The thing that tends to get people bogged down when trying to make a game is the coding. But really, all that the coding does is give all of your characters rules for the world they exist in. Today, we are going to build a video game.
This is the tool we use to control our character in our game.
This is a Makey Makey. A Makey Makey is a small computer board that can plug into a USB computer port.
When the wires touch, it completes a circuit, just like flipping a switch or pushing a button.
When you plug wires into these spots you can send signals to the computer. But here's the thing. The computer just thinks this is a regular keyboard.

The computer thinks that you are typing A,S,D or F on a keyboard every time that circuit is completed.
This is the other side of the Makey Makey. What does THIS side of the Makey Makey look like?
An old school Nintendo controller! I could use this Makey Makey to play a video game. I could plug wires into this and then plug those wires into just about anything!
...like a banana or a piece of wood. Then by touching it, the keyboard thinks that you are typing normal keys.
What can you make your own game controllers out of?
Game Maker is a free software you can use to make games on a PC or Mac. You can start with drag and drop tools for a low barrier start, then introduce coding later on.
This is the backstage of the game. We will start by putting an enemy on a path.
Here is our list of folders. Remember, here are all of these folders that say ‘sprites’ and ‘objects’ and ‘rooms’, and these folders have all of the things that make up our game.
Next, open the ‘Object’ folder.
As you can see, inside our Objects folder are all of the different pieces of the game. Our hero, our enemies, the green spinning wheels of death, even the bricks for our walls.
But let's focus on opening the object folder for our enemy. This is the place where all of the rules for this enemy exist. Anything we want this bad guy to do we put in his profile, and we call those things his EVENTS.
In the first column you can see the word EVENTS. These are all of the rules that make up our baddie.
First, we have our lightbulb that means create. It means that this is what is going to happen when our game starts.
When our enemy is created, we want him to start doing something. In the actions column we give him an instruction.
Start walking on path #0. Do this over and over again for as long as the game is played!
But let's take a closer look at how that path is assigned.
We are back at all of our game folders.
Now we open the folder marked paths. Inside we can see where we drew the path for our enemy to walk.
Here are actually seeing the game. See the grid? This tells our game where we want all of our characters, all of our objects to be located. The grid is the same that you might see in your math class. You have an x axis and a y axis. Now, if we were making a first person 3D style game, there would be a z axis also, which would give the room depth.
This is the path that our little blue bad guy will walk on. Our enemy is going to walk down that path forever and ever and ever and ever. They are not going to chase you, or think for themselves. BUT, if you walk into their path and they bump into your hero, what will happen? They will destroy you!
Creating paths and actions is like telling a story. If the object enemy 2 is created then set on a path, it will continue on that path forever, unless you tell it to do a specific command at a certain instance.
But again, we have to tell the enemy to do that. Because remember, computers are really smart and powerful...
but they are also kind of dumb! Unless you tell them exactly what to do.
We are going back into our object folder for this same blue bad guy. Under events, we have our create lightbulb where the path is, right?
But UNDER that is the direction for what happens when the enemy bumps into the player one. When two things touch each other in the game world, we call it COLLISION. Does anyone know what that means? It’s when two things bump into each other, like a car collision. But in this case, it’s when any two things bump into each other, something else happens. See those two sided red arrows?
For example, what happens when Mario hits a question mark brick? THAT is collision.
Collision

These 2 arrows facing each other are the symbol for collision in GameMaker.
Collision!

If

obj_player

Then

If this bumps into that, then 'X' happens. It’s like we are telling a story about what is going to happen.
So in this case, our Event column says that when our blue bad guy collides with our player one,
It will destroy the instance, meaning it destroys our player one. But is that all? Nope, because we also have to tell the computer to restart the game. Otherwise, the game will just sit there, our little bad guys will keep walking their paths, but you won’t be able to play because your player is destroyed! So you have to restart the game, too.
Let’s add lasers!

