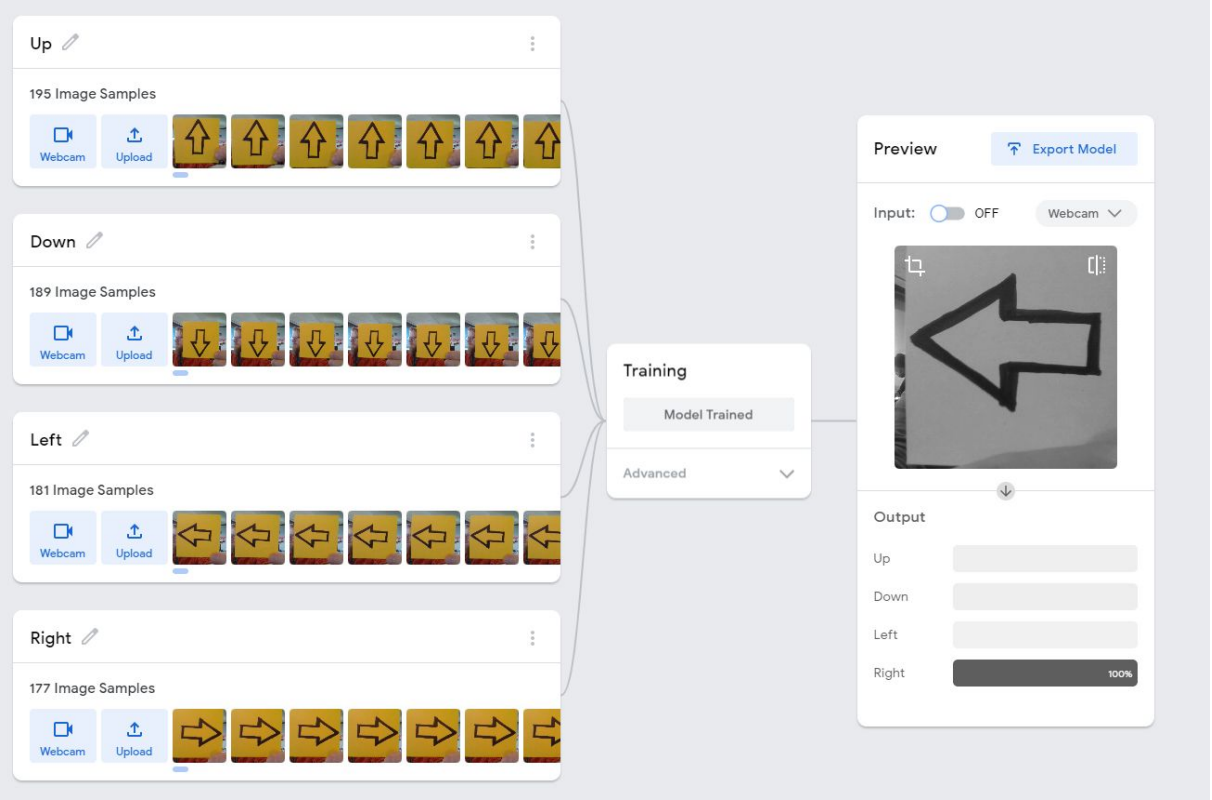
Let's build and control a program based on the classic 'Snake' game using Teachable Machine and p5.js

First we need to make a model on Teachable Machine.

Create four classes and fill each of them separately with pictures of your hand or arrows in different shapes.



Train this model and preview it to see if it works.
Leave this page open, we'll need it later.

Making the game in p5.js

Use the open in p5 option for some starter code.
The first thing we are going to do is create a playable game in p5.js.

Our player character

We are going to use something called a constructor function to create our playable character.

