



Re-MakeCode the Classics: Py Hunter

Created by John Park



<https://learn.adafruit.com/re-makecode-the-classics-spy-hunter>

Last updated on 2024-06-03 02:47:59 PM EDT

Table of Contents

| | |
|---|----|
| Overview | 3 |
| • Parts | |
| Re-Make Py Hunter: Setup and Mechanics | 6 |
| • MakeCode Arcade | |
| • Py Hunter | |
| • Load the Code | |
| • On Start | |
| • BGM | |
| • Create Player Car Sprite | |
| • Health Bar | |
| • Instructions | |
| • Game Start | |
| • Trees | |
| • Cars | |
| • Enemy A is a Wacky Driver! | |
| • Oil Slicks | |
| • Freeze Rays and Smoke Screens | |
| Re-MakeCode Py Hunter: Collisions and Game Play | 18 |
| • Don't Bump Innocent Cars! | |
| • Enemies Bump Bystanders | |
| • Player Slams into Enemy | |
| • Hitting Hazards | |
| • Projectile Hits | |
| • Custom Health Meter | |
| • Game Over | |
| Update the PyBadge/PyGamer Bootloader | 24 |
| • PyBadge/PyBadge LC Bootloader | |
| • PyGamer Bootloader | |
| • Hardware Checks | |
| Load a MakeCode Game on PyGamer/PyBadge | 26 |
| • Board Definition | |
| • Change Board screen | |
| • Bootloader Mode | |
| • Drag and Drop | |
| • Play! | |
| Troubleshooting MakeCode Arcade | 30 |

Overview



In this guide, we'll Re-MakeCode the thrilling car-chase arcade game classic, **Spy Hunter!** But since its on a PyBadge or PyGamer we'll rename it (and tweak the design a bit) to create **Py Hunter**

We'll learn how to create an endless road level, high speed enemies, car bumping mechanics, smoke screens, freeze rays, a custom health meter, and more!

[Microsoft MakeCode Arcade](https://adafru.it/EQg) (<https://adafru.it/EQg>) is a **web-based** beginner-friendly code editor to create **retro arcade games** for the web and for microcontrollers.



Beta Zone: MakeCode Arcade is still in its beta. It might still have a few rough edges.

Parts

You can play your game in the browser, with no added parts, or get one of these great handhelds to play with real hardware on the go!



Adafruit PyGamer Starter Kit

Please note: you may get a royal blue or purple case with your starter kit (they're both lovely colors)What fits in your pocket, is fully Open...

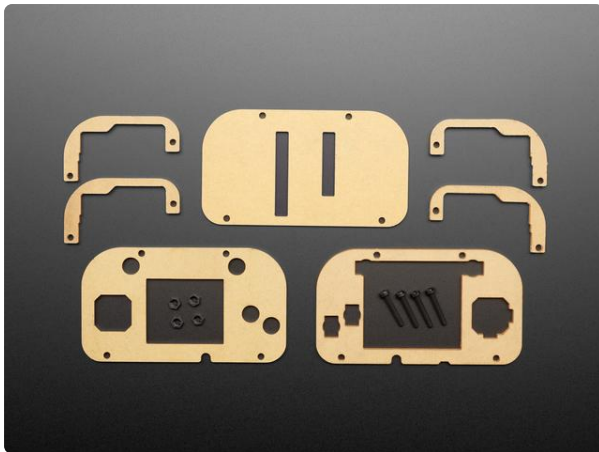
<https://www.adafruit.com/product/4277>



Adafruit PyGamer for MakeCode Arcade, CircuitPython or Arduino

What fits in your pocket, is fully Open Source, and can run CircuitPython, MakeCode Arcade or Arduino games you write yourself? That's right, it's the Adafruit...

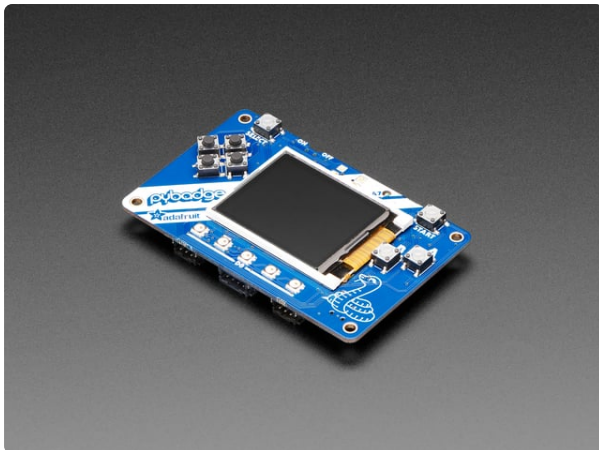
<https://www.adafruit.com/product/4242>



Adafruit PyGamer Acrylic Enclosure Kit

You've got your PyGamer, and you're ready to start jammin' on your favorite arcade games. You gaze adoringly at the charming silkscreen designed by Adafruit...

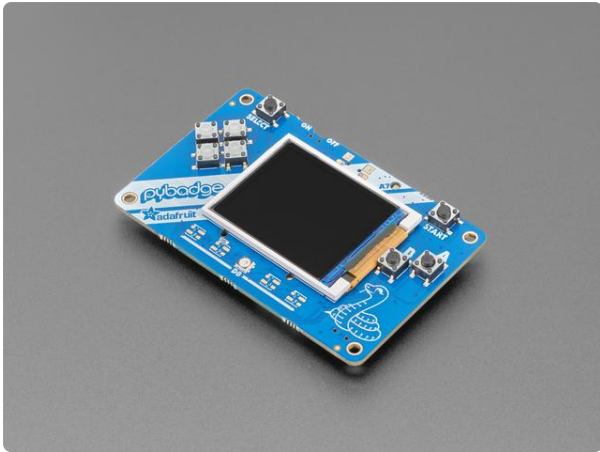
<https://www.adafruit.com/product/4238>



Adafruit PyBadge for MakeCode Arcade, CircuitPython, or Arduino

What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino? That's right, its the Adafruit PyBadge! We wanted to see how much we...

<https://www.adafruit.com/product/4200>



Adafruit PyBadge LC - MakeCode Arcade, CircuitPython, or Arduino

What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino even when you're on a budget? That's right, it's the Adafruit...

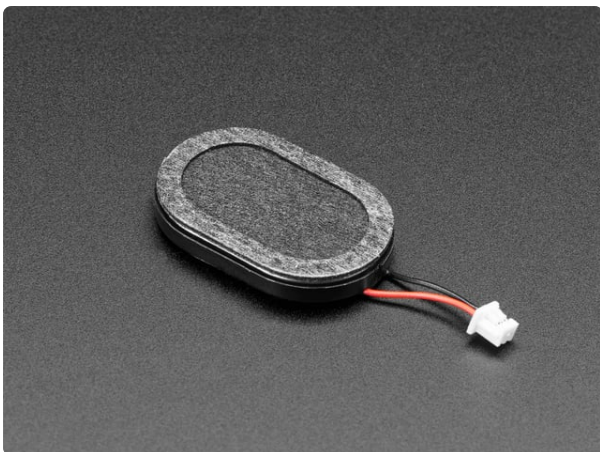
<https://www.adafruit.com/product/3939>



Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh

Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...

<https://www.adafruit.com/product/4237>



Mini Oval Speaker with Short Wires - 8 Ohm 1 Watt

Hear the good news! This wee speaker is a great addition to any audio project where you need 8 ohm impedance and 1W or less of power. We particularly like...

<https://www.adafruit.com/product/4227>



Pink and Purple Braided USB A to Micro B Cable - 2 meter long

This cable is super-fashionable with a woven pink and purple Blinka-like pattern! First let's talk about the cover and over-molding. We got these in custom colors,...

<https://www.adafruit.com/product/4148>

Re-Make Py Hunter: Setup and Mechanics

MakeCode Arcade

If you're not already familiar with the basics of MakeCode Arcade, [check out this guide \(https://adafru.it/Elc\)](https://adafru.it/Elc) on creating a character sprite and moving it with controls.

To start, open a new Chrome browser window (Chrome works best) and go to [MakeCode Arcade beta \(https://adafru.it/EQg\)](https://adafru.it/EQg).

These MakeCode Arcade guides are designed to take you through the fundamentals before tackling more complex games.

