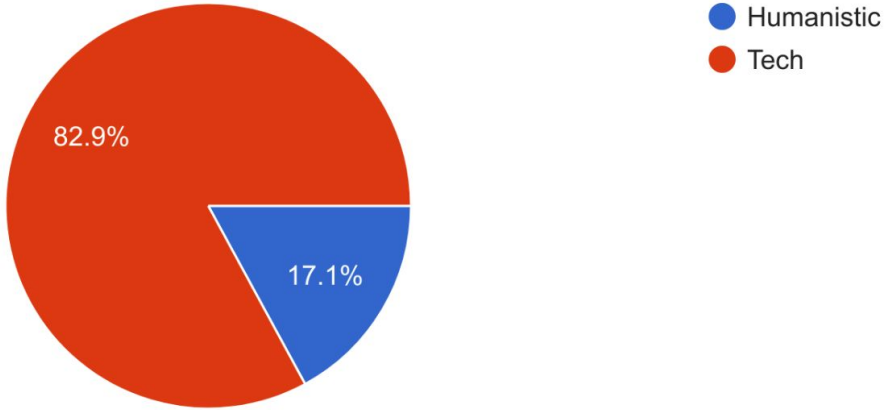
Basic Programming

Survey answers

Are you a humanistic or tech student?

41 responses



Motivations

- Because I really like games
- It looked fun and interesting
- I want to learn programming, but in a fun creative way
- It's more entertaining than Computer Science, and I can still learn programming.

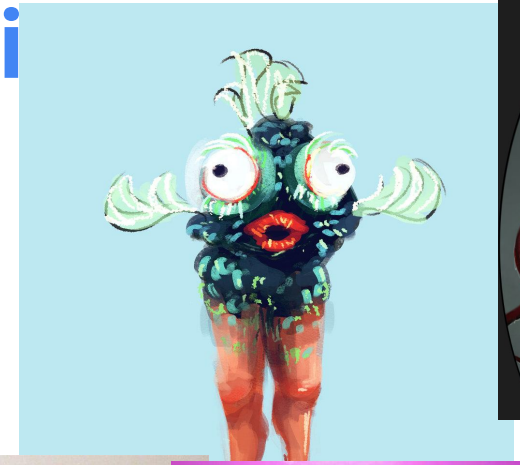
Topics that you want to see in the course

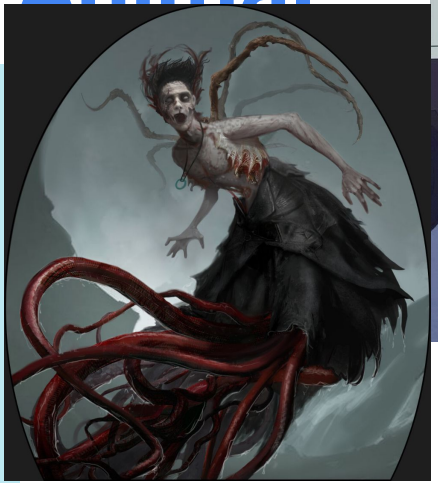
- Any type of programming
- Lots of different game types and how to program them
- Multiplayer code (not local)
- how to make a game
- Pcg
- Maybe how to use multiple inputs for multiplayer
- Procedural animation
- Artificial intelligence in games
- How to make a good bossfight
- physics
- Self made game engines
- Game design, of course. Perhaps the meaning of life, if you have it avaliable

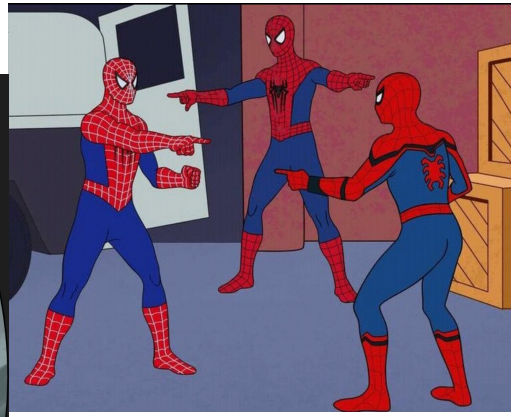
What do you want from this class?

