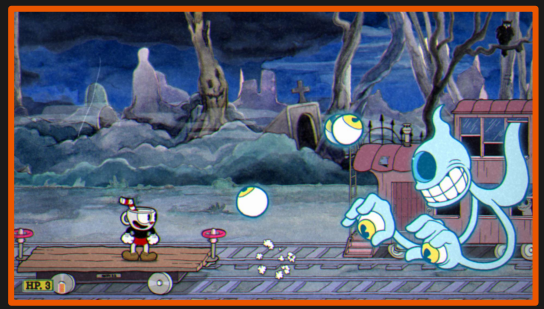
- Port it to any platform; Unity allows you to create executable files for your game to pretty much any platform that's currently available. This includes: Windows, Mac, Linux, Web Browser, Android, iOS, AR, VR, Playstation, Xbox, and Switch consoles.
- Is free software for personal use as long as your 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.

Sebastian Grygorczuk - STEM Institute at CCNY

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.

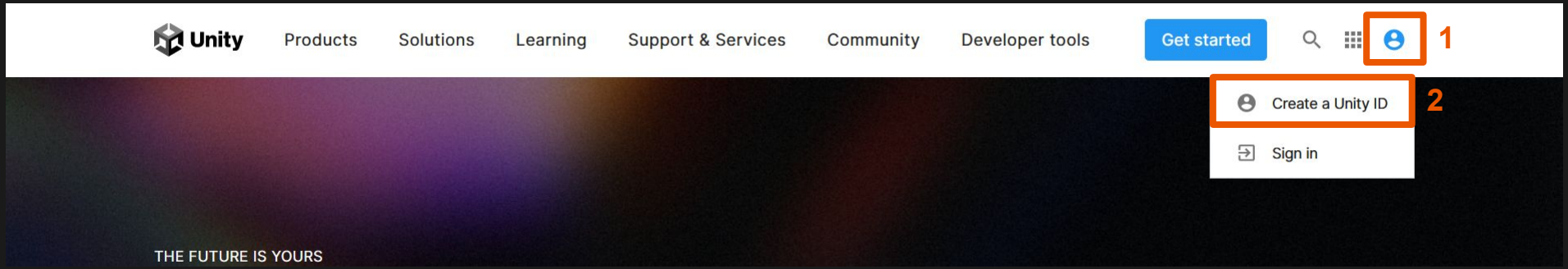
There dozens of game engines all different from each other however, many use similar standardized tools so that once you are familiar in one you can transfer your skills to another easily.

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

Making A Unity Account

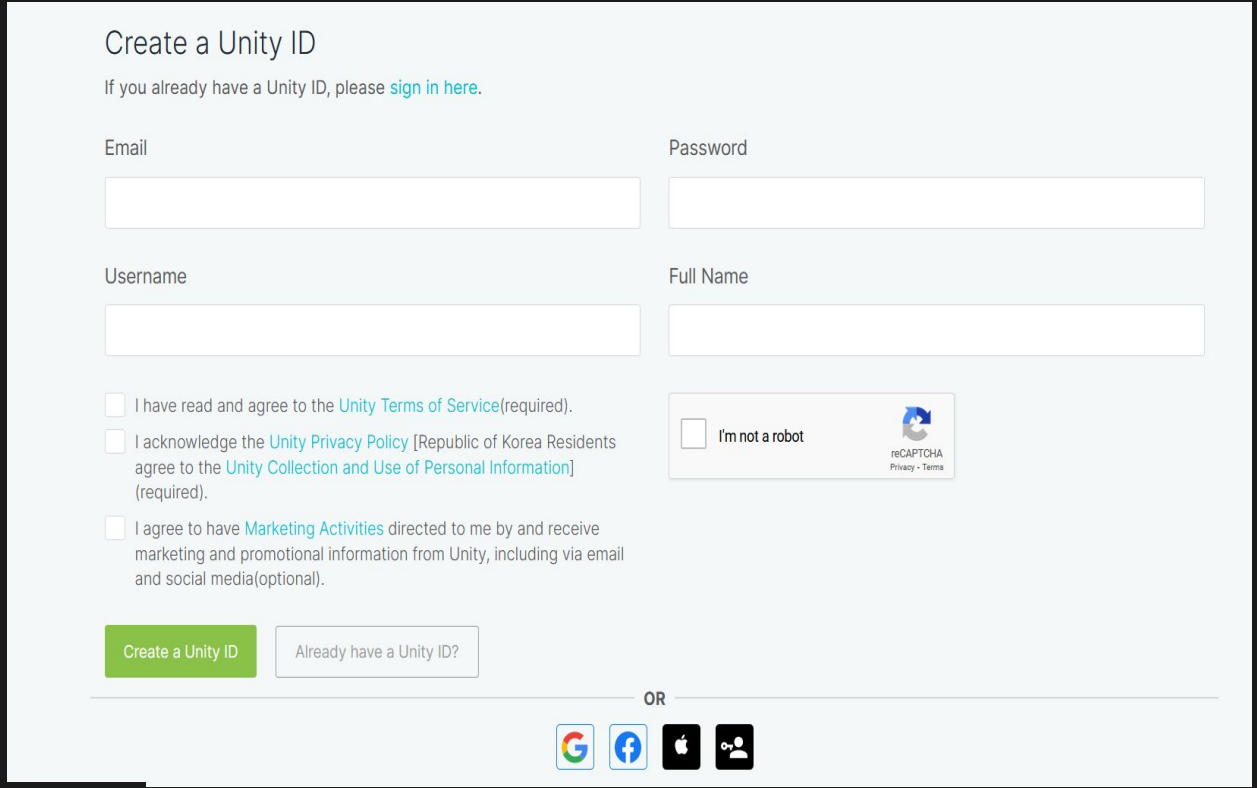
Before we install anything we will first create an account with Unity. This account will allow you to access the Unity Asset Store from which you'll be able to download 3D and 2D assets to create your projects throughout this class and possibly for your final project.

Head to https://unity.com/ and click the avatar in top right [1] and then click Create a Unity ID [2]



Making A Unity Account

Fill out the information or use an existing account to connect to Unity.



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.

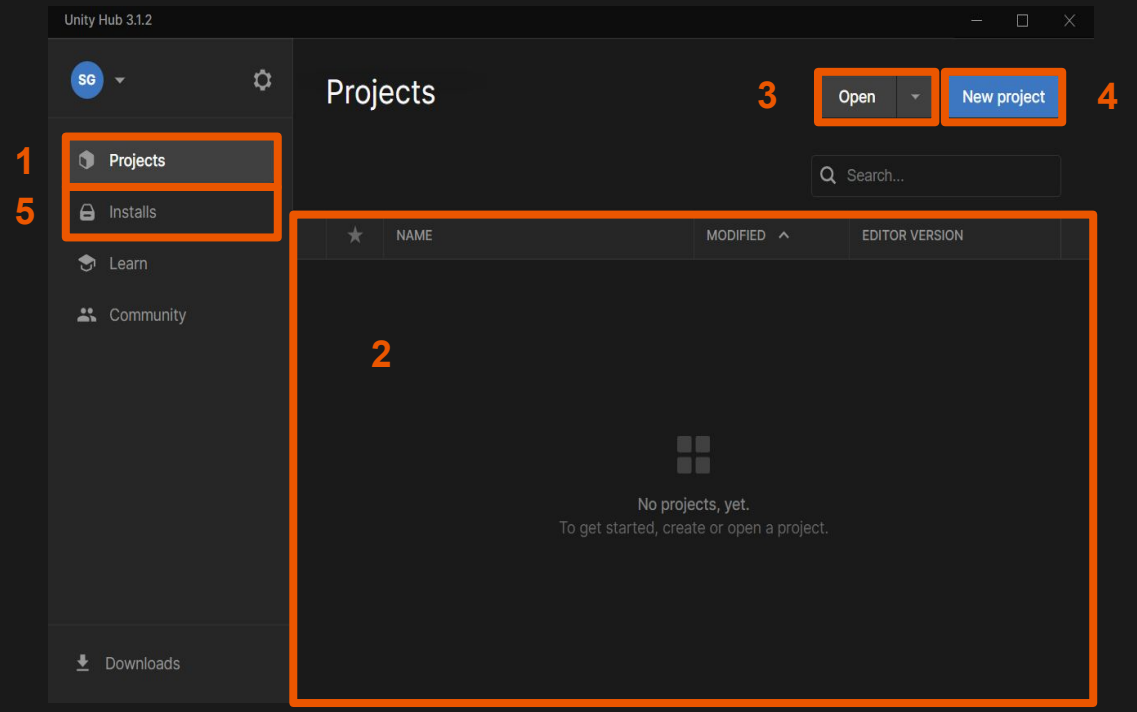
[1] The Projects Tab allows you to view all of the project you've recently worked on.

[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.