- [Pixel Art \(https://adafru.it/EOI\)](https://adafru.it/EOI)
- [Animation \(https://adafru.it/EOk\)](https://adafru.it/EOk)
- [Level Design \(https://adafru.it/EOj\)](https://adafru.it/EOj)
- [Sparky Invaders \(https://adafru.it/EYf\)](https://adafru.it/EYf)
- [Next Level Game Techniques \(https://adafru.it/EYg\)](https://adafru.it/EYg)

For another Re-Make that'll get you caught up on some intermediate-level techniques, check out [Re-MakeCode the Classics: Arkanoid \(https://adafru.it/E-o\)](https://adafru.it/E-o)

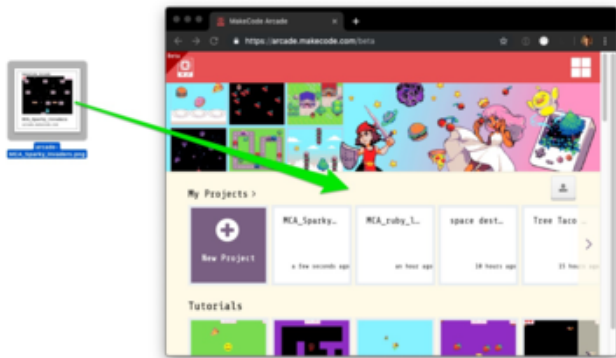
Only use the Google Chrome browser with MakeCode!



Py Hunter

We'll start by loading the finished version of MakeCode Arcade Py Hunter and try it out. Then, we'll have a look at some of the unique features and how they work.

Start by launching [MakeCode Arcade beta](https://adafru.it/EQg) (<https://adafru.it/EQg>) using the Google Chrome web browser. Then, download the **Re-MakeCode-PyHunter.png** file above by right-clicking on the image above and saving it to your computer.



Load the Code

This is a special .png file that contains not only an image, but the entire game is embedded in it as well!

Simply drag it from the location to which you saved the image on your computer (such as the desktop as shown here) onto the Chrome browser window that is already running MakeCode Arcade (MCA). Note that the image in this graphic is of a different game, but you'll be dragging the Py Hunter png file.

This will open the code into the MCA editor.



If you're ever unsure where a MakeCode block comes from, you can often find it by matching the block's color to a category on the left side of the editor. You can also use the handy search function!

Have a go at the game now so you will have a feel for how it plays before we look at the individual elements.

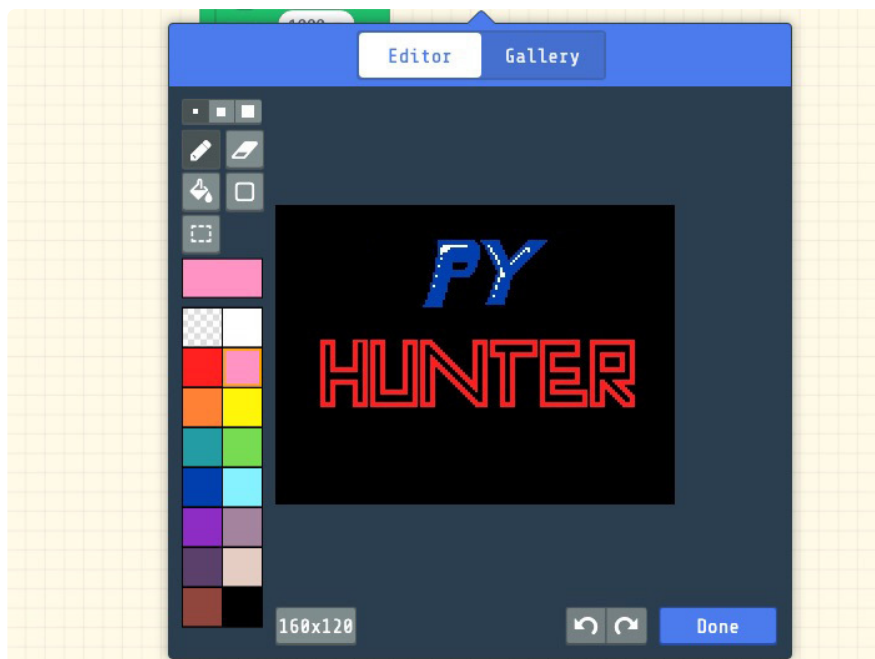
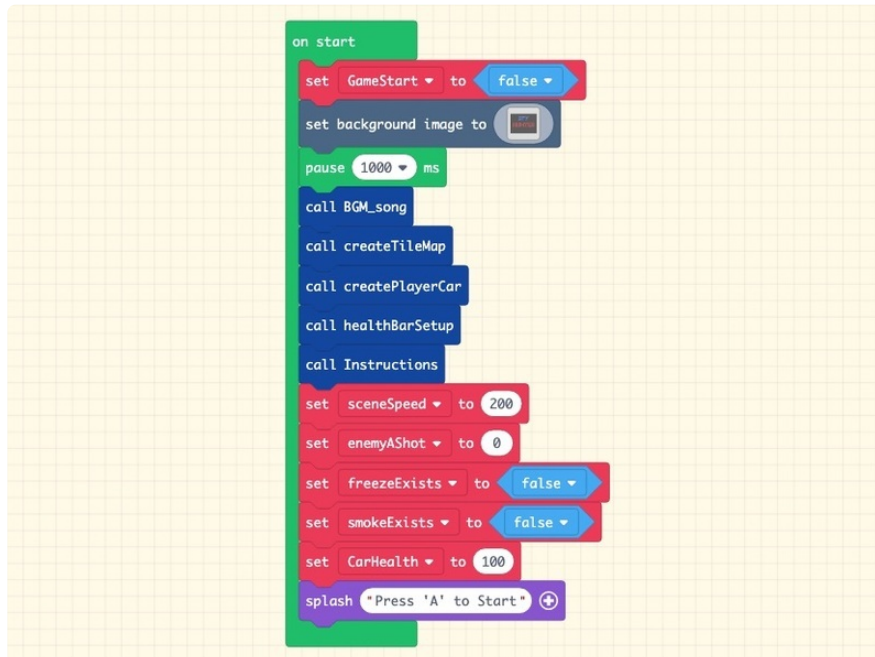
When it starts you'll see the splash screen title graphic and the theme song will play.

Then read the instructions and play! You'll use the d-pad to move the car and the A and B buttons to fire the freeze ray and smoke screen emitters.

Avoid bumping into innocent bystander cars (they're red) while shooting and bumping the bad guys. Don't hit the trees or oil slicks!

On Start

Here's the first block set which will run when the game loads, the **on start** loop.



```

function BGM_song
  call BGM_bass
  call BGM_bass
  play tone at High A# for 2 * beat
  play tone at High A# for 1 * beat
  play tone at High A# for 1/2 * beat
  play tone at High A# for 1/4 * beat
  play tone at High G for 1/4 * beat
  call BGM_bass
  play tone at High A# for 2 * beat
  play tone at High A# for 1 * beat
  play tone at 1396 Hz for 1/2 * beat
  play tone at 1308 Hz for 1/4 * beat
  play tone at 1108 Hz for 1/4 * beat
  call BGM_bass

```

```

function BGM_bass
  play tone at Middle C for 1/2 * beat
  play tone at Middle C for 1/2 * beat
  play tone at Middle C for 1/2 * beat
  play tone at Middle D# for 1/2 * beat
  play tone at Middle C for 1/2 * beat
  play tone at Middle D# for 1/2 * beat
  play tone at Middle D# for 1/2 * beat

```

Game Start & Splash Screen

The first thing that happens is we set a variable called **GameStart** to **false** -- this will be used later to determine when we can allow the other cars to start moving, and takes the safety off of the player car's fire buttons!

Then, we set the background image to the game's title splash screen, wait one second, and then start playing the background music (BGM) using the **call BGM_song** function.

```

function createTileMap
  set background image to [Image]
  set tile map to [Map]
  set tile [Color] to [Color] with wall OFF
  set tile [Color] to [Color] with wall OFF
  set tile [Color] to [Color] with wall OFF

```

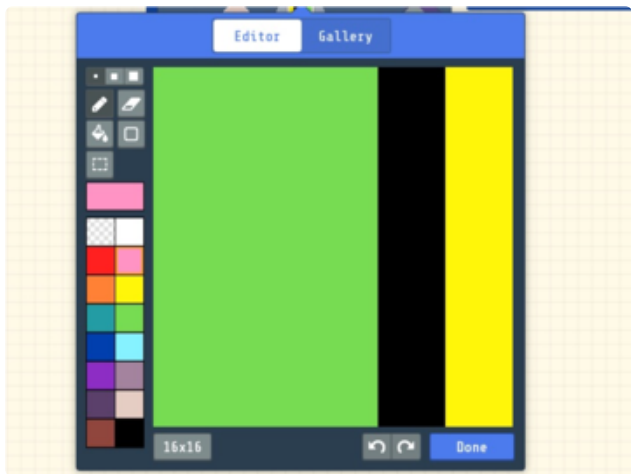
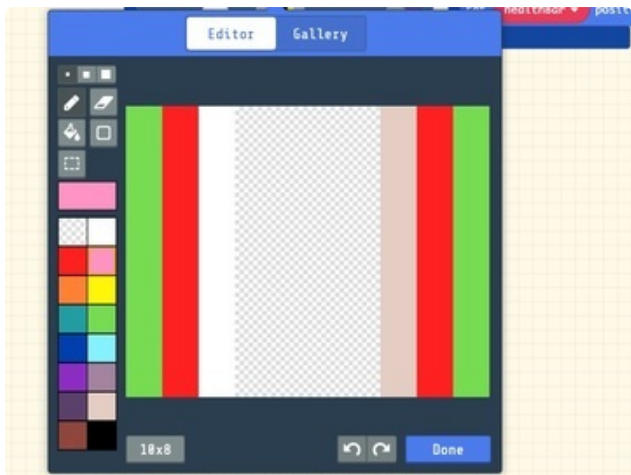
```

function healthBarSetup
  set healthBar to sprite [Image] of kind Meter
  set healthBar z (depth) to 1
  set healthBar position to x 1 y 16