- Knowledge about programming and usage of different programs
- The ability to create games
- Game dev skills i suppose?
- Some more knowledge of the game development word
- The ability to learn to comfortably at least create the basics of a unity game
- The ability to create exciting stuff
- An understanding of how to figure out the different strategy and achieve a structure in the code while making the games.
- To be an absolute legend in the game development community

Weirdest Human/Animal combi



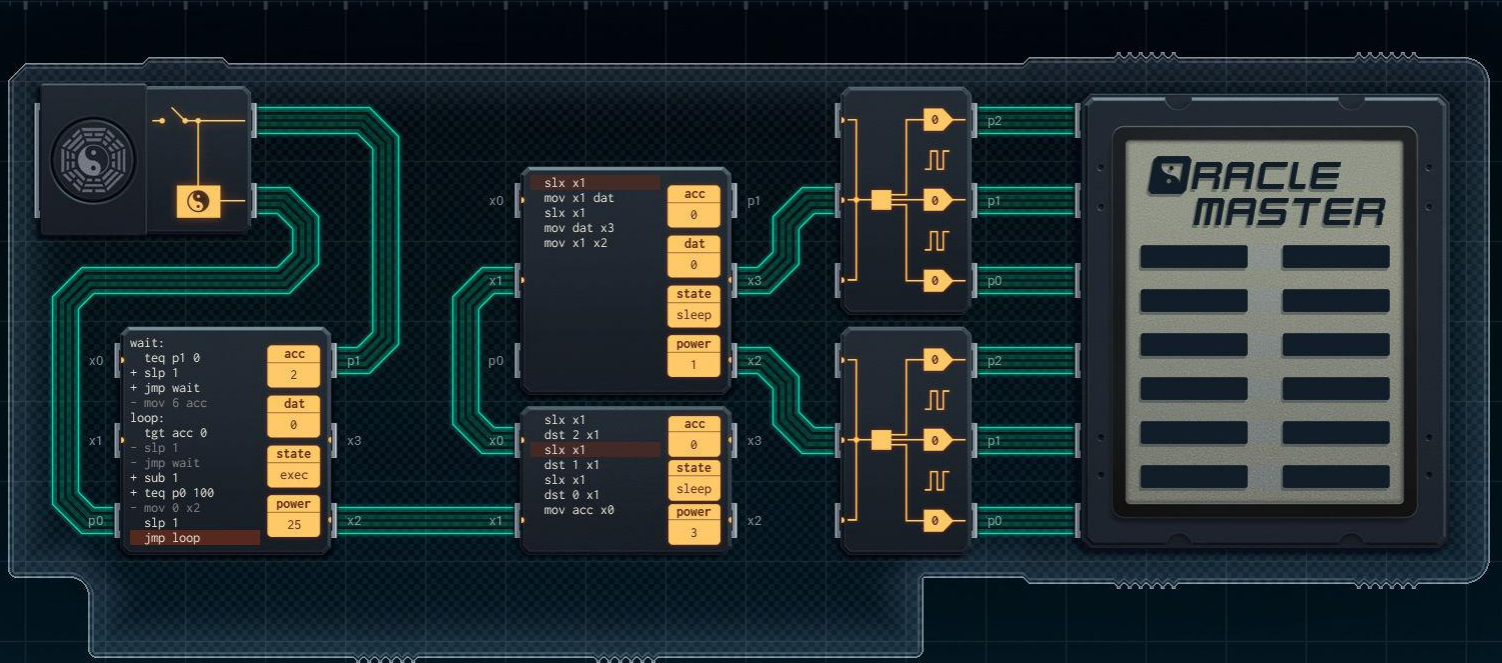




- Perhaps half Homo Sapiens, half human-ape. "Weird" hybrids are cool af and a lot less weird, and more cool than human-humans
- The magnificent human-tardigrade

Game suggestions of the day: Programming games


HUMAN
RESOURCE
MACHINE



Write code in a compact and powerful assembly language where every instruction can be conditionally executed

Recap of last time

How do custom Scripts work in Unity?

- Scripts are executed in Unity as Behaviours, which have to be attached to a GameObject present in the scene

- To attach a script drag-and-drop it on the object or add it in the Inspector
- The : MonoBehaviour part is important, we'll see why later

What is a Type?

You need to use different types for different information you want to store, the most common ones are:

- Integers: int
- Real numbers: float or double (for bigger numbers)
- True or False statements: bool
- Sentences: string

What are Variables?