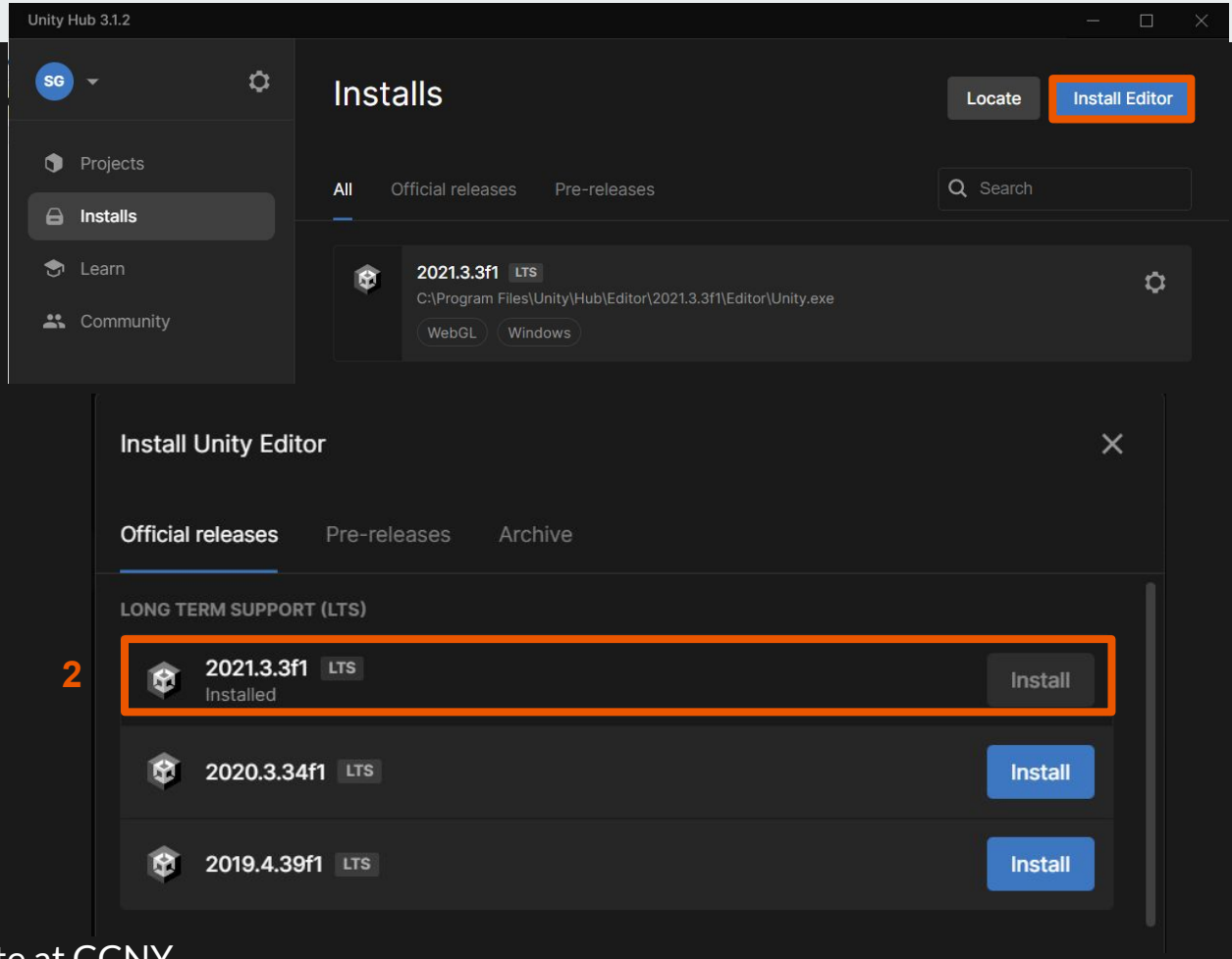


Sebastian Grygorczuk - STEM Institute at CCNY

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.

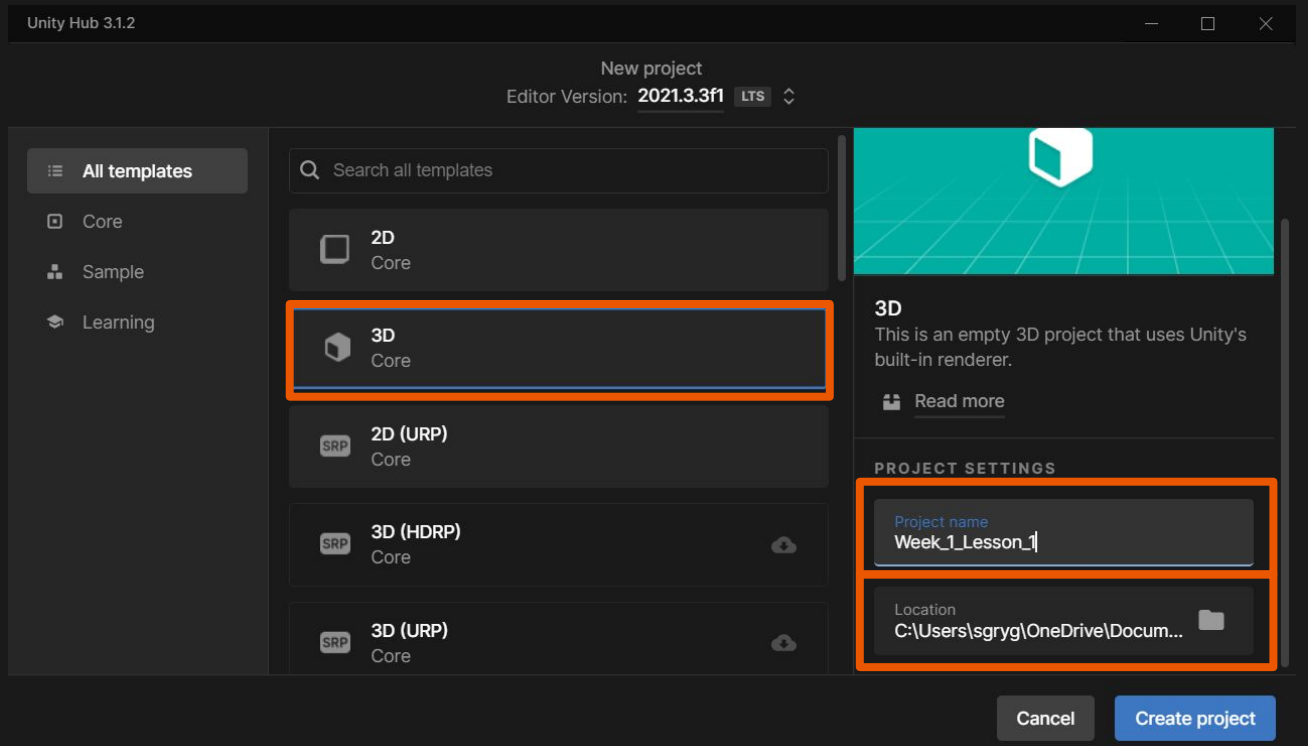


Opening up a new Project

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.

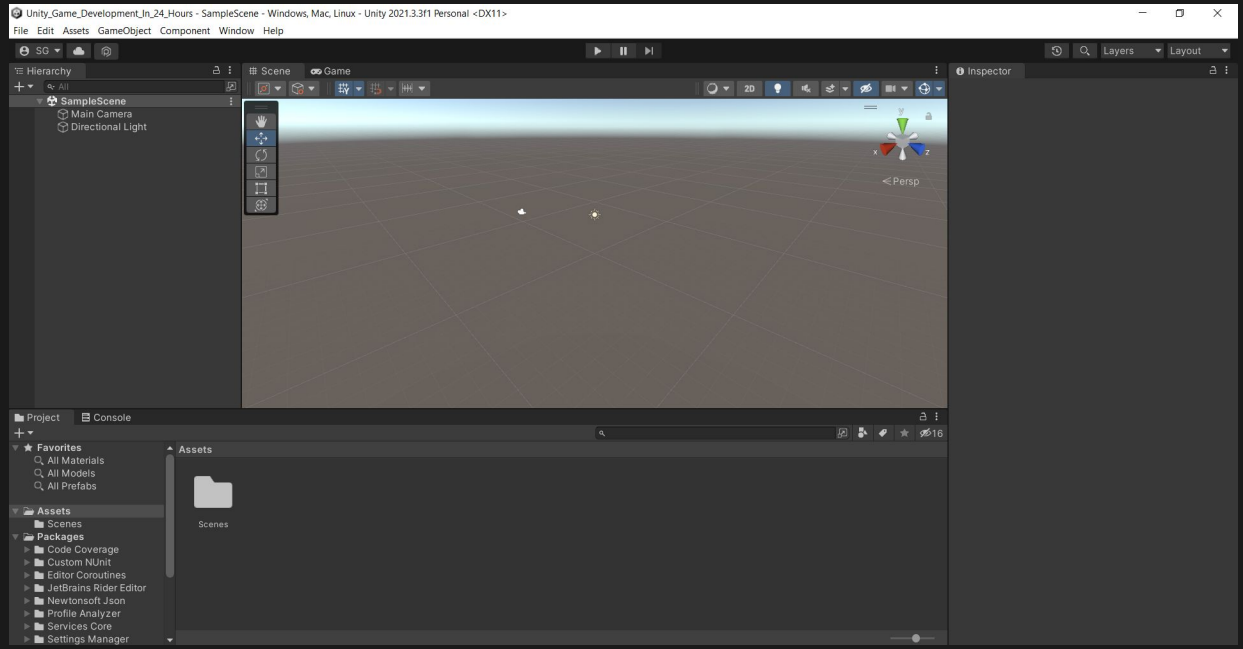
Once all of that is set, feel free to click Create Project.