```

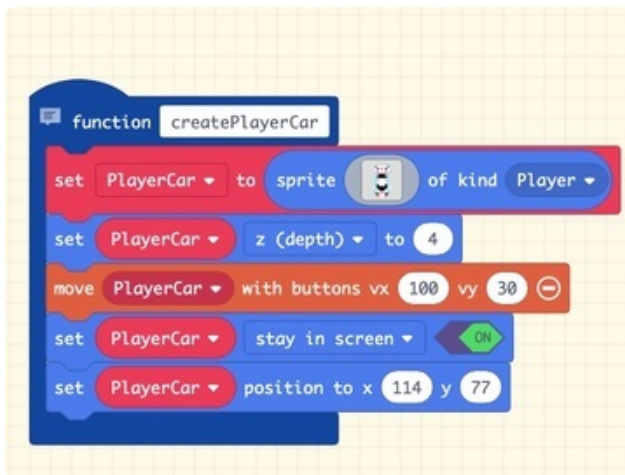
BGM

As we've done in previous MakeCode Arcade games, we'll use functions to organize different repeating sections of the music.



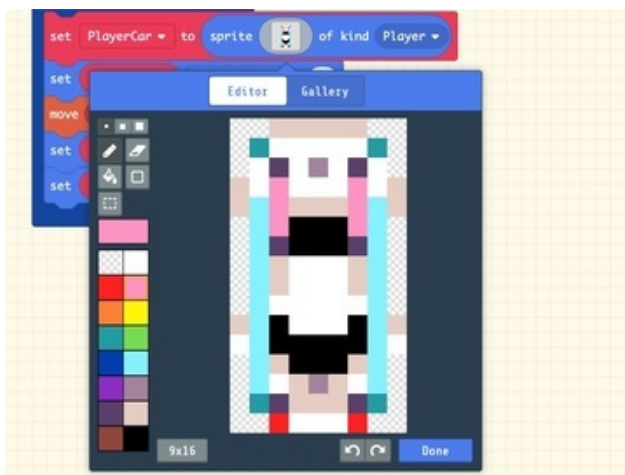
Tile Map

Then, we'll create a tile map and tiles to define the game screen. Note, these will be plain colors so we don't notice they aren't moving! We'll use moving trees to give the scene a sense of motion.



Create Player Car Sprite

Create the player's car sprite with these settings for control, position, z-depth, and the stay in screen parameter as shown.



The sprite graphic is a cool, white "Interceptor" sports car, just like the original!



Health Bar

The next function sets the custom health bar. Rather than use the default heart counter of MakeCode Arcade, we'll make a meter graphic as a sprite with ten divisions on it. We'll also set the z depth to 3 so that it remains above any other objects that occupy the same space, such as trees.

We'll place the position to the upper left corner. Later we'll create a function to change the health bar graphic depending on the player's health status.



Instructions

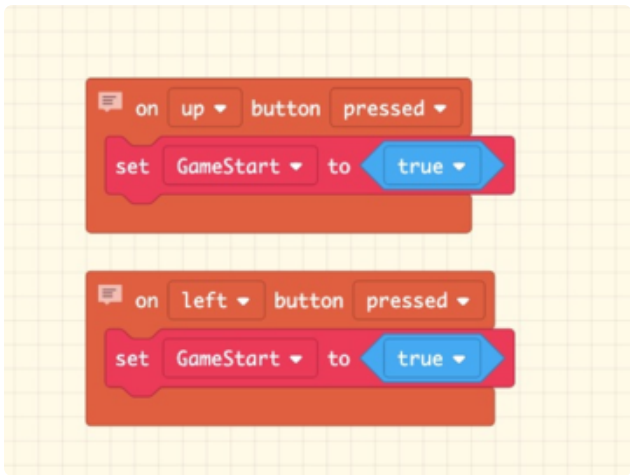
We'll put all of the starting instructions into **show long text** blocks. Each one requires the player to press 'A' to advance, and we can choose where on the screen to display each line.

The last thing we'll do in the setup is create and set some variables we'll use later, and use the **splash** block to have the player press 'A' to start the game.

These are the variables:

- **sceneSpeed = 200** is used to set **Y velocity** of trees & oil slicks, as well as derive car speeds
- **enemyAShot = 0** used to change behavior of enemy cars when shot
- **freezeExists = false** state of the freeze ray projectile
- **smokeExists = false** state of the smoke screen projectile
- **CarHealth = 100** player car health



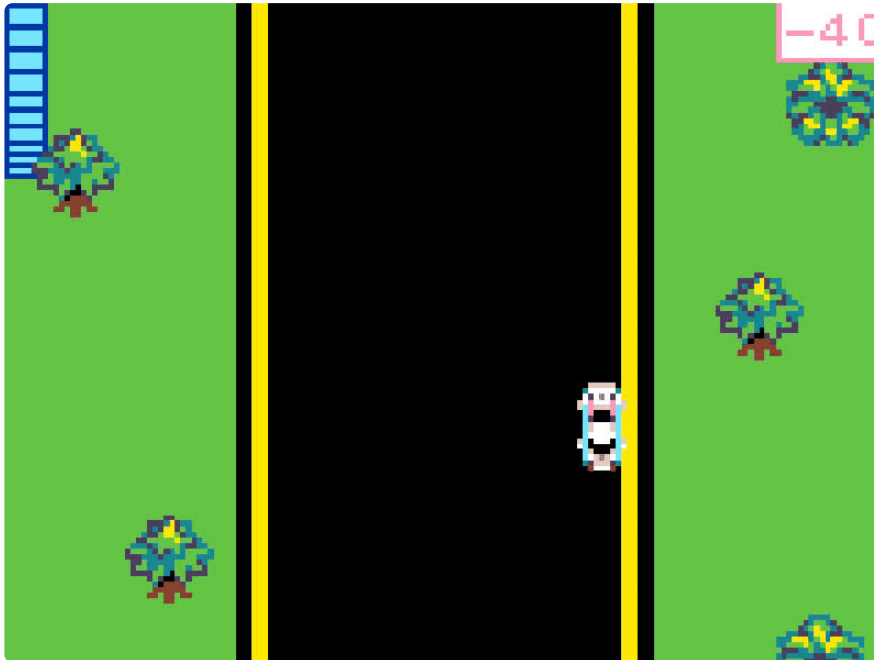


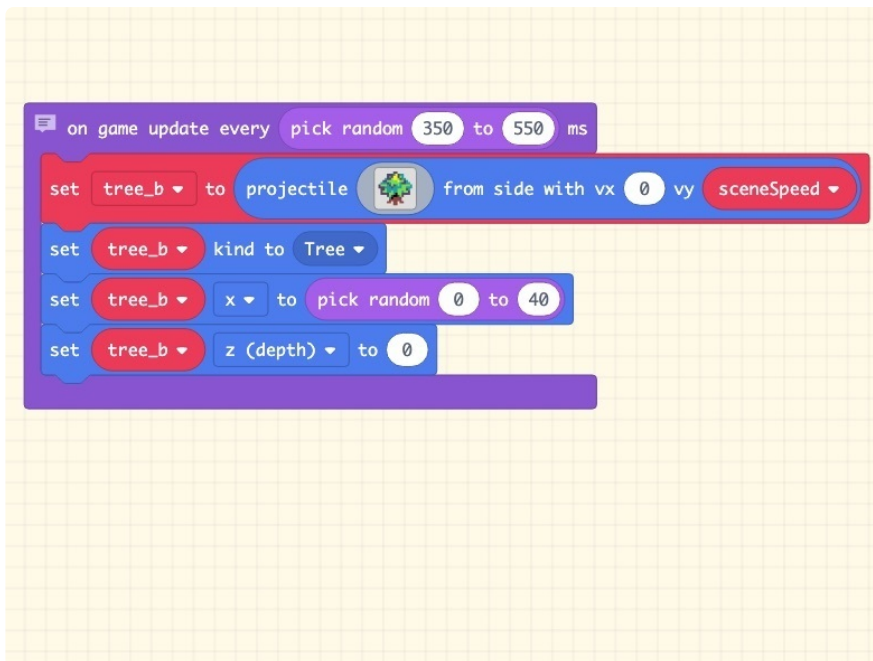
Game Start

We want to give the player a moment to orient themselves before the onslaught of cars begins! So, until they drive the car to the left or forward, the **GameStart** variable is still set to **false** and the other cars won't appear.