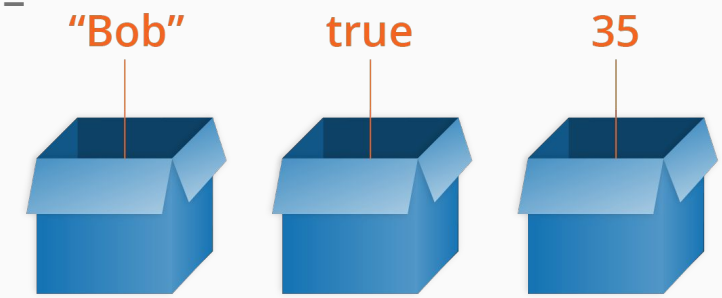
A variable is a container for values

You define (or initiate) a variable by:

```
TYPE NAME;        //e.g: int myVariable;
```

You can put values in a variable using the operator =

```
myVariable = 5;
```



What is a Function?

Functions allow you to write small reusable parts of code, which take some input and return an output.

```
int Double(int number){
	int result;
	result = number * 2;
	return result;
}
```

How does an if statement work?

Allows you to change the execution of the program depending on a condition. The else part of the statement is optional.

```
if (myVariable == true){
	//Code to execute when the condition is true
}
else{
	//Code to execute when the condition is false
}
```

Do you remember how Start() and Update() work?

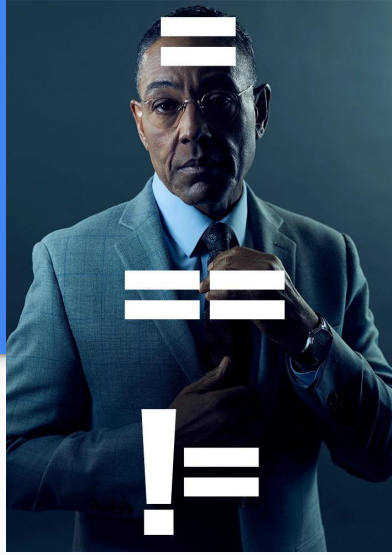
Start(): this function is executed only once when the object to which the script is connected is activated.
Use for initializations.

Update(): this function is executed every frame of the game execution, this is where you will write most of your code.

Recap over, any other questions?

Before we continue, let’s introduce some other useful concepts

Boolean operators



In IF statements you will need to use boolean operators, these are:

- == equal: 5 == 5 -> true, 4 == 5 -> false
- > greater than: 4 > 4 -> false, 5 > 4 -> true
- < lesser than: 4 < 4 -> false, 3 < 4 -> true
- >= greater or equal than: 4 >= 4 -> true, 5 >= 4 -> true, 3 >= 4 -> false
- <= lesser or equal than: 4 <= 4 -> true, 5 <= 4 -> false, 3 <= 4 -> true
- != not equal: 5 != 5 -> false, 4 != 5 -> true
- ! not: !false -> true, !(5 == 5) -> false
- && and
- || or

Arrays

Arrays are collection of variables, especially useful to store related information, for example stats.

```
TYPE[] arrayName = new TYPE[NUMBER_OF_ELEMENTS];
```

```
int[] stats = new int[6];
```

Arrays

You can initialize arrays when you declare them

```
int[] stats = {14, 18, 16, 13, 18, 14};
```

And you can access specific values:

```
Debug.Log(stats[2]); //prints “16”

stats[2] = 3; //now stats will contain {14, 18, 3, 13, 18, 14}
```

Note that arrays indexes start from 0

While loop

Loops help you repeat a piece of code multiple times. The while loop continues until a condition is not met anymore.

```
while (CONDITION){
CODE
}
```

```
while(enemies < 5){
        spawnEnemy();
        enemies = enemies + 1;
}
```

For loop

The for loop is similar to the while loop but is (generally) used when you know how many times to execute the loop.

Note that the INIT instruction is executed before the loop (only once!), while the INCREASE instruction is executed after the execution of every loop.

```
for (INIT; CONDITION; INCREASE){
CODE
}
```

```
for(int i = 1; i <=5; i++){
        spawnEnemy();
}
```

Collaborative programming

How to handle your projects

The problems:

- You’re working on on your laptop and then you want to continue on your desktop, how do you move the project?
- You’re working with some teammates, how do you make sure everybody has the same version, and changes you make aren’t conflicting?