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.



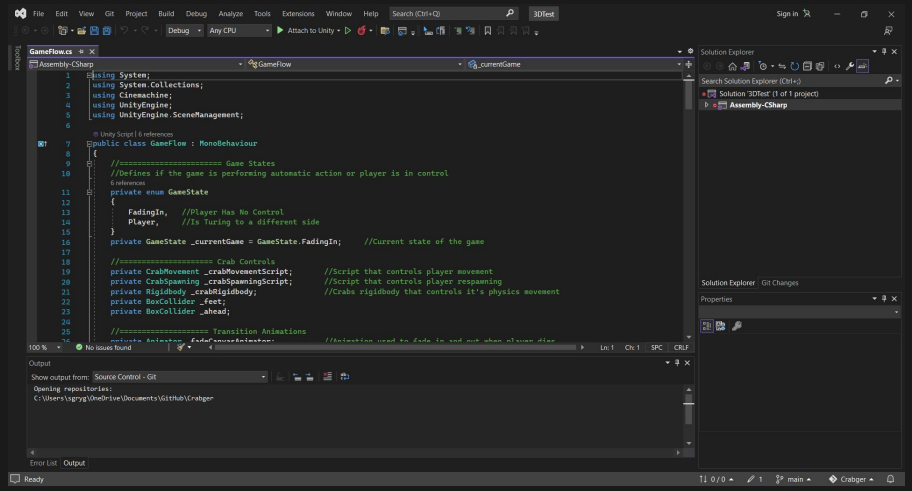
Visual Studio

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.



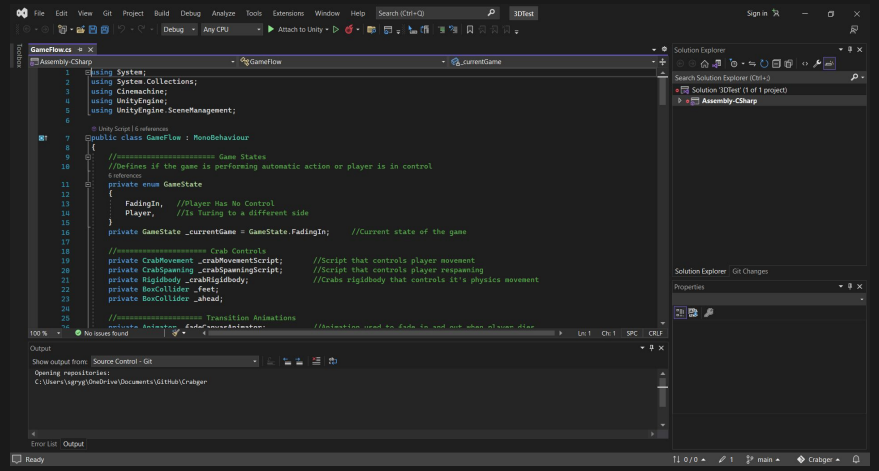
IDE vs Text Editor

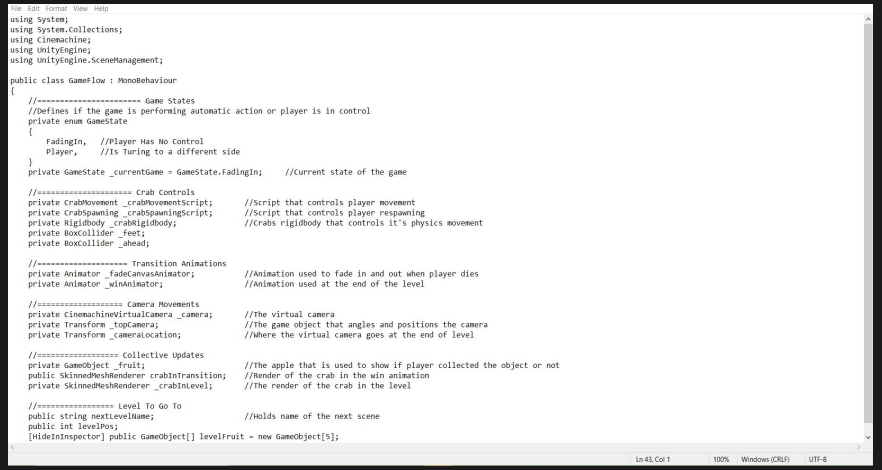
Here is the code shown in Visual Studio versus Windows Notepad. They more or less fulfill the same purpose of allowing you to write code.

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.





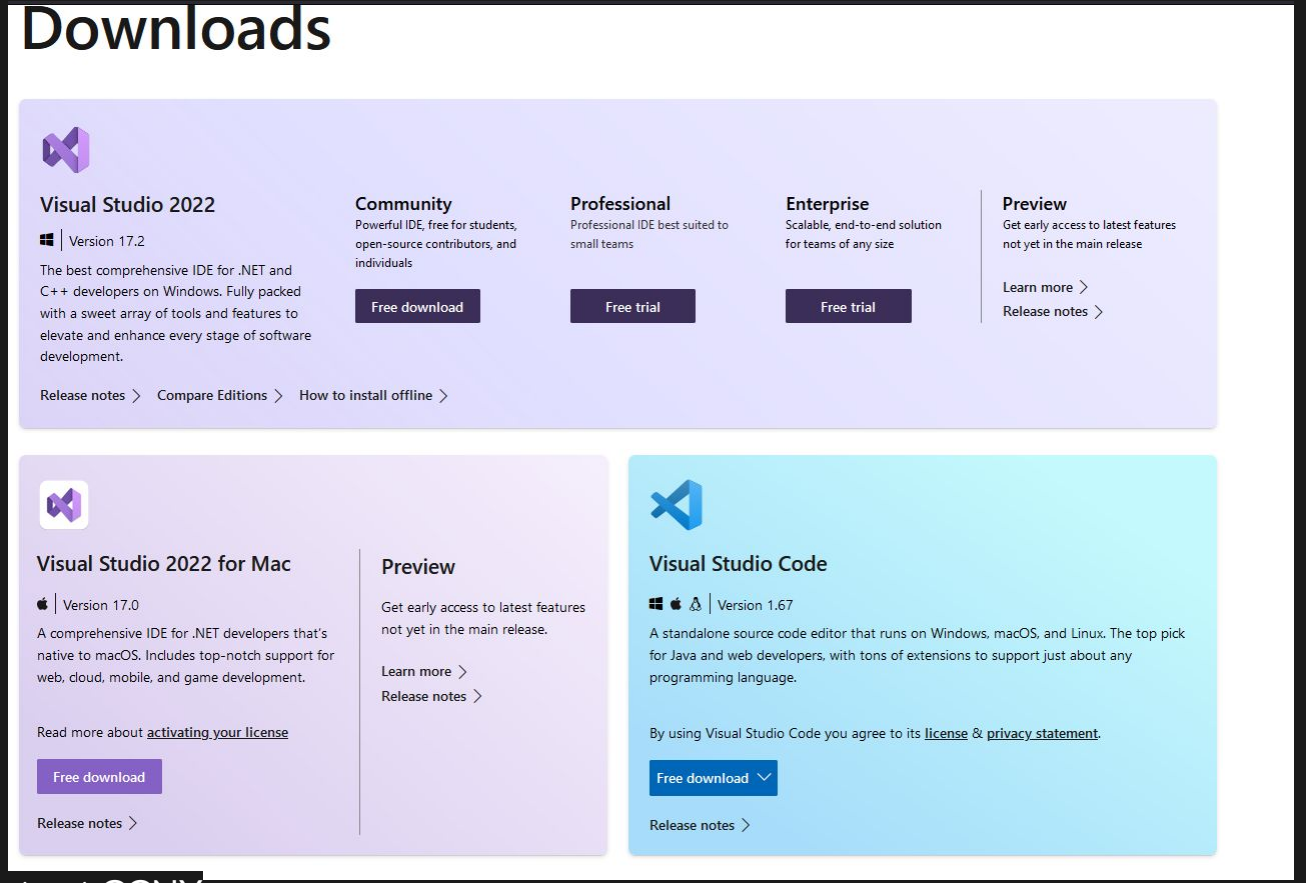
Sebastian Grygorczuk - STEM Institute at CCNY