Once the player presses either **up** or **left**, the **GameStart** variable is flipped to **true**.

Trees





The moving trees and bushes will give the scene its sense of speed. Here's how we'll create them.



We'll make one tree and then duplicate the process four times with different parameters.

First, we'll start with a **on game update every** block and randomize it a bit so the trees don't appear to be moving in a set interval.

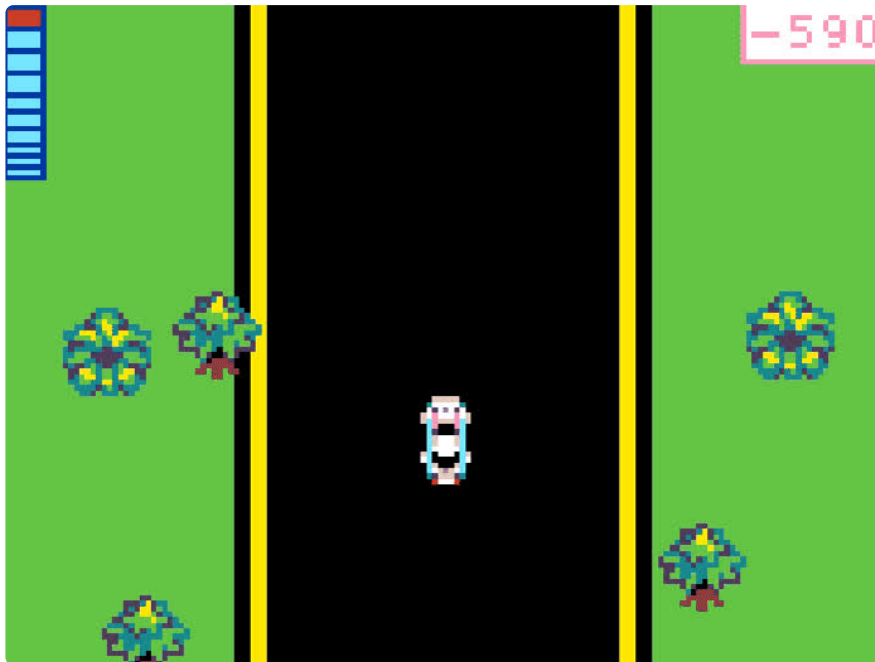
Then, we'll create a tree sprite as a projectile with all of it's motion on the vertical axis with the **vy (velocity Y)** set to the **sceneSpeed** variable.

We'll create a new sprite type named **Tree**, so we can later test for collisions with the cars and this type of sprite uniquely.

We'll have the start position on the **x-axis** be a **random** placement between either **0-40** for the left side of the road or **130-150** for the right side.

Then, place the trees at a **z-depth** of **0** so that other sprites appear on top of them.

Cars



```
on game update every pick random 4000 to 6000 ms
if GameStart == true then
  set enemyA to projectile from side with vx 5 vy sceneSpeed * 5
  set enemyA_posX to pick random 00 to screen width 60
  set enemyA kind to Enemy
  set enemyA x to enemyA_posX
  set enemyA z (depth) to 3
  set enA_exists to 1
  set enemyAShot to 0
```

Here's how the enemy and innocent bystander cars work.

First, we'll use a randomized time interval for **on game update** so they don't spawn at predictable times.

Next, we'll check the **GameStart** variable we set at the beginning to see if the player has moved their car left or up, signalling the beginning of game play.

Then we'll generate the car as a projectile sprite with a velocity that is a fraction of the **sceneSpeed** variable. This means that if you adjust the **sceneSpeed** the car speed will also adjust.

```
on game update every pick random 4000 to 6000 ms
  set enemyA to projectile from side with vx 5 vy sceneSpeed * 5
  set enemyA_posX to pick random 00 to screen width 60
  set enemyA kind to Enemy
  set enemyA x to enemyA_posX
  set enemyA z (depth) to 3
  set enA_exists to 1
  set enemyAShot to 0

  set enemyB to projectile from side with vx 5 vy sceneSpeed * 5
  set enemyB_posX to pick random 00 to screen width 60
  set enemyB kind to Enemy
  set enemyB x to enemyB_posX
  set enemyB z (depth) to 3
  set enB_exists to 1
  set enemyBShot to 0
```

The horizontal **x position** of the **enemyA** type of car is a random value somewhere in the area of the road, while **enemyB** starts out at the same **x position** as the player's car at the time it is spawned.

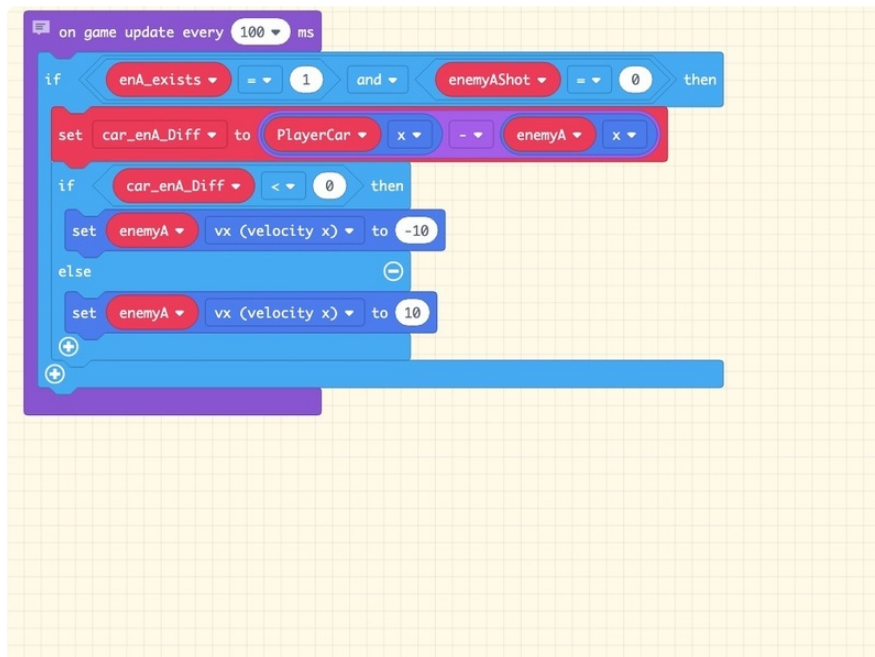
The enemy cars are set to type **Enemy**, while the **bystander** cars are of type **Bystander**. Again, just like with the trees, we can test collisions more easily later if we have unique types for the different sprites.

The **z-depth** of the cars are set to 3 so they appear above things such as oil slicks.

The **enemyAShot** variable is created and set to **0** -- this will be used later to determine the moment of these cars when they're shot.

Enemy A is a Wacky Driver!

You'll notice that the **enemyA** cars try to drive into the player's car no matter where you steer. Here's how this is done:

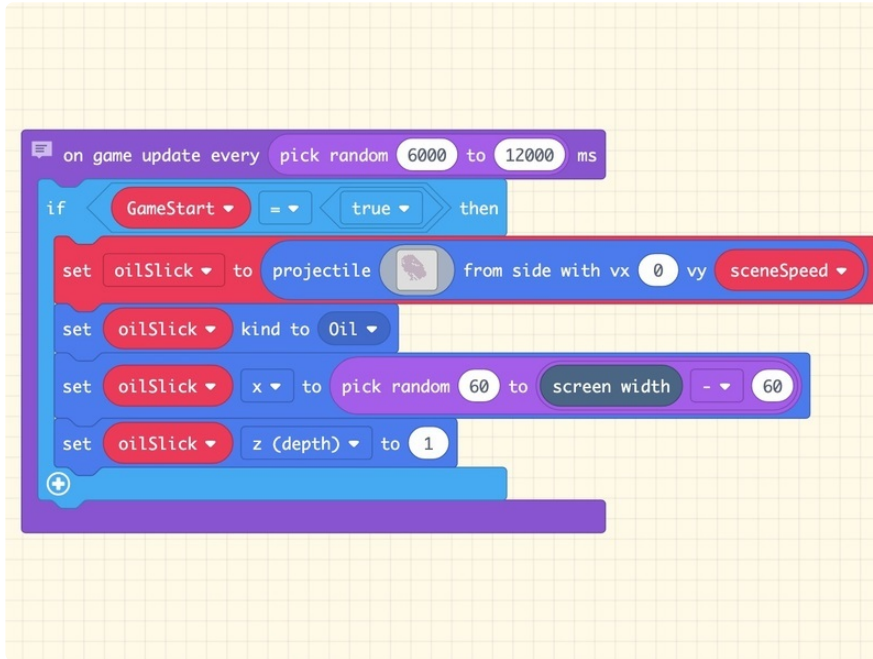


Every 100ms the **on game update** block checks to see if there's an enemy A car in the scene with the **enA_exists** variable, and it also checks to see if the car has been shot. If not, then it sets the **car_enA_Diff** variable to the **PlayerCar x position** minus the **enemyA x position**.