Version Control

The solution:

- Using a VCS (version control system), for example Git

Why?

- Can keep track of all changes
- Allows you to update your local version by only getting the changes (so you don’t have to download the entire thing)
- Facilitates a lot collaboration, and deals with conflicts automatically (***most of the time***)

How do you use git?

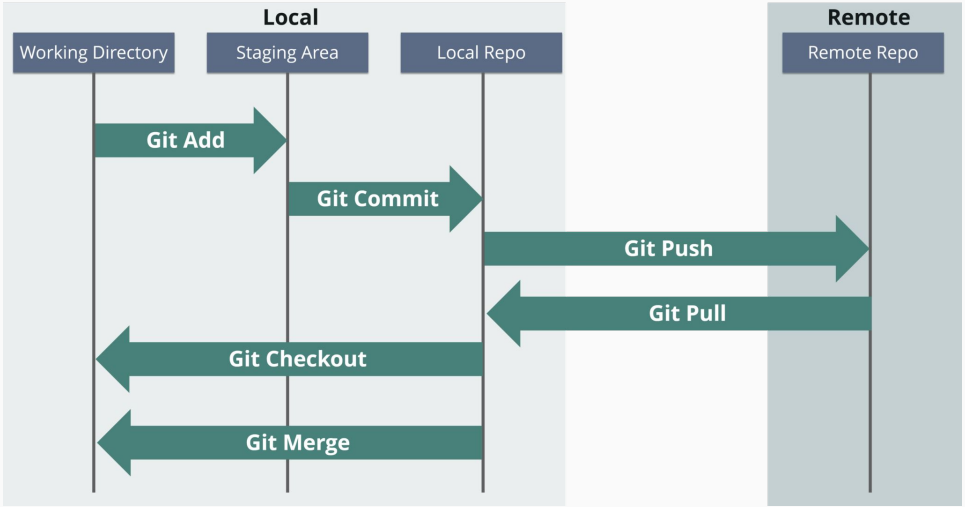
Things you need:

- A repository server (where your project files will live so you can download them whenever)
  - Github, bitbucket, etc.
  - ...or you self-host on your server, if you have one
- Software on your machine to access it
  - With GUI there's plenty of options: Github Desktop, Sourcetree, gitExtensions (I use this), and many more
  - ...or you can just install git and use the command line
- Unity has a version of this built-in, but with limitations
  - https://unity.com/solutions/version-control

Let’s try to see how it works :)

Basic workflow

1. Set up your repository
2. Pull the existing code (if any)
3. Code!
4. Commit changes to the local repository
5. (optional) Push to the remote repository (send your changes to the server)
6. Go back to 2 or 3 :)

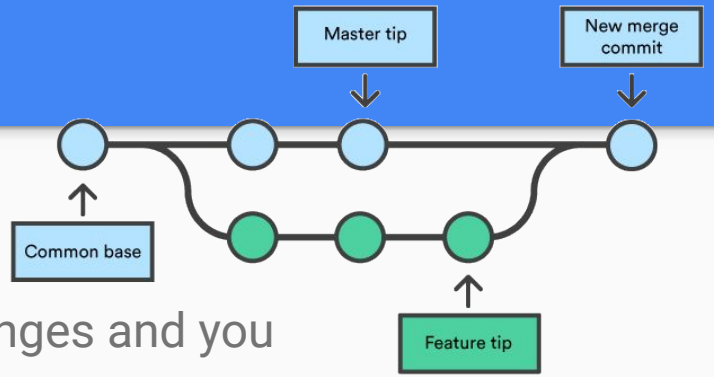


Some advice with Unity

- **Use a .gitignore file!**
- Try to only have one person working on a scene at any time, scene merging is not great
- Use small “development scenes” to work on your part and only add to the main scenes when you think things are working well enough
- Communicate with your teammates what you’re working on :)

More detailed advice later in the course!