Downloading Visual Studio

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

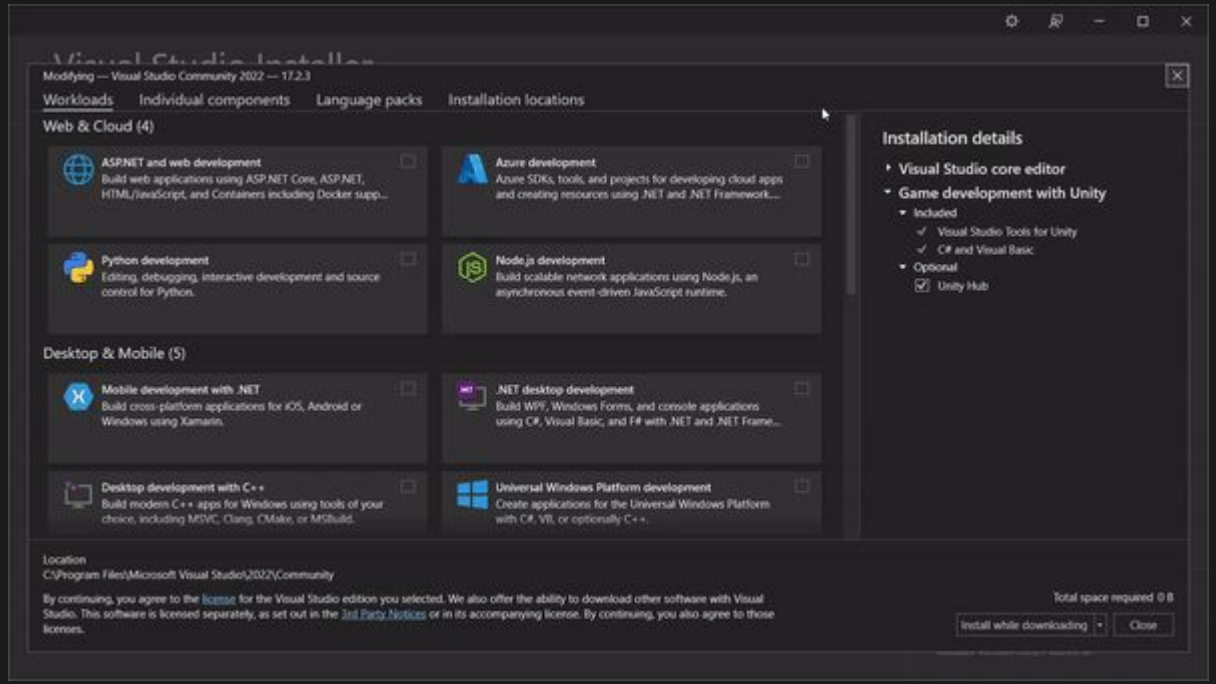
From here select the Community download for Windows and Free Download for Mac.

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

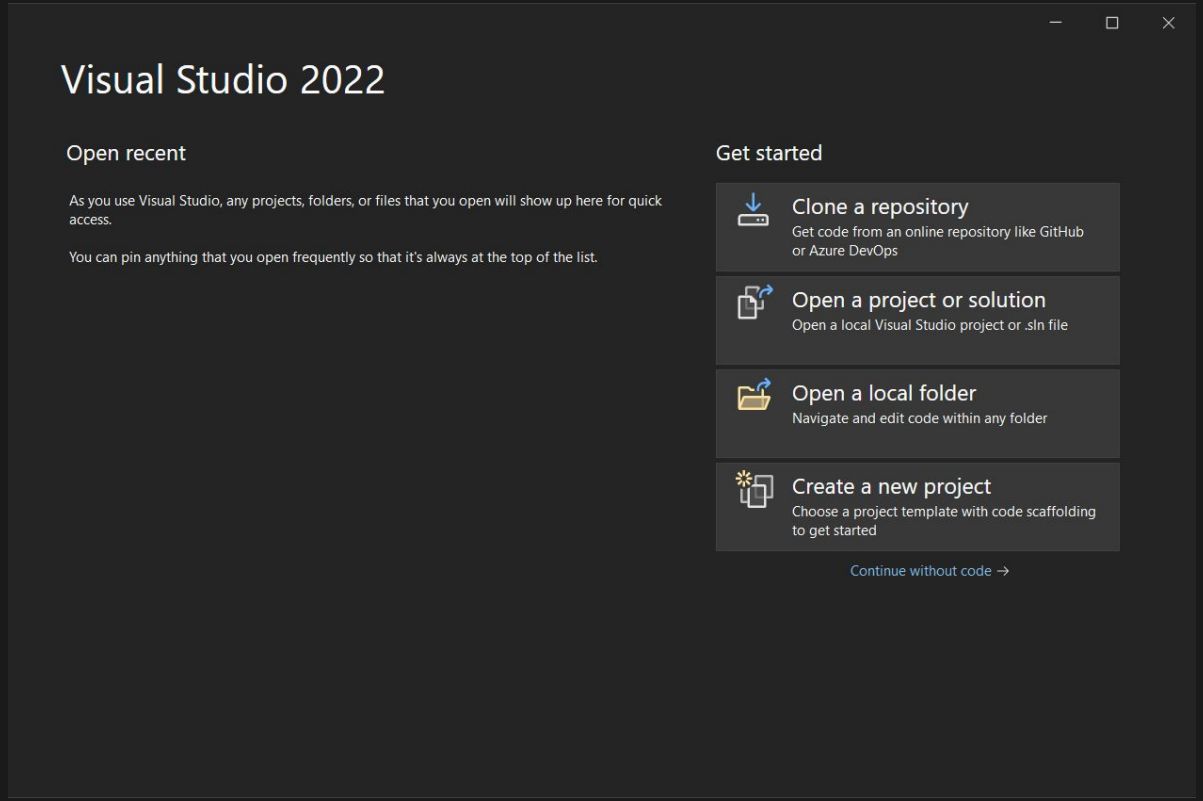
During the installation it will ask you if you want to download any Workloads, scroll down and check the "Game Development with Unity" that's the only thing you'll need for this course.



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.

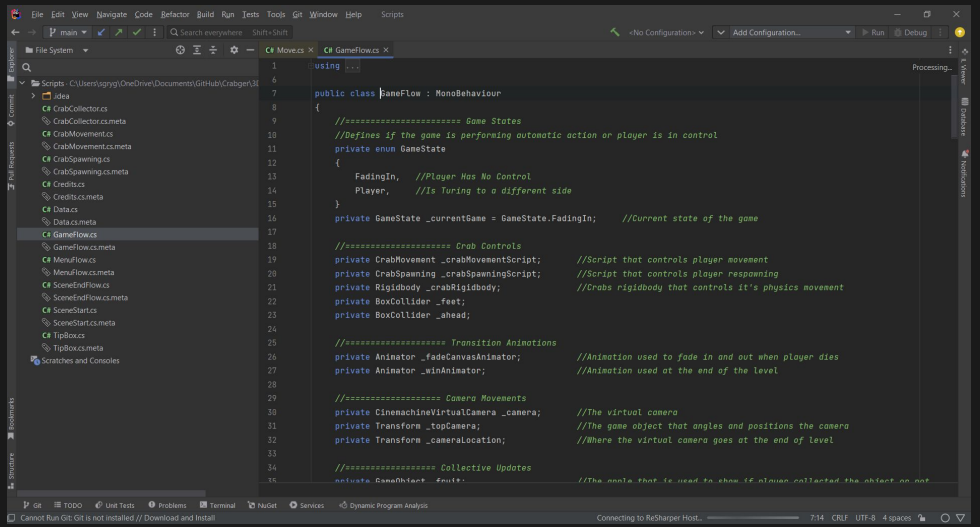


Possible Alternative - Jetbrains Rider

If you ever end up working on Linux, Visual Studio Code will not be your friend. It's heavily scaled back compared to it's Windows and Mac versions becoming more or less a Text Editor with color coded code.

An alternative to Visual Studio Code would work on Windows, Mac and Linux would be Jetbrains Rider, it has more features, has better UI and I personally use it to program games. However, Rider has no free version and you have to licence it out.

Unless you are actively programming games for an extend period of time I don't recommend you spend money on it.



Additional Resources: Jetbrains Rider & Visual Studio vs JetBrains Rider: A Detailed Comparison

GitHub - Version Control System

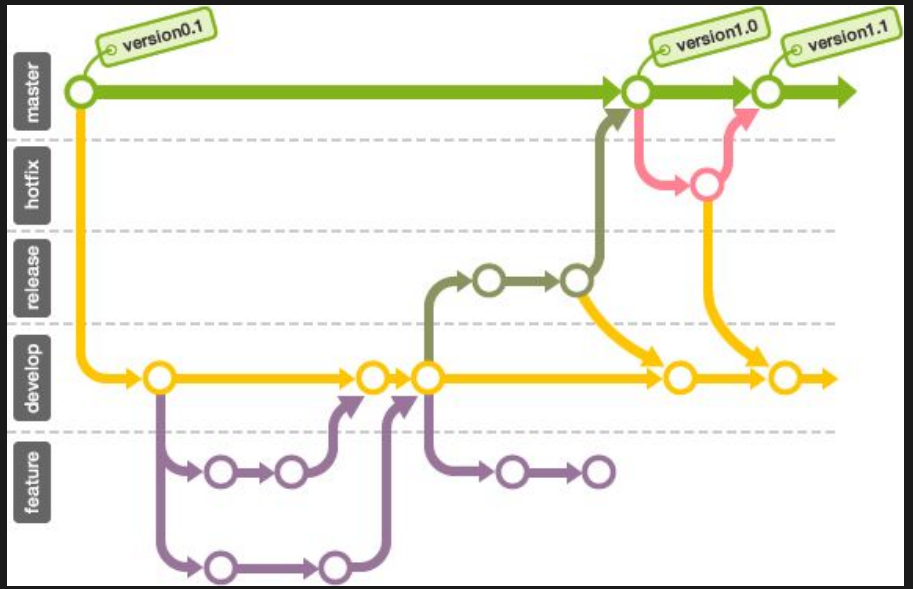
What is Version Control System (VCS)?