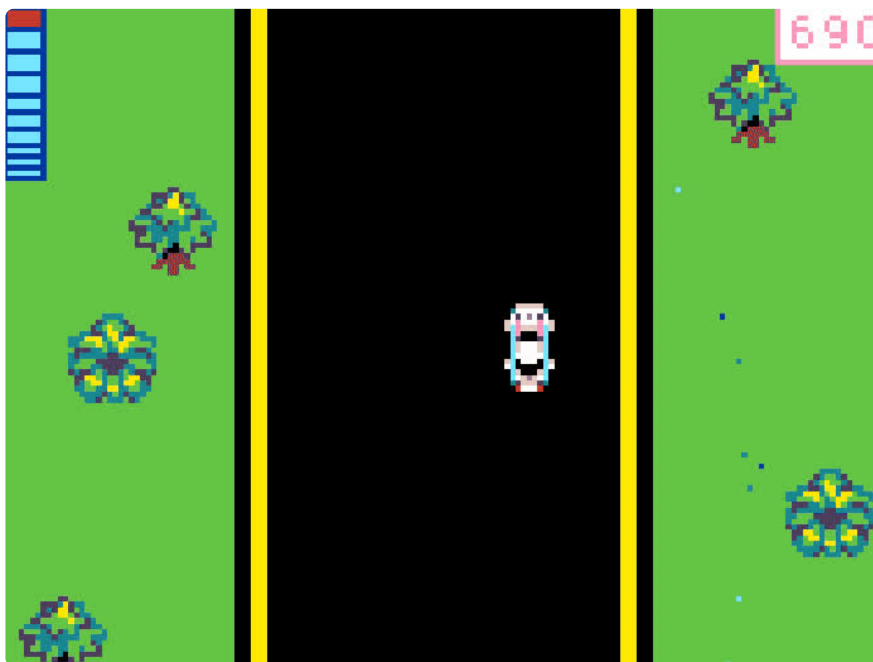
If the difference is less than 0 -- meaning the player is to the left of the player car -- then the enemy's velocity on x is set to -10 to steer it toward the player car, and to 10 if it's on the other side.

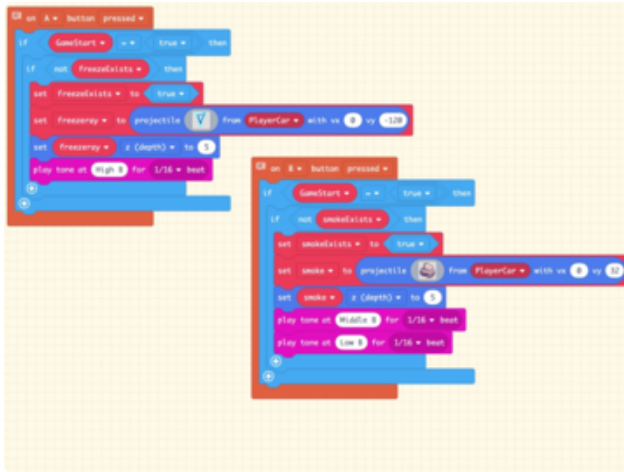
Oil Slicks

Just to add one more hazard to the scene, we'll also spawn oil slicks! These are handled in nearly the same way as the trees -- they're **projectiles** that move at the **sceneSpeed** (so they appear to be stuck to the road). They're of type **Oil** so that we can test overlap collisions against them as a unique type later.



Freeze Rays and Smoke Screens





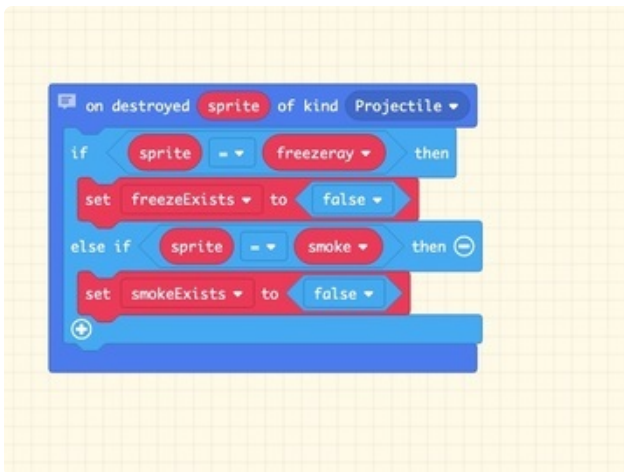
The player can shoot Freeze Rays and Smoke Screens in order to defeat enemy cars.

When the **A button** is pressed, we check to see if the **GameStart** variable is true (otherwise the player could shoot the splash screen!).

Next, we test to see if a freeze ray already exists to prevent the player from spamming the fire button. The **freezeExists** variable stores this state.

If we're in the clear, then we flip that variable, and shoot a **freezray** projectile "forward" with -120 velocity on the y-axis.

We'll also play a tone for a satisfying beep with each shot.



The smoke screen works the same way but with the **B button** and it travels slowly in the positive y direction, behind the car.

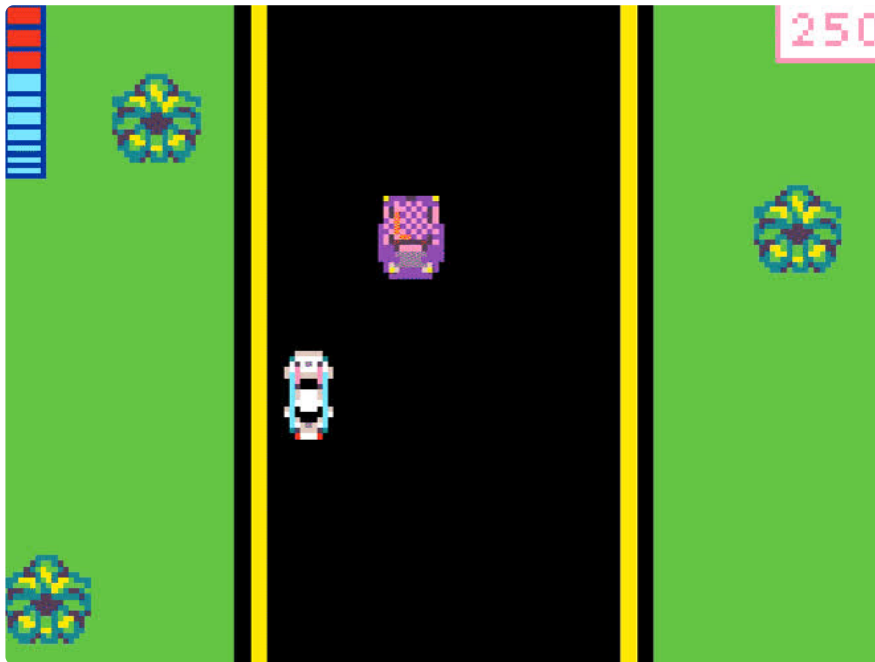
In order to know when we can allow another shot to be fired, the **on destroyed sprite of kind Projectile** block does a couple of checks against the name of the sprite. If it is **freezeray** or **smoke** then their respective **Exists** variable gets flipped back to **false**, allowing them to be fired again.

Re-MakeCode Py Hunter: Collisions and Game Play

Most of the events in the game come down to one thing -- collisions. When a freeze ray hits and enemy, the play bumps into a bystander car, or runs into a tree, all of these things are collisions.

MakeCode Arcade determines when collisions occur by using the **sprite overlap** blocks. We can set them to activate when all the different types of collisions happen, and then run other blocks depending on the pairings. For example, here's a simple one: when a player car sprite overlaps an innocent bystander car sprite, we'll create a

bump effect by moving the player over six pixels in one direction and the bystander 10 pixels in the other direction.



Don't Bump Innocent Cars!