Cool stuff you can do



Using branches:

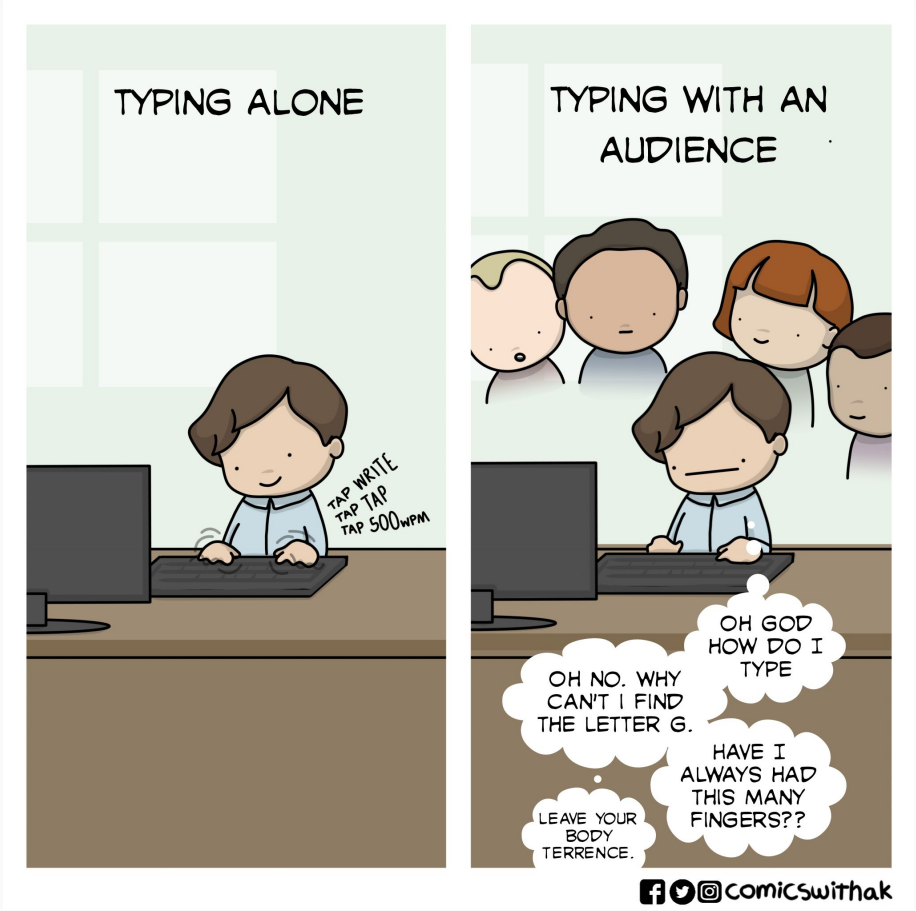
- when you want to make some larger changes and you want to make sure things don’t get broken you can make a branch
- If things go well “merge” the branch back in the main one, otherwise your original code is still fine in the main branch :)
- Also very nice when you work with others!

More suggested video tutorials

https://learn.unity.com/project/beginner-gameplay-scripting

8. Awake and Start
9. Update and FixedUpdate
11. Enabling and Disabling components
12. Activating GameObjects
13. Translate and Rotate
17. GetButton and GetKey
18. GetAxis
19. OnMouseDown
20. GetComponent
21. Delta Time

Let’s continue our cookie clicker!



Warm-up step:

- IF Statements - make something happen only on condition, like every 10 cookies
- Let’s actually click the cookie! Can you figure out how to do it by looking at the documentation? (Tip: look for the “MonoBehaviour” class)

Numbers point to relevant videos at
https://learn.unity.com/project/beginner-gameplay-scripting

Making a Worker to click the cookie for us:

- Let’s create a new script
- Make the counter increase by 1 each frame (8 and 9)
- We will need to access the score variable in the original Cookie script (20)
- Enable the worker when we have clicked 10 cookies (11)

Numbers point to relevant videos at https://learn.unity.com/project/beginner-gameplay-scripting

Let’s “juice” it up a bit:

- Let’s add some movement (spin) to the cookie (13)
- Let’s try to add a light too
  - Use an array to have a list of color we can use

Numbers point to relevant videos at
https://learn.unity.com/project/beginner-gameplay-scripting

Next lecture

Start doing the Roll-a-ball tutorial:
https://learn.unity.com/project/roll-a-ball

You should at least finish part 3, so you should have the movement, camera, and play area ready for next time.