- A way of storing and updating project files for individual or group project.

Think of our game as a Google Doc and VCS will allow us to all make changes to it actively while recording the history of changes and allowing us to rollback to previous moment in time.

It allows two or more people to work on a project and if those individuals edited the same work it will show them as conflicts allowing you to go line by line of code to choose the best of both parts.

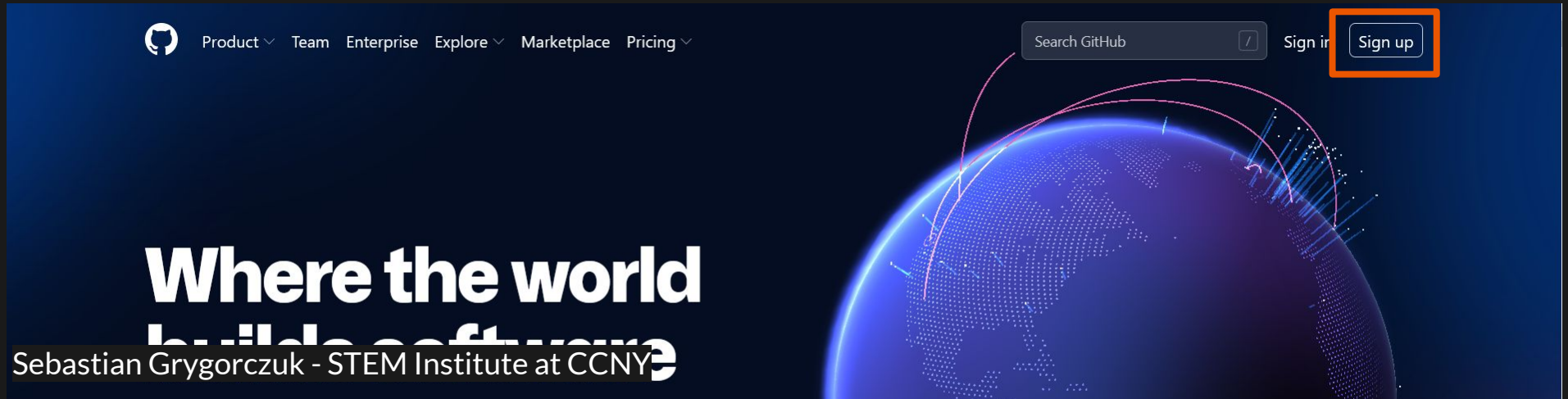
What is GitHub? It's the Hosted Version Control, it's all of your files you store on Google.



Additional Resources: Git vs. GitHub: What's the difference

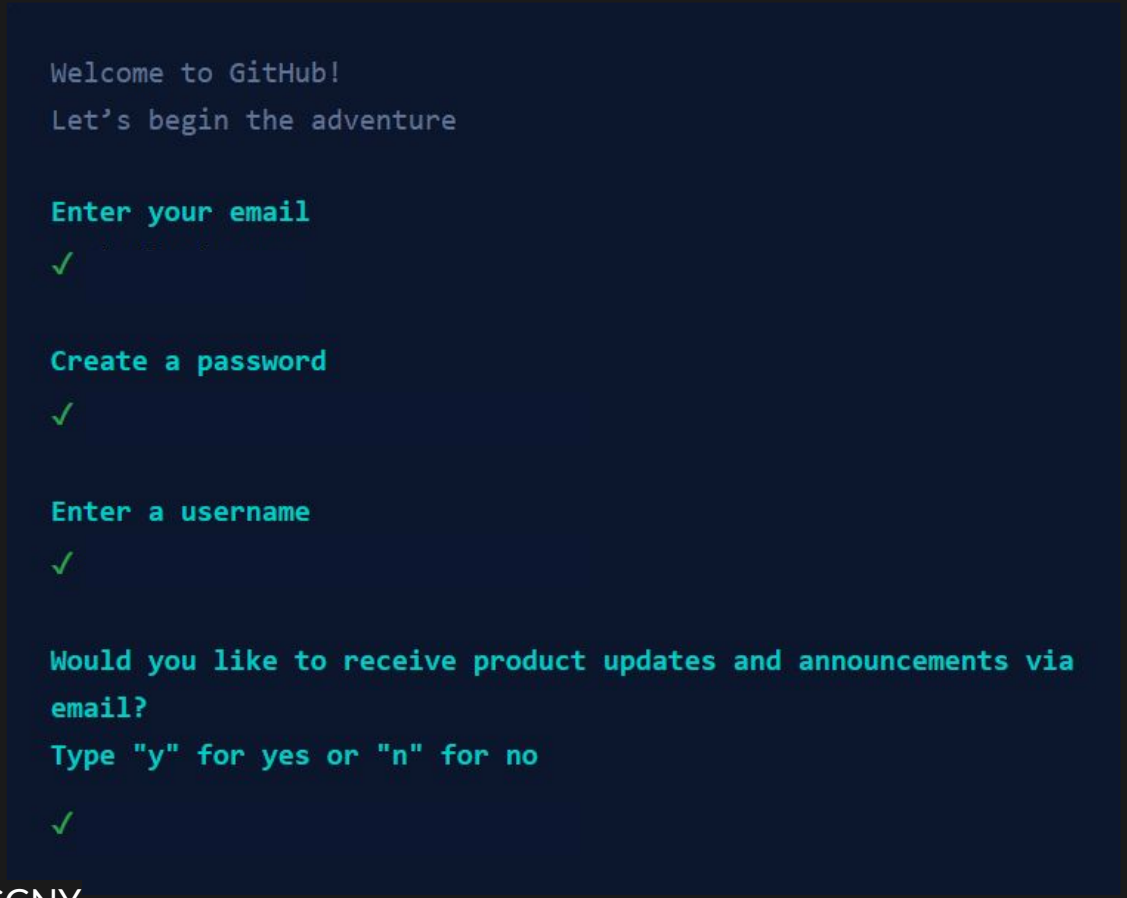
Creating A GitHub Account

Let's start by making an account on GitHub https://github.com/, GitHub is a web-based version-control and collaboration platform for software developers and is operated by Microsoft.



Creating A GitHub Account

Create a GitHub account by filling out your information in GitHub's adventure styled form.



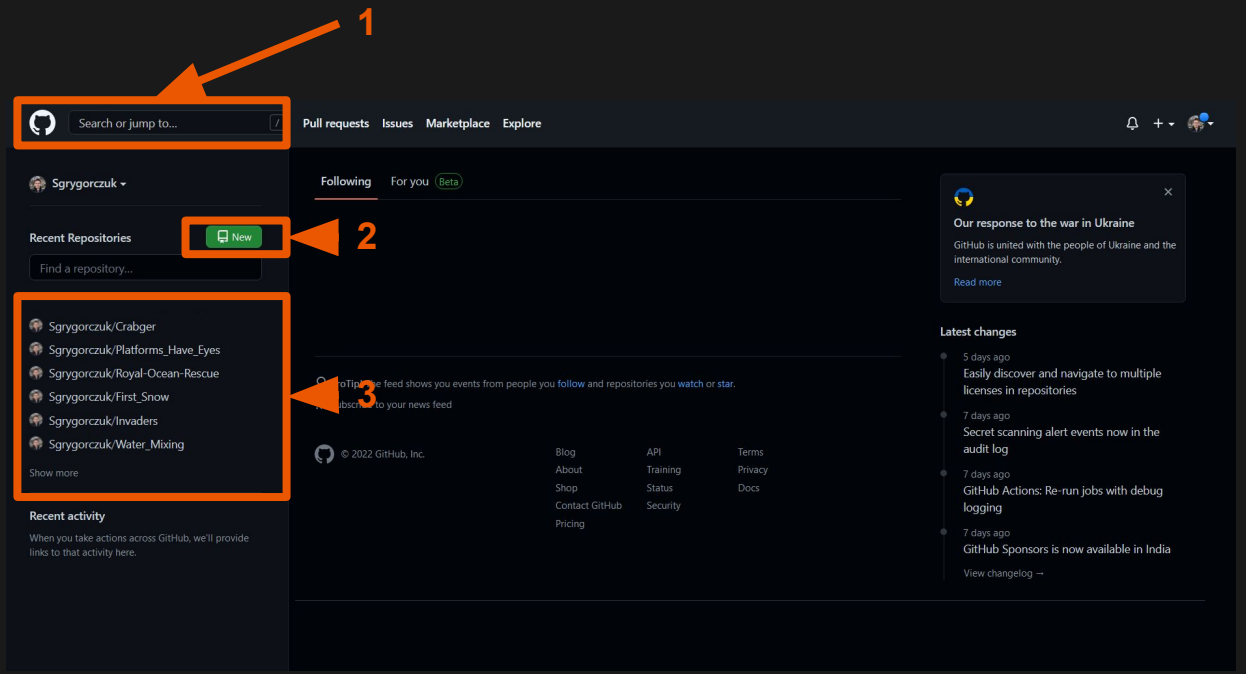
GitHub

Once you're done setting up your details you will be sent to the Landing Page. Let's go over few of it's features

[1] the search bar allow you to browse through everything that is publicly stored on GitHub. All of the code you fill find here is called Open-Sourced, means it's available for everyone to access and modify. There may be limitations depending on project, check the ReadMe attached to the project.