Creating a laser sounds cool, right? But again, remember, computers are really smart.....
But they are also kind of stupid.
So instead, you have to go through all of the steps to create a laser. Here we are in the backstage again with all of our folders.
Open the OBJECTS folder
And then open the Object Player folder
Now we create the directions of our laser, if our laser position on the X axis is positive, we fire right. If it is negative, our laser fires left.

In this case, if its position is not positive, it is “Else.”

Remember, because our computer is smart but kind of dumb, it doesn't know “left” or “right,” it only knows positions on the X and Y axis.
The first box has us set the speed, let’s do it to 8, I know you’ll want to set it at 800 but let’s keep it slower for now. Set direction, click relative which means where the laser starts from is going to be relative to the object that creates it (The Purple square)
And that’s it, we’re done with the laser right?
We need collision! But this time, we need to make sure that when we set our collision, it destroys both the bad guy and the laser. The collision is set in our baddie folder.
ARE THE ENEMIES FIRING AT YOU?
No, they are just shooting straight ahead!
Let’s see how they know how to do that.
Here we are backstage again, this time looking at the object events for our enemy. Remember, these are all of the rules that make up our baddie. His “brains.”
First, we have our lightbulb that means create, right? Then we create a path for our enemy to walk on.
Now, up next we have something called a STEP EVENT. The best way to think about this is to imagine that some of the directions and rules that we give to our world aren't happening all of the time, they need reminders every so often. That is when we use a step event. A step event allows certain rules to pop up every so often and say ‘Don't forget about doing this, bad guy!’ So a STEP EVENT is just one of those reminders of the rules.
This rule here is actually CODE! So this says that our enemy laser isn't going to fire all of the time. Just randomly, since it doesn't have a button to fire like we do on our hero. Who can tell me what random means?
Is it like when you roll dice? The number could be different every time. How many sides does a dice have? Six, right? But if we played with six sided dice, then every time we landed on one, our bad guy would shoot. That would be a really expert game level. Way too hard.
Let's pretend we are dungeon masters in a role playing game instead, and our dice has 120 sides to it!
Super nerdy, right? If we have dice with 120 sides, our baddie will only shoot a laser when the dice lands on ONE. The more sides our dice has, the more random the laser becomes and the less it fires.
Let's see what it looks like if we had a TWO sided dice instead of 120. That's not even dice, it's like...

```java
if random(2) <= 1 {
    // do something
}
```
Flipping a coin. When it’s heads up, it fires. Heads down, it takes a break.
Want to see what it would look like if we had a reminder of 1 instead of 120? It might make the game too hard! This is like selecting a higher difficulty in a video game. And it’s just as easy as changing one number!
Take a look at the landscape of our game. Where our little spinning wheels of death are, where all of our platforms, our enemies, our hero. We are going to keep all of our elements the same, but we will be changing the way that everything looks.
We are going to change the sprites. Sprites are the costume that a character wears. Characters in games like Angry Birds, candy pieces in Candy Crush and blocks in Tetris are all made of Sprites. They are just the costume.
Mario, for example. Mario can run, jump, kick koopas, break bricks, but all of this is just the events in the Mario object. Those are just the rules. The guy with the moustache and the little red pants is just his SPRITE. It’s the costume that the Mario object wears. You could change Mario into ANY other sprite and he would still behave as Mario.
But because all of the rules that you put in place in the Object Events means the characters behavior will stay the same, regardless of what they look like. So we are going to change our sprites, since they all are just weird squares right now. First, our blue square baddies are going to be big crazy killer robots, cause that’s awesome.
Then, our hero purple square is going to be the coolest thing ever, a CHICKEN! If she was just this single image sprite, when we move her around, she would just be kind of sliding across the ground, like she was moonwalking! So instead we are going to make what is called a WALK CYCLE.
WALK CYCLE!

One last thing I want to mention before we get back to the game. Now this will make your sprite look a lot cooler than a bunch of lame boxes. We’re going to add a walk cycle so your character will be animated and not just a floating statue. You make a bunch of individual frames of each of the characters movements. The sprite will flip through them to animate.
Did you ever make a flip book? It’s exactly the same thing.
You can make your own characters and walk cycles at piskel.com for free.
NOW who wants to make a video game?
Thank you!