We are going to call our character player. Use the code below to create a player.

```
function player() {
  this.x = 0;
  this.y = 0;
  this.show = function(){
    fill(255);
    circle(this.x, this.y, 10, 10);
  }
}
```

In the draw function add in the **player.show();** code and and in the setup function add in the line player = new.player();

```
function draw() {
  background(220);
  player.show();
}
```

```
function setup() {
  createCanvas(400, 400);
  player = new player();
}
```

So the code should look like this by now.

```
function setup() {
  createCanvas(400, 400);
  player = new player();
}


function draw() {
  background(220);
  player.show();
}

function player() {
  this.x = 0;
  this.y = 0;

  this.show = function(){
    fill(255);
    circle(this.x, this.y, 10, 10);
  }
}
```

We're now going to add in the code to get this player moving on it's own. To do this we are going to use the x and y of the object and add a value to it.

```
this.xmove = 1;
this.ymove = 0;
```

These go inside the player function.

These set the variables that we are going to use to help the player move around the screen.

We now need to create a function that makes the player move. To do this we take the coordinates of the player and add the variable we've just made to it.

```
this.update = function() {
  this.x = this.x + this.xmove;
  this.y = this.y + this.ymove;
}
```

We also need to call this function in the draw()

```
function draw() {
  background(220);
  player.show();
  player.update();
}
```

Your player should now move across the screen.

Using the keyboard

Let's get the player moving using the keyboard to control it's direction.

We need to use the function function keyPressed() and we also need to use selection to tell it what to do when certain keys are pressed.

```
if (SOMETHING){
  DO_THIS;
    } else if (SOMETHING_ELSE) {
  DO_THIS;
  }
```

In this case we are going to check whether a key has been pressed we are going to do this by using the following code.

```
function keyPressed(){
  if (keyCode === RIGHT_ARROW) {
  player.direction(3,0);
  } else if (keyCode === DOWN_ARROW) {
  player.direction(0,3);
  }
  else if (keyCode === UP_ARROW) {
  player.direction(0,-3);
  }
  else if (keyCode === LEFT_ARROW) {
  player.direction(-3,0);
  }
}
```

The player wont move yet because we need to write another function called dirtection.

This will look like this.

```
this.direction = function(x,y) {
  player.xmove = x;
  player.ymove = y;
}
```

This function sets the direction of the player and how it's coordinates change. The player will now move when the keyboard arrows are pressed.

You'll notice that the player goes off the screen. We can stop this by using a built-in function called **constrain.**

```
this.update = function() {
  this.x = this.x + this.xmove;
  this.y = this.y + this.ymove;

  this.x = constrain(this.x,0,width)
  this.y = constrain(this.y,0,height)
}
```

Now the player doesn't go off the screen.

Making the collectable object

In this case we'll call the collectable object 'food' and the code below when called will place an object at a random location.

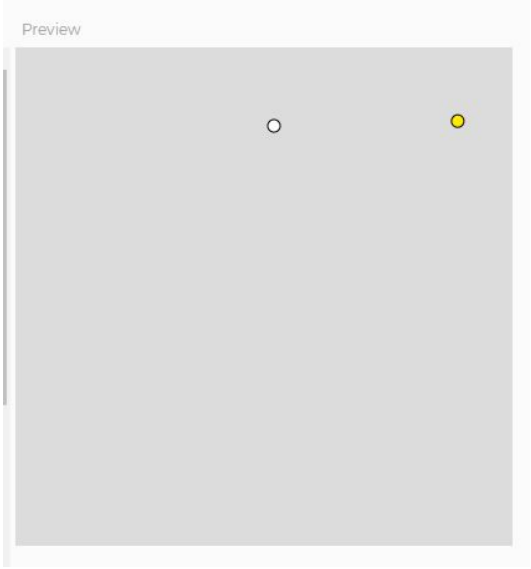
```
function food() {
  this.x = random(width);
  this.y = random(height);
  this.show = function(){
  fill(255,234,0)
  circle(this.x, this.y, 10, 10)
  }
}
```

Now you need to add it to the draw and the setup functions.

```
function draw() {
  background(220);
  player.show();
  player.update();
  food.show();
}
```

```
function setup() {
  createCanvas(500, 500);
  player = new player;
  food = new food;
}
```

The display should look like this



Adding in the ability to collect or eat the the collectable object. In this case the best way to do this is to text whether the two things are at the same coordinates.