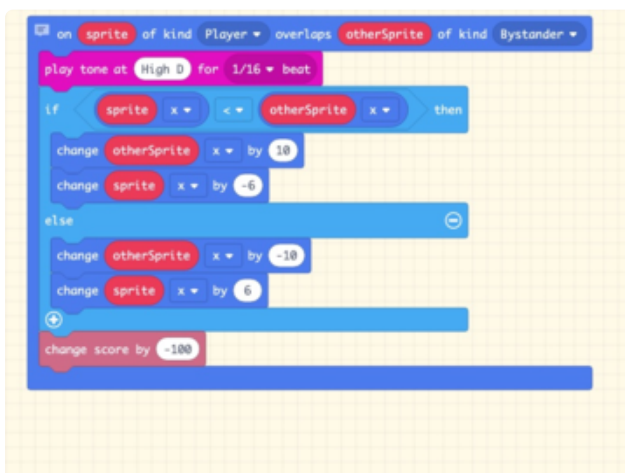
Here the `sprite` of kind `Player` overlaps `otherSprite` of kind `Bystander`.

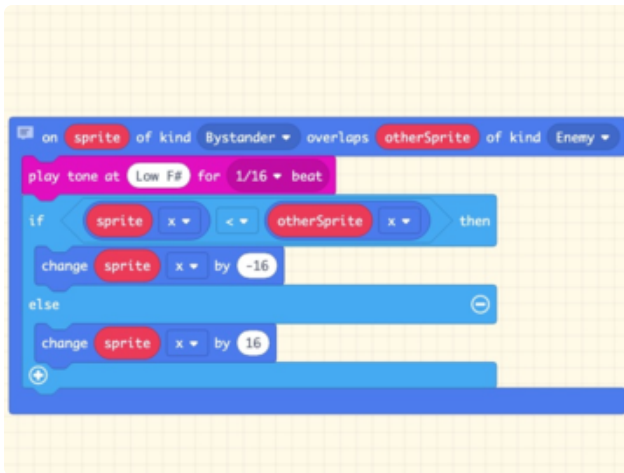
We'll play a short tone to indicate the contact.

Then, we check to see if the player car's **x position** is less than the bystander car's **x position**. If so, this means the player is to the left of the bystander car, so we'll move the bystander (`otherSprite`) **10** pixels on **x** and the player (`sprite`) **-6** pixels on **x**.

If the player's **x position** is greater than that of the bystander car's, then the player is to the right of the bystander and we send them in the other directions.

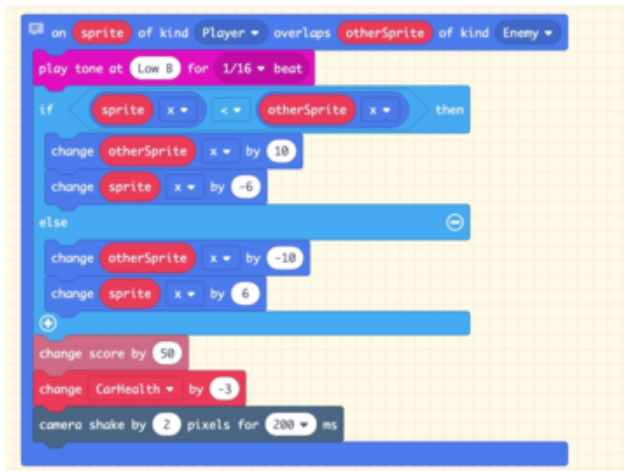
Finally, we penalize the player by lowering their score by **-100** points.





Enemies Bump Bystanders

Here you can see a similar collision setup for when **Bystander** sprites overlap **Enemy** sprites. In this case, only the bystander car is moved to the side. This makes the bad guys seem even badder.



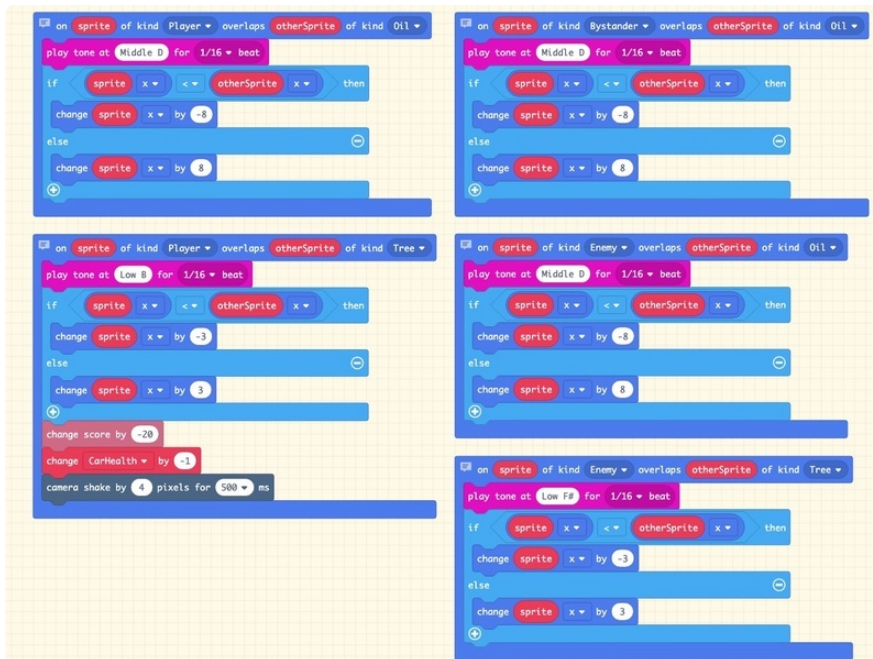
Player Slams into Enemy

When the player bumps into an enemy we use a similar setup as before. However we also do the following:

- add 50 points to the player's score
- change the **CarHealth** by -3 points
- shake the camera for hard hitting impact!

Hitting Hazards

Another type of collision to account for are the trees and oil slicks. These are similar to the car-on-car sprite overlaps.



Projectile Hits

When the freeze rays or smoke screens hit an enemy car or bystander car, we'll have them run off to the side of the screen.

If it's an enemy car, the player will gain 300 points; if it's an innocent bystander car, they'll lose 300 points.





Here's how this works:

- check for **sprite of kind Projectile overlaps otherSprite of kind Enemy** (the Bystander loop is similar, but we'll just go over the details of Enemy)
- Play a tone
- if **sprite = freezeray** then destroy the freezeray sprite, and start a **cool radial effect** on the **otherSprite**
- **change CarHealth by 2** to give the player some health back

Run off the road

Next, we check if the Enemy sprite is on the left or right half of the screen by using **if otherSprite x > screen width / 2**

The enemy is then launch off to the appropriate side at the speed of the scene with **set otherSprite velocity to vx 60 vy sceneSpeed**

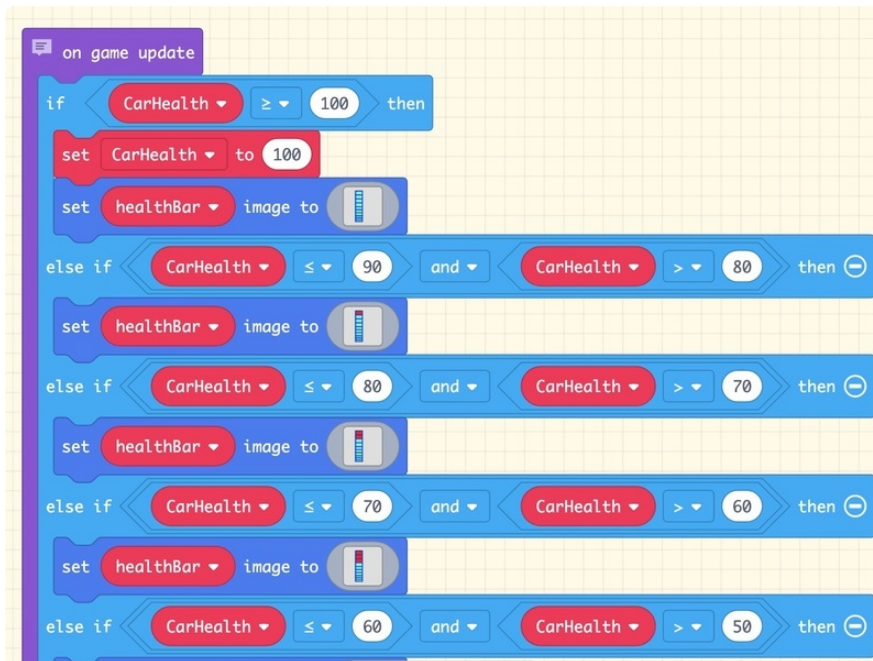
Since the **enemyA** car already has a regularly updating loop using **vx** to steer it toward the player car, we need to flip the **enemyAShot** variable to **1**, which prevents the steering from activating, allowing it to careen off the road.

This same process happens with the **smoke** sprite type, but using the **warm radial** effect instead.