[2] Creating a new Repository, repositories are just folders.

[3] History of Recent Work, will allow you for quick access to whatever you're working on.



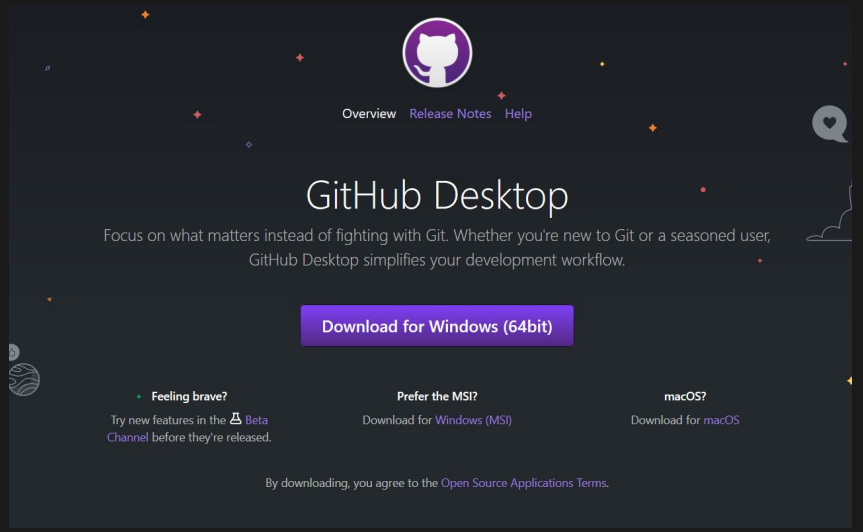
Downloading GitHub Desktop

The way a lot of people work with VCS is by using the Command Line, although giving you greater control over how and what you are doing it can be very overwhelming to have to move folders and uploading project files using command line prompts.

To alleviate that we will download GitHub Desktop which we will connect to the GitHub account we just made to have a visual way to control our files.

Go to https://desktop.github.com/ select the system you're working on and download the program.

Anyone feeling more daring and wants to use the Command line can use this Cheat Sheet.



Sebastian Grygorczuk - STEM Institute at CCNY

GitHub Desktop

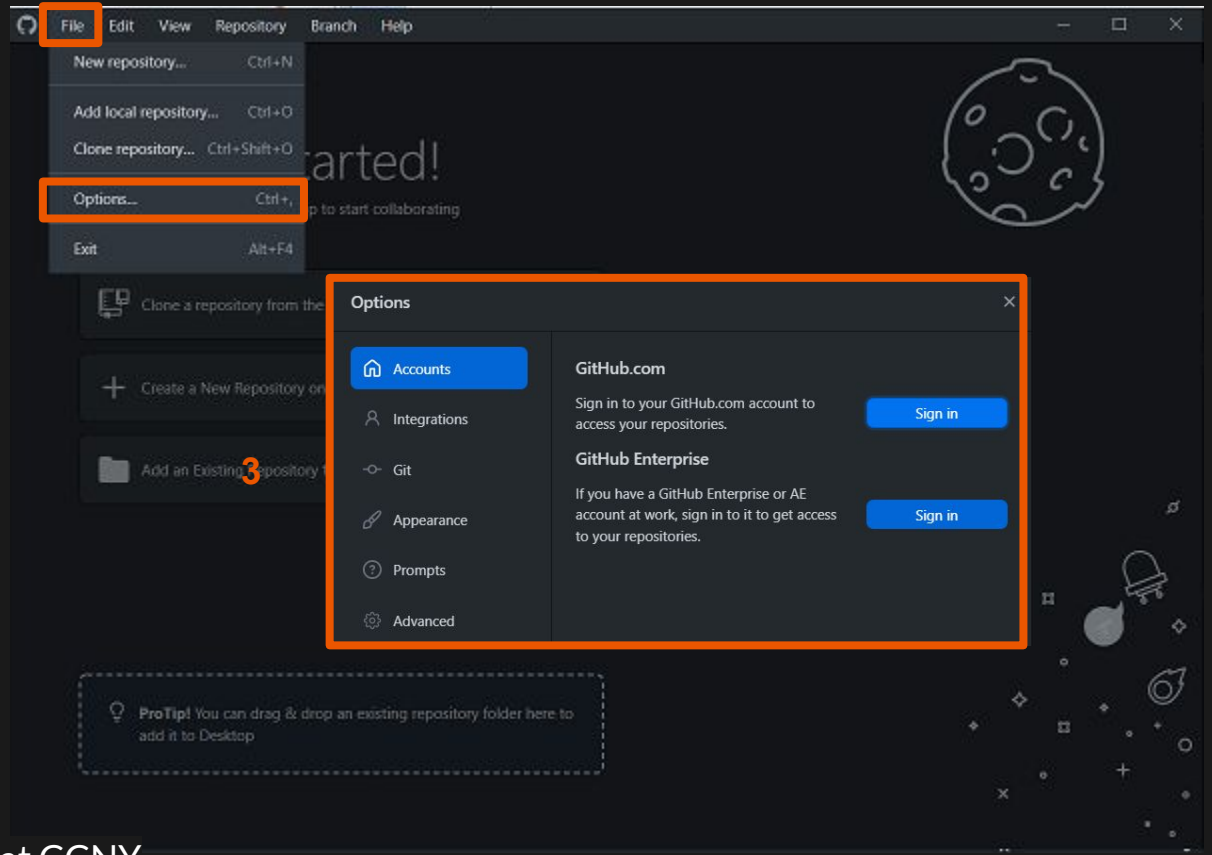
Once GitHub Desktop is installed you will be met with this page.

First thing we're going to do is connect you to the GitHub account we just made.

Click on [1] the File in the Toolbar and then select [2] Options.

Alternatively you can just click **Ctrl + ,**

From there you'll receive the [3] PopUp, click sign in. It's going to send you to your browser and since you've just made your account and should still be logged in it will automatically connect.



Version Control System Commands

Git is full of keywords that we will use inside of GitHub Desktop to access and modify our projects. The ones we'll encounter the most are:

Clone: Copying/Downloading the Repository to make a local copy.

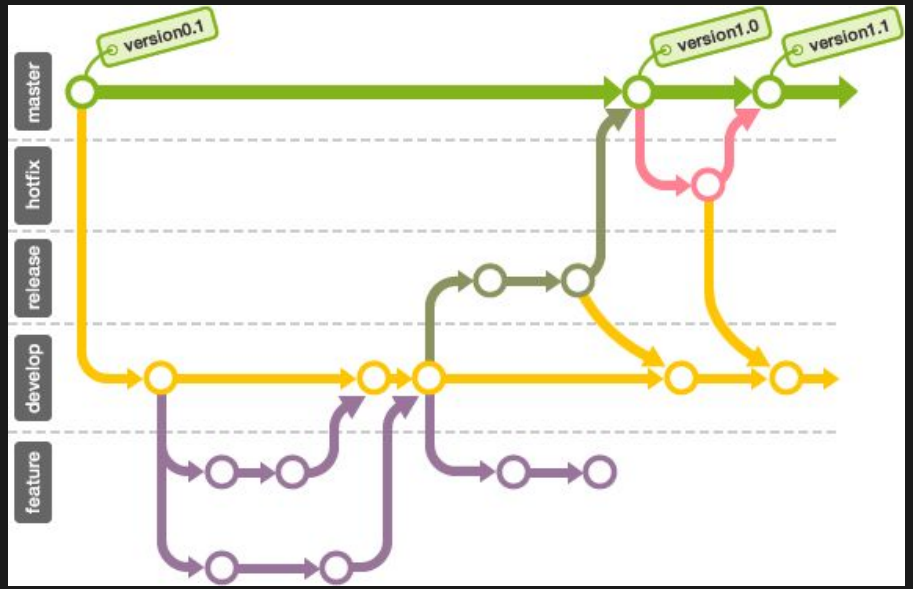
Fork: Separating from the current branch. Allowing you to one day Merge or keep it as your own personal project.

Merge: Connecting your branch to it's parent. Have to resolve any conflict.

Commit: Indicates that you are creating a snapshot of what your repository looks like now.

Push: Allows you to put whatever changes have been saved in the Commit into the cloud version of the project.

Pull: Pulls down any updates that occurred in the project, if you already have the project on your desktop.