Let's create a function that checks this.

```
this.eat = function(){
  var loc = dist(this.x,this.y,food.x,food.y);
  if (loc < 10){
    food.x = random(width);
    food.y = random(height);
    }
}
```

Here we're creating a variable called loc (standing for location) and using another built in function called dist. This compares the location of the first two points with the second two and returns a value. In our case if either one is less than 10 then we'll change the location of the food so it appears to change when it's collected.

Remember we need to add the function to the draw so it actually gets called.

```
player.eat();
```

Great! So we've now got the main part of the game working. The nest steps are:

- Adding a score
- Adding a timer

Let's add a timer
So we're going to add a timer and when it runs out we'll set the location of the food to off the screen and stop the player from moving.
Declare a variable at the state of your program called timer and set it to 20. This will be 20 seconds

```
var timer = 20
```

Include this code in your draw function() as what it does is subtract 1 from the timer every 60 frames (60 frames is one second in p5.js)

```
if (frameCount % 60 == 0 && timer > 0) {
  timer --;
}
```

The ‘timer --’ code is the same as ‘timer = timer - 1’

We now need to add a conditional to tell the computer what to do if the timer gets to 0. The code for this is below.

```
if (frameCount % 60 == 0 && timer > 0) {
  timer --;
}

if (timer == 0) {

  textAlign(CENTER, CENTER);
  textSize(40);
  text("Game Over",width/2, height/2);
  player.direction(0,0)
  food.x = -10;
  food.y = -10;
}
```

This code draws text to the screen, sets the food to a location out of the canvas range, and sets the direction of the player to 0,0 this means it doesn't move.

Let’s add a score

So the idea around the score is that when the food gets eaten the score goes up by one.

First thing we need to do is create the variable. This is created at the top of our file.

```
var score = 0;
```

Next we need to add the code in the section where the player eats the food.

We want to increase the score by 1 so the logic would be:
score = score + 1

In p5.js we can use score ++ and it does the same thing.

The player.eat() function now becomes.

```
this.eat = function(){
  var loc = dist(this.x,this.y,food.x,food.y);
  if (loc < 10){
    food.x = random(width);
    food.y = random(height);

    score ++
    }
}
```

Finally we need to display it at the end. So let's change the end text to show the score at the end of the game.

The code in the timer now changes to include the score in the end screen.

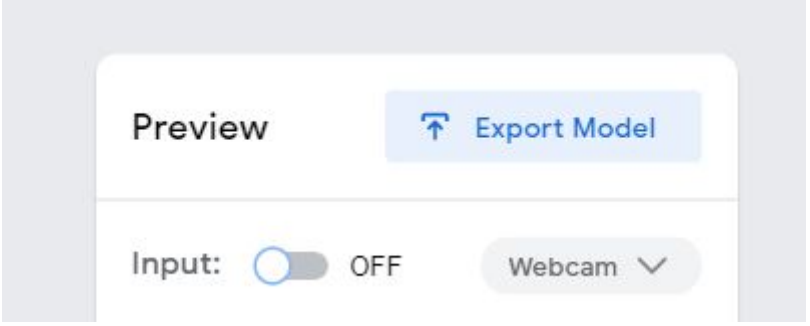
```
if (timer == 0) {

  textAlign(CENTER, CENTER);
  textSize(40);
  text("You score:" + score,width/2, height/2);
  player.direction(0,0)
  food.x = -10;
  food.y = -10;
}
```

Extra challenges

- Can you display the timer in the display window?
- Can you add another player using WASD?
- Make the player faster every time they collect a food
- Add in static objects that kill the player when they touch it.
- Add in moving objects that kill the player when they touch it.

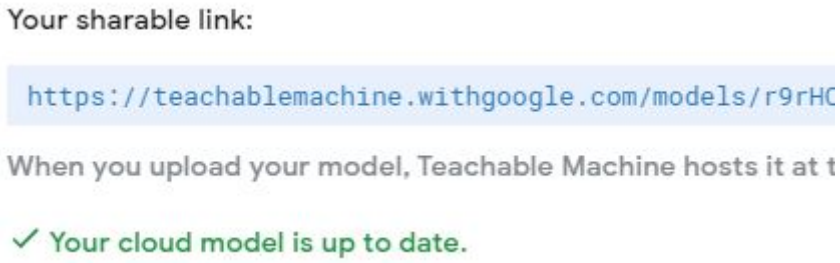
Using your model to control the game!