Custom Health Meter

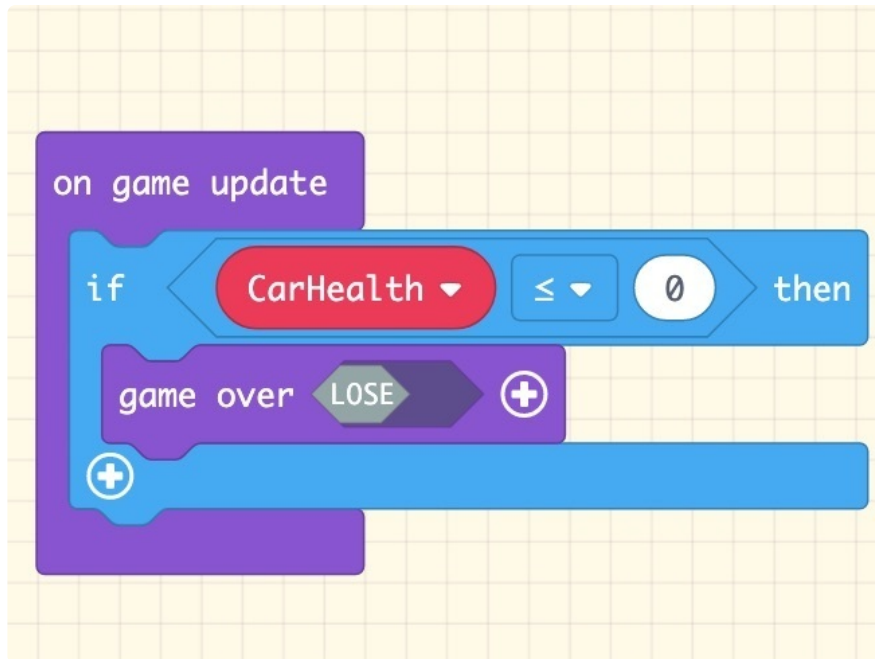
We'll get pretty fancy with the user interface (UI) here and create our own custom health meter. The player starts with 100 points of health, each bump with an enemy car or tree removes a few points, and each successful hit of an enemy adds back a few points.

In order to display this visually, we'll use the custom meter graphic and a number of **if...if else** logic tests -- as the health reaches each 10 point threshold we'll display an associated version of the graphic.



Game Over

When the **CarHealth** value reaches **0** it's game over!

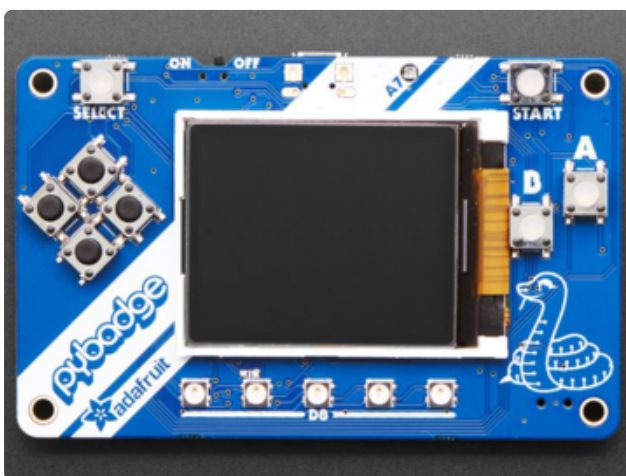


Now, you can play the game in your browser, upload it to the PyGamer or PyBadge, or even start making your own modifications to the game!

Update the PyBadge/PyGamer Bootloader

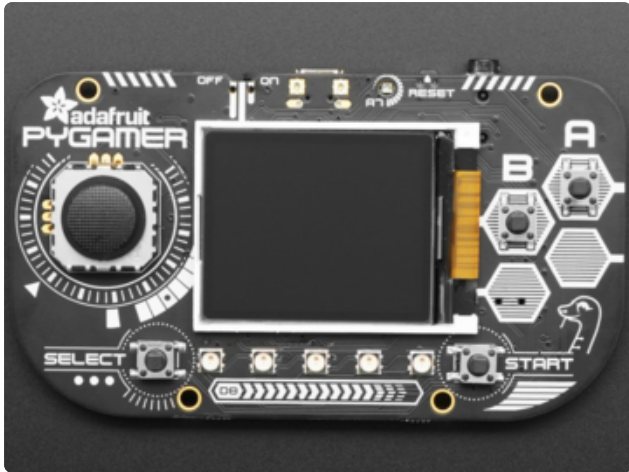
You are at the bleeding edge of handheld, open source, game playing hardware and software, what with your PyBadge/PyBadge LC or PyGamer! Congratulations! It's fun and exciting! It is also changing and improving all the time, so please update your bootloaders before proceeding to put your MakeCode Arcade games on the board!!

Among lots of other reasons, update the bootloader to prevent a problem with MacOS 10.14.4, to fix button problems, and get the thumbstick to work!



PyBadge/PyBadge LC Bootloader

If you have a **PyBadge** or **PyBadge LC**, please go to this page for instructions on updating the bootloader. (<https://adafruit.it/EWI>)



PyGamer Bootloader

If you have a PyGamer, please go to this page for instructions on updating the bootloader. (<https://adafru.it/EWJ>)

A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times people have nearly given up due to a flakey USB cable! Enter Alert Text...

Hardware Checks

If, after updating your board's bootloader, you still think you may have a hardware problem, here's a great way to test out all of the functions. From buttons, to the light sensor, thumbstick (PyGamer only), accelerometer (PyGamer and PyBadge only, not the LC), and more, we've got a super nifty set of hardware test .UF2 files you can use.

Click on the link for your board below for more info and a link to the appropriate UF2 file.

PyBadge/PyBadge LC Hardware Check

<https://adafru.it/EWK>

PyGamer Hardware Check

<https://adafru.it/EWL>

Another way to do a hardware check is with the handy, dandy MakeCode Arcade Basic Hardware Test. This was created with MakeCode Arcade and you can use it to check that your d-pad buttons or thumb joystick can move the yellow face around the screen, and that the A and B buttons work to play a sound (just make sure you have a speaker plugged in to the PyGamer first).

You can [open this link \(https://adafru.it/EWP\)](https://adafru.it/EWP) to get to it, or download the UF2 file below and drag it onto your board's USB drive in bootloader mode.

arcade-Basic-Hardware-Test.UF2

<https://adafru.it/EWQ>

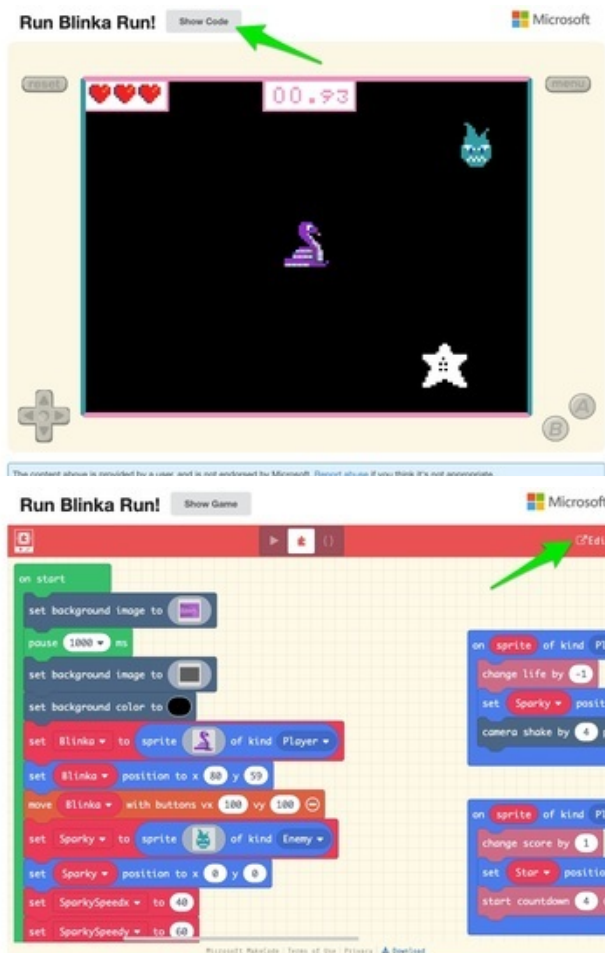


Load a MakeCode Game on PyGamer/ PyBadge

Let's load a game! For example, here's a link to **Run, Blinka, Run!** To open the game in the MakeCode Arcade editor, first, click the share link below. This will allow you to play the game in the browser right away.

Makecode Arcade Game: Run,
Blinka, Run!