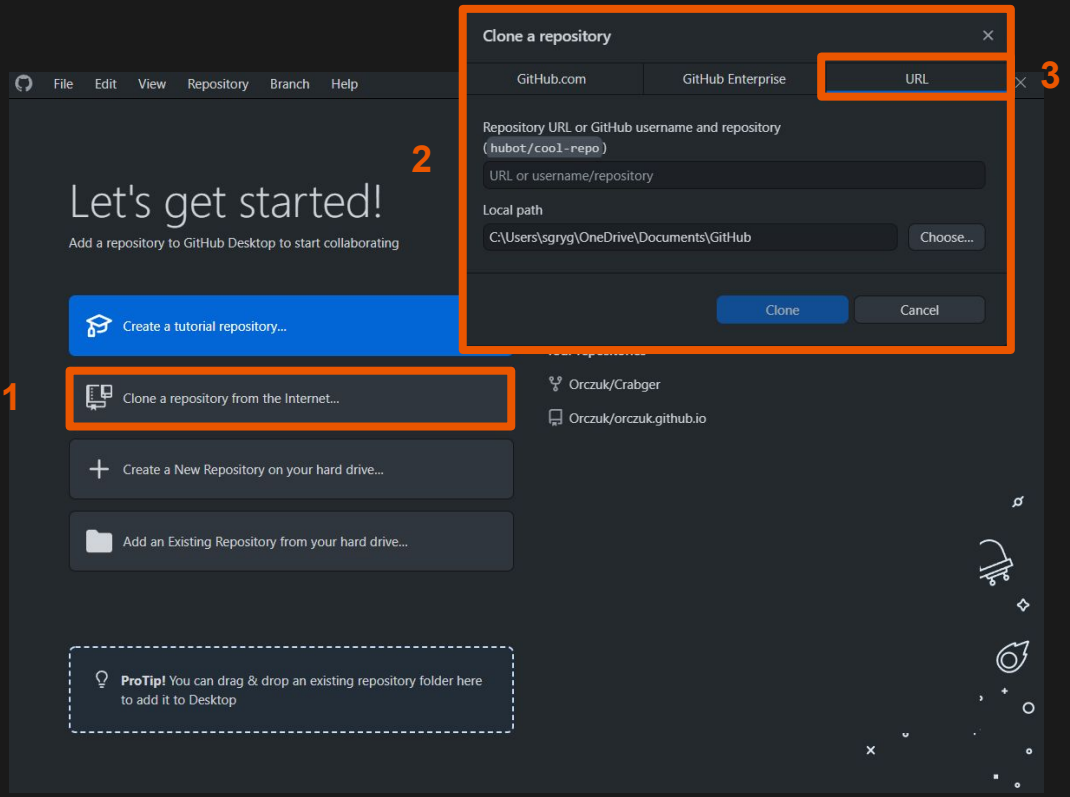
Version Control System Commands In Action

Let's start by cloning the Project Files for Day 2 Lesson. From you landing page click on the [1] Clone a repository from the Internet.

That should give you the [2] Clone a repository PopUp, select [3] URL and paste in

https://github.com/Sgrygorczuk/Week_1_Lesson_2

This repository holds the project files that we will use tomorrow to explore the Unity Game Engine.



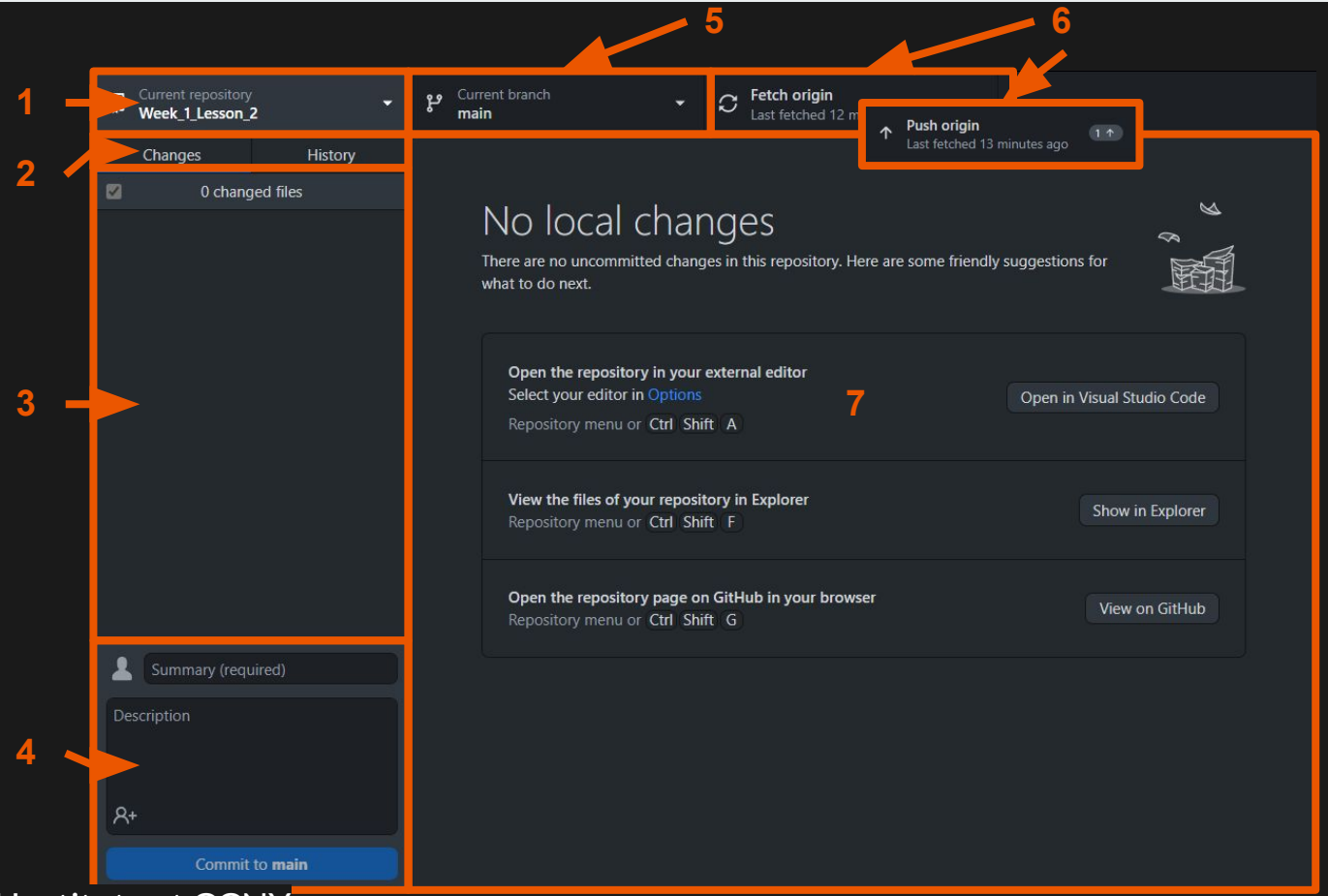
Sebastian Grygorczuk - STEM Institute at CCNY

Version Control System Commands In Action

Now the GitHub Desktop will look a lot different. Let's break down the sections.

[1] Project name, clicking on it will allow you to switch to a different project.

[2] Changes and History Tabs, allows to change what's shown in [3] either all the files that are changed or the history of Pushes the Respostiary has received.



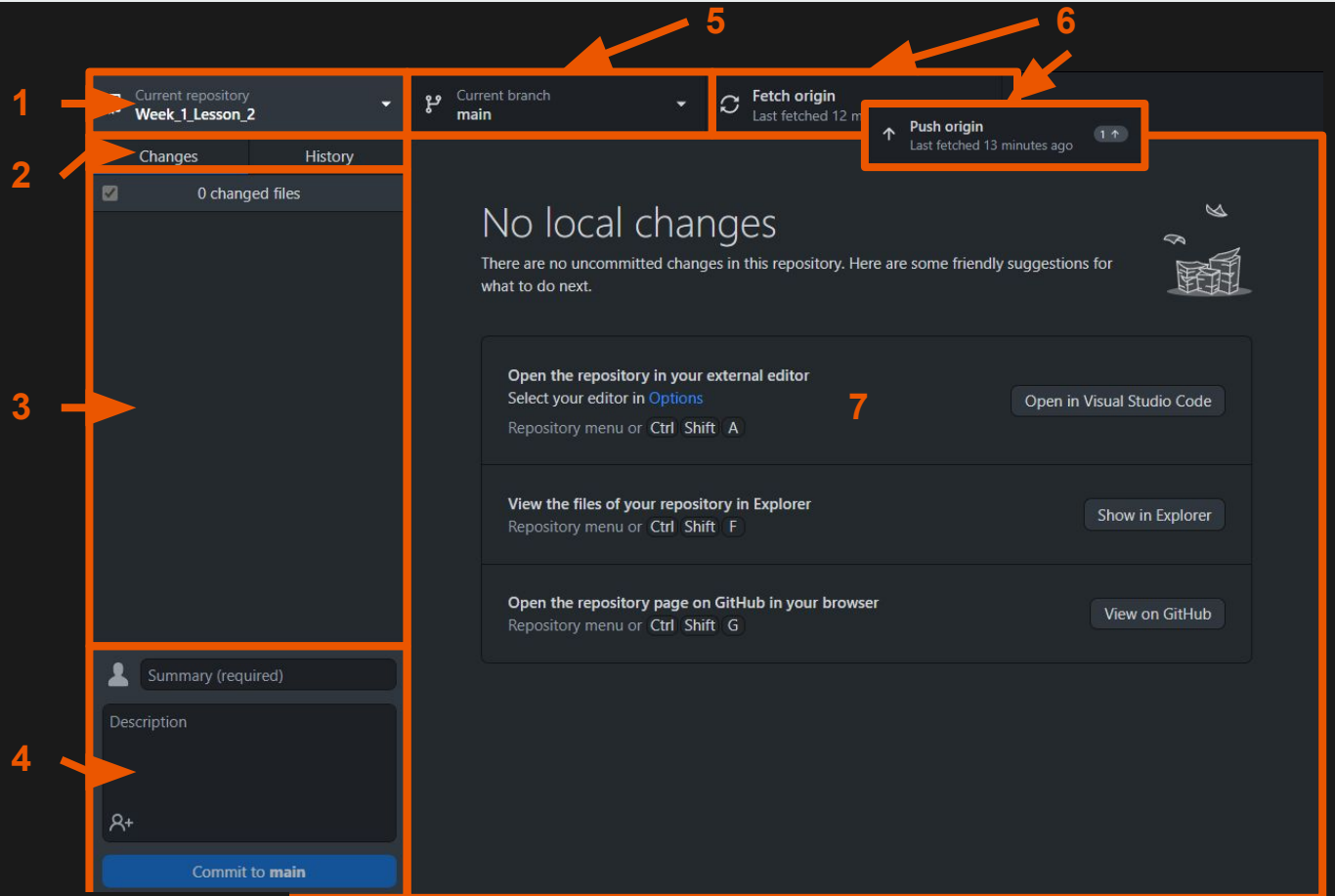
Version Control System Commands In Action

[4] Allows you to Commit or snapshot your changes, you have to provide it with a Title and a Description so that when you look back on history you know what you were doing with the given changes.