Go to your model on teachableMachine and copy the model url. To do this click on the 'Export Model' button.



Upload the model.

Wait until the model is uploaded. And copy the link of the model.



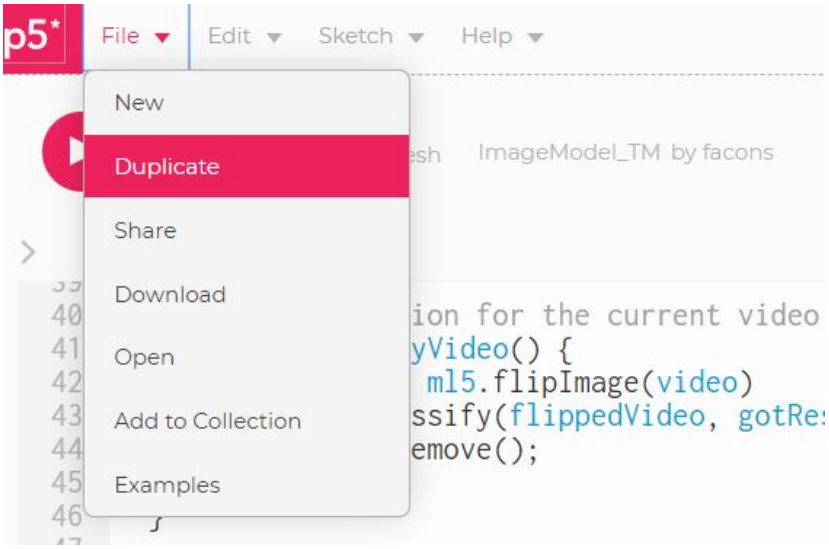
Add this code to your file by copying the code into a new p5. Only copy from the line let classifier, we don't need the rest

Javascript p5.js Contribute on

Open up the code snippet below directly in the p5.js Web Editor.

```
<div>Teachable Machine Image Model - p5.js and ml5.js</div>
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/addons/p5.dom.min.js"></script>
<script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js"></script>
<script type="text/javascript">
  // Classifier Variable
  let classifier;
  // Model URL
  let imageModelURL = 'https://teachablemachine.withgoogle.com/models/6QIfoLib0/';

  // Video
  let video;
  let flippedVideo;
  // To store the classification
  let label = "";

  // Load the model first
  function preload() {
    classifier = ml5.imageClassifier(imageModelURL + 'model.json');
  }

  function setup() {
    createCanvas(320, 260);
    // Create the video
    video = createCapture(VIDEO);
    video.size(320, 240);
```

Duplicate the program so it is saved in your account



Copy your whole snake game code underneath the new project.Find the first instance of draw() (the one that is not your game) and delete it.

Next find the setup() from the model file and copy the highlighted lines into your game setup()

```
function setup() {
  createCanvas(500, 500);
  player = new player;
  food = new food;
  video = createCapture(VIDEO);
  video.size(320, 240);
  video.hide();

  flippedVideo = ml5.flipImage(video);
    // Start classifying
  classifyVideo();
}
```

Delete the model setup()

Now we need to tell the program what to do with the A.I information. Add the code below into the classification function.

```
// console.log(results[0]);
label = results[0].label;
if (label == "right") {
player.direction(3,0);
} else if (label == "down") {
player.direction(0,3);
}
else if (label == "up") {
player.direction(0,-3);
}
else if (label == "left") {
player.direction(-3,0);
}
// Classifiy again!
classifyVideo();
```

Play and Enjoy!!!!!

Examples

Snake keyboard game
https://editor.p5js.org/ftc/sketches/i7BhrOIC

Snake game with Teachable machine.
https://editor.p5js.org/ftc/sketches/lxAmkSXmp

If you want to use the above with your own model just update line 3 with your model url.