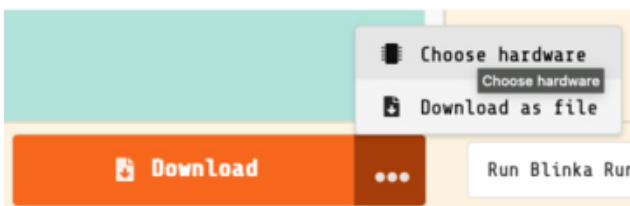
<https://adafru.it/Fqf>



Then, click on the Show Code button in the upper left corner. This shows the code for the game, and by clicking the Edit button in the upper right corner, it'll open into the editor where you can upload it to your PyGamer/PyBadge.

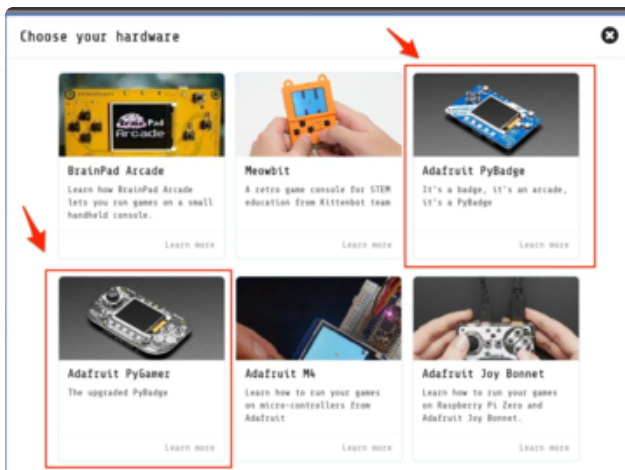
Once you have a game working on the MakeCode Arcade web editor, it's time to download it and flash it onto your board.

Please only use the Google Chrome browser with MakeCode! It has WebUSB support and seems to work best



Board Definition

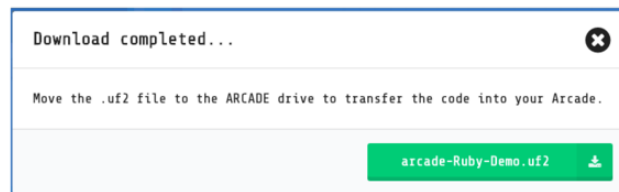
In order to load a game made in MakeCode Arcade onto the PyBadge, first choose the proper board definition inside of MakeCode. Click the ellipsis (...) next to **DOWNLOAD** and then the **Choose Hardware** item.



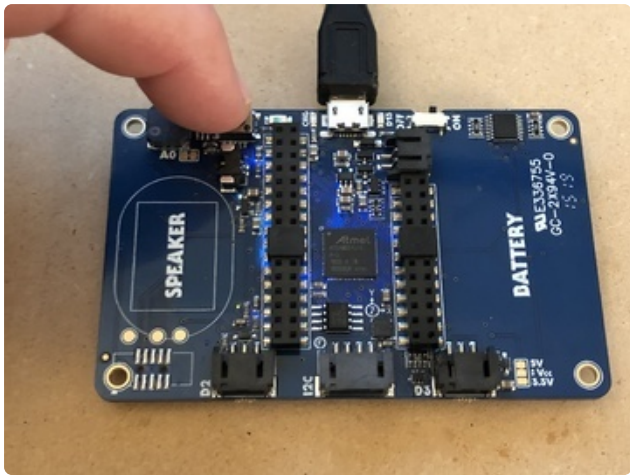
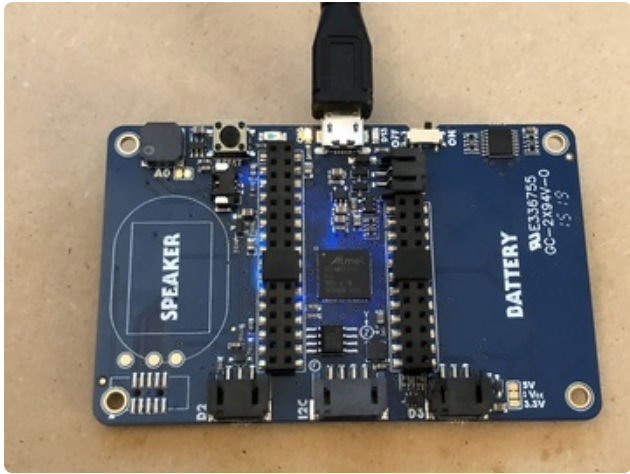
Change Board screen

Click on the image of your board, either the PyBadge/PyBadge LC or the PyGamer

This will cause the game .uf2 file for your particular board to be saved to your hard drive. You only need to do this the first time you use a new board. Thereafter you can simply click the **Download** button on the MakeCode Arcade editor page.



A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times people have nearly given up due to a flakey USB cable!

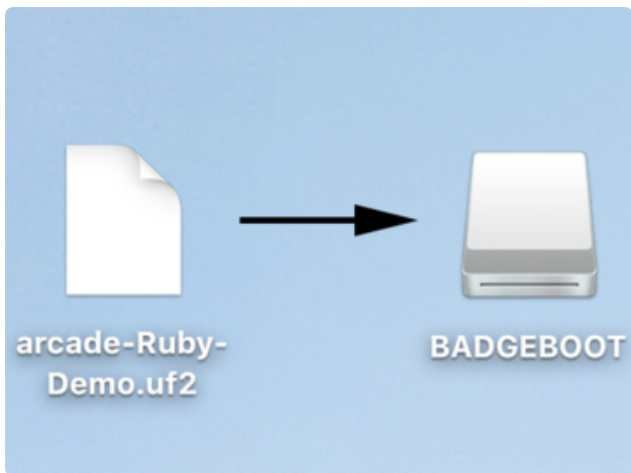


Bootloader Mode

Now, we'll put the board into bootloader mode so we can drag on the saved .uf2 file. On the back side of the board you'll see a reset button at the top. Make sure the board is plugged into your computer via USB with a USB micro B to A data cable. Also, be sure the board is turned on.

Then, press the reset button. This will initiate bootloader mode.

When the board is in bootloader mode you'll see a screen similar to this one show up.

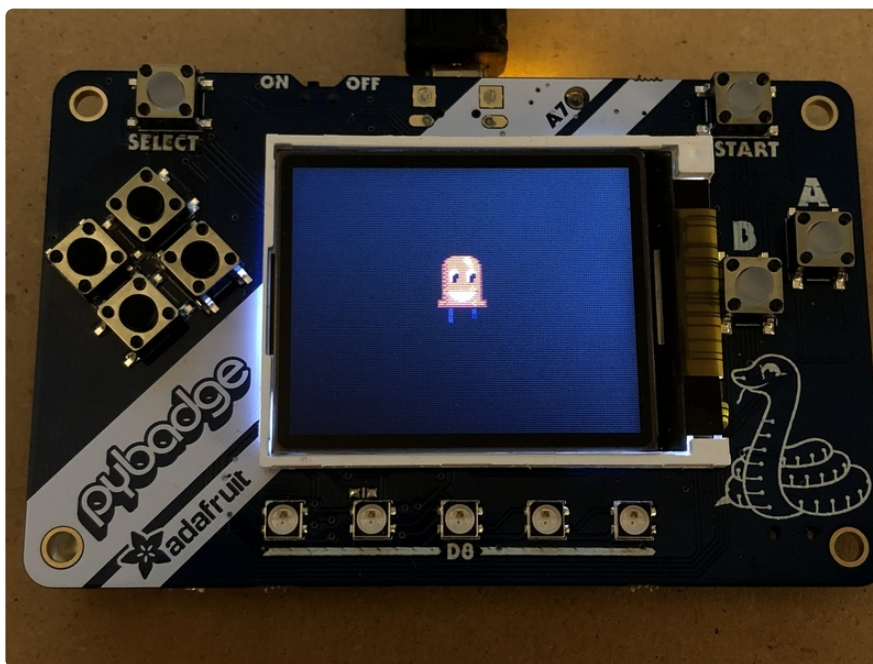


Drag and Drop

Now that the board is in bootloader mode, you should see a **BADGEBOOT** drive show up on your computer as a USB flash drive. Simply drag the arcade game .uf2 file onto the drive.

Play!

That's all there is to it! Once the file is copied over the board will restart and launch the game!



Keep an eye on [Adafruit.com](https://adafruit.com) for additional game related content.

Troubleshooting MakeCode Arcade

If you run into trouble with MakeCode Arcade, here are some resources for getting help:

- [Microsoft MakeCode Arcade Forum \(https://adafru.it/EXI\)](https://adafru.it/EXI)
- [Adafruit MakeCode Forum \(https://adafru.it/EXJ\)](https://adafru.it/EXJ)

- [Microsoft MakeCode Arcade Discord \(https://adafru.it/EXK\)](https://adafru.it/EXK) -- look for the #arcade channel
- [Adafruit MakeCode Discord \(\)](#) -- look for the #makecode channel

Only use the Google Chrome browser with MakeCode!