[5] Tells you what brach you're on, most of the time it will be main and you won't need to change it.

[6] Tells you last time the project was updated and if you have a commit ready will allow you to push it to the cloud.

[7] Will display the changes line by line of the file currently selected in [3].

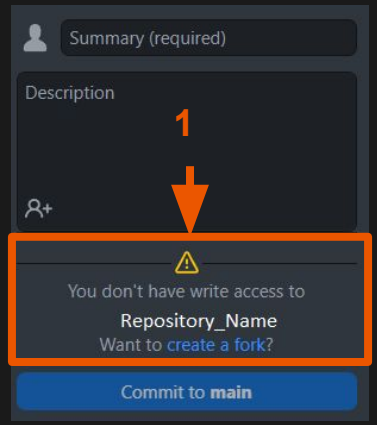


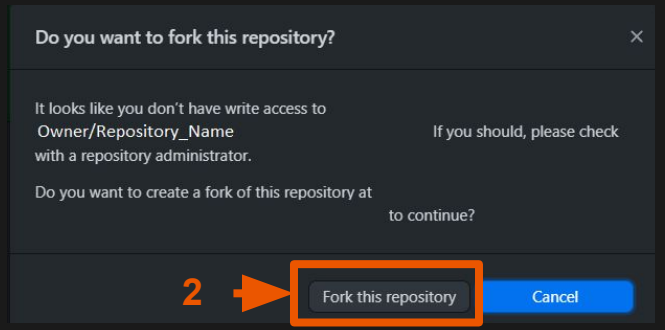
Forking Your Own Copy

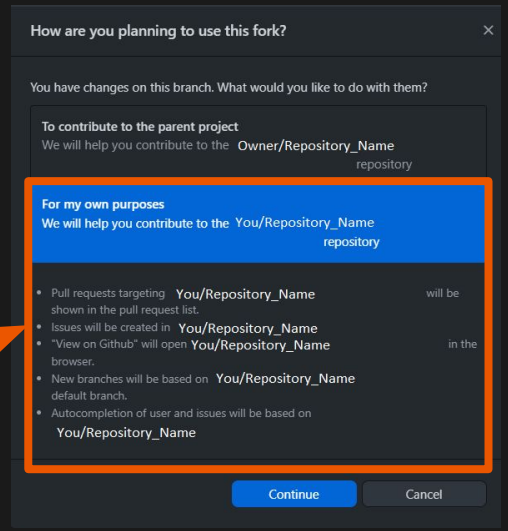
Because the Project Files that were downloaded are my files, you won't be able to just Commit your changes to my copy of the project.

To make your own Copy of the project we will Fork it. GitHub Desktop will put a [1] warning on Commit telling you that you don't have access to the Repository and you should Fork.

If so click [1] Create a Fork, then in the PopUp menu select [2] Fork this Repository and select [3] For My Own purposes, that will create a distinct copy that only you will be able to edit while the first option will fork off of mine branch.







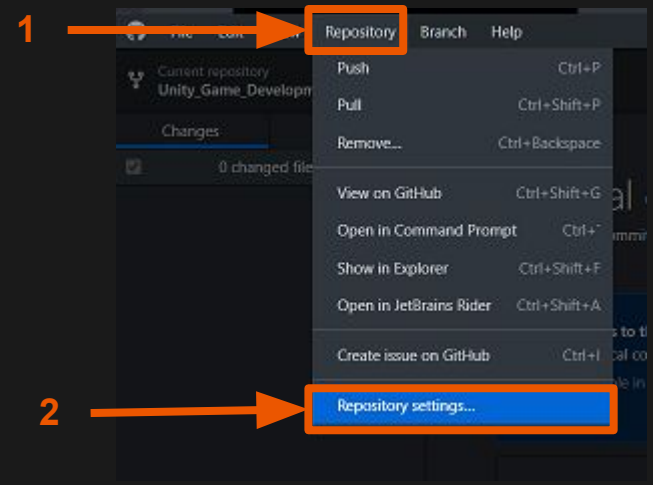
Sebastian Grygorczuk - STEM Institute at CCNY

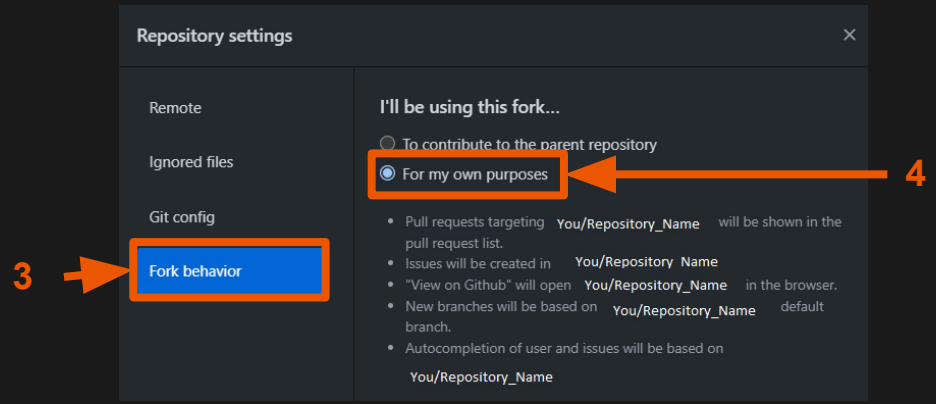
Forking Your Own Copy

Incase that Warning Sign doesn't show up in the Commit Area, you can go to the Toolbar select [1] Respiratory and go to [2] Repostiary Settinging.

This will bring up the PopUp go to [3] Fork Behavior and Select [4] For My Own Purposes. After that click Save.

This will have the same effect as if the Warning PopUp giving you a copy of the Repository to commit and Push with your own changes.





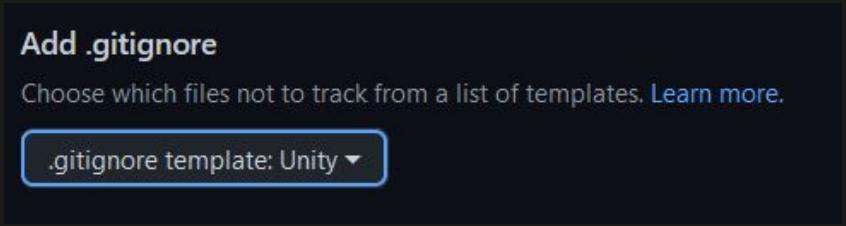
Sebastian Grygorczuk - STEM Institute at CCNY

.gitignore File

When creating a repository GitHub it will ask if there is any files that shouldn't be tracked. For Unity you definitely want to do that.

If you don't select .gitignore template: Unity, GitHub will keep track of thousands of files that are used during the execution of the program that don't need to be uploaded.

If you didn't selected it you can alway just add a text file called .gitignore with the code to the right and save it in the project folder.




```
# This .gitignore file should be placed at the root of your Unity project directory
#
# Get latest from https://github.com/github/gitignore/blob/master/Unity.gitignore
#
/[Ll]ibrary/
/[Tt]emp/
/[Oo]bj/
/[Bb]uild/
/[Bb]uilds/
/[Ll]ogs/
/[Mm]emoryCaptures/

# Asset meta data should only be ignored when the corresponding asset is also ignored
!/[Aa]ssets/**/*.meta

# Uncomment this line if you wish to ignore the asset store tools plugin
# /[Aa]ssets/AssetStoreTools*

# Autogenerated Jetbrains Rider plugin
[Aa]ssets/Plugins/Editor/JetBrains*

# Visual Studio cache directory
.vs/

# Gradle cache directory
.gradle/

# Autogenerated VS/MD/Consulo solution and project files
ExportedObj/
.consulo/
*.csproj
*.unityproj
*.sln
*.suo
*.tmp
*.user
*.userprefs
*.pidb
*.booproj
*.svd
*.pdb
*.mdb
*.opendb
*.VC.db

# Unity3D generated meta files
*.pidb.meta
*.pdb.meta
*.mdb.meta

# Unity3D generated file on crash reports
sysinfo.txt

# Builds
*.apk
*.unitypackage

# Crashlytics generated file
crashlytics-build.properties
```