Day 2 of KAPLAY Workshop - Building a Game

Starting Point

We'll pick up where we left off last time:

```
// this sets up the game and changes the background color to green
kaplay({
    background: "#5ba675"
})
// this loads the kat sprite so we can use it. we'll be using more sprites
today
loadSprite("kat", "/crew/kat.png")

let kat = add([
    sprite("kat"),
    pos(center()),
    area(),
    body(),
    "player"
])

setGravity(1000)

let platform = add([
    rect(width(), height()),
    pos(0, height() - 20),
    area(),
    body({ isStatic: true }),
    outline(5),
])
```

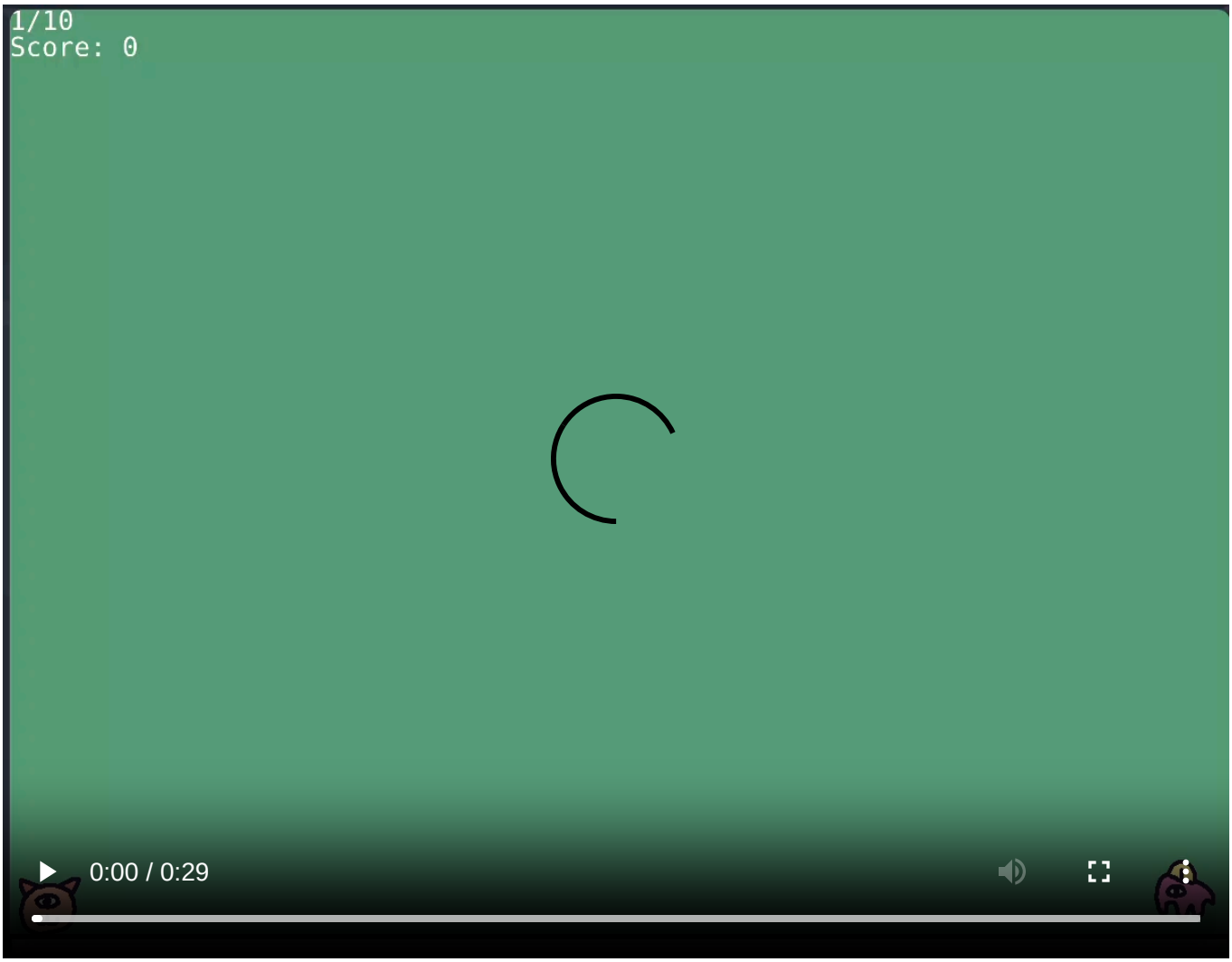
> Tell the class to go to this link: https://shorturl.at/TeoXr and resume from the Day 1 of KAPLAY Workshop - Learning the Basics of JavaScript > Controlling the Character

> Talk about each line and what happens while restarting the game to demonstrate the concepts.

What We're Building Today

> Once we complete the Day 1 material, show them what we'll be changing it to.

We'll build off of our base to create this game:



The game works like this:

1. Add some coins falling from the sky
2. The player can collect up to 10 coins
3. To earn score, the player needs to give the coins to the coin elephant
4. If the player has 10 coins and tries to collect an 11th, they lose

We might not get to *all* of this, but it does use all of the things we've learned so far (and then some).

This is *somewhat* different from what we talked about last week. I was thinking this would be a platformer, but there are a lot of concepts we would need to cover to make a fun one. With this, we'll be able to build a complete game, play-test it, and tweak it to make it more fun.

Full Game Code

This does NOT include any extra updates we made after play-testing.

```
kaplay({
    background: "#5ba675"
})

loadSprite("kat", "/crew/kat.png")
loadSprite("money_bag", "/crew/money_bag.png")
loadSprite("coin", "/crew/coin.png")

let kat = add([
    sprite("kat"),
    pos(10, center().y),
    area(),
    body(),
    "player"
])

setGravity(1000)

let platform = add([
    rect(width(), height()),
    pos(0, height() - 20),
    area(),
    body({ isStatic: true }),
    outline(5),
])

kat.onKeyPress("space", function () {
    if (kat.isGrounded()) {
        kat.jump()
    }
})

kat.onKeyDown("left", function () {
    kat.move(-250, 0)
})

kat.onKeyDown("right", function () {
    kat.move(250, 0)
})

let goal = add([
    sprite("money_bag", { flipX: true }),
    area(),
    pos(width() - 75, height() - 100)
])
```

```
goal.onCollide("player", function () {
    score = score + coinsCarried
    coinsCarried = 0
})

loop(1, function () {
    add([
        sprite("coin"),
        area(),
        body(),
        pos(randi(0, width() - 50), 0),
        "coin"
    ])
})

platform.onCollide("coin", function (coin) {
    coin.destroy()
})

let coinsCarried = 0
let maximumCoinsCarried = 10
let score = 0

onUpdate(function () {
    drawText({
        text: "Coins Carried: " + coinsCarried,
        size: 26
    })
    drawText({
        text: "Score: " + score,
        size: 26,
        pos: vec2(0, 26)
    })
})

kat.onCollide("coin", function (coin) {
    if (coinsCarried < maximumCoinsCarried) {
        coinsCarried = coinsCarried + 1
    } else {
        kat.destroy()
        addKaboom(kat.pos)
        burp()
    }
    coin.destroy()
})
```

Changing the Goal Sprite

Let's start with something easy: changing the sprite of the "goal". Previously, we only had access to the "kat" sprite, so we used that. Now, if you look at the bottom, you'll see a whole slew of different characters. We'll use some of those for our game today. Let's change the goal to the "money bag" sprite.

At the top of the file (after `kaplay()`), add the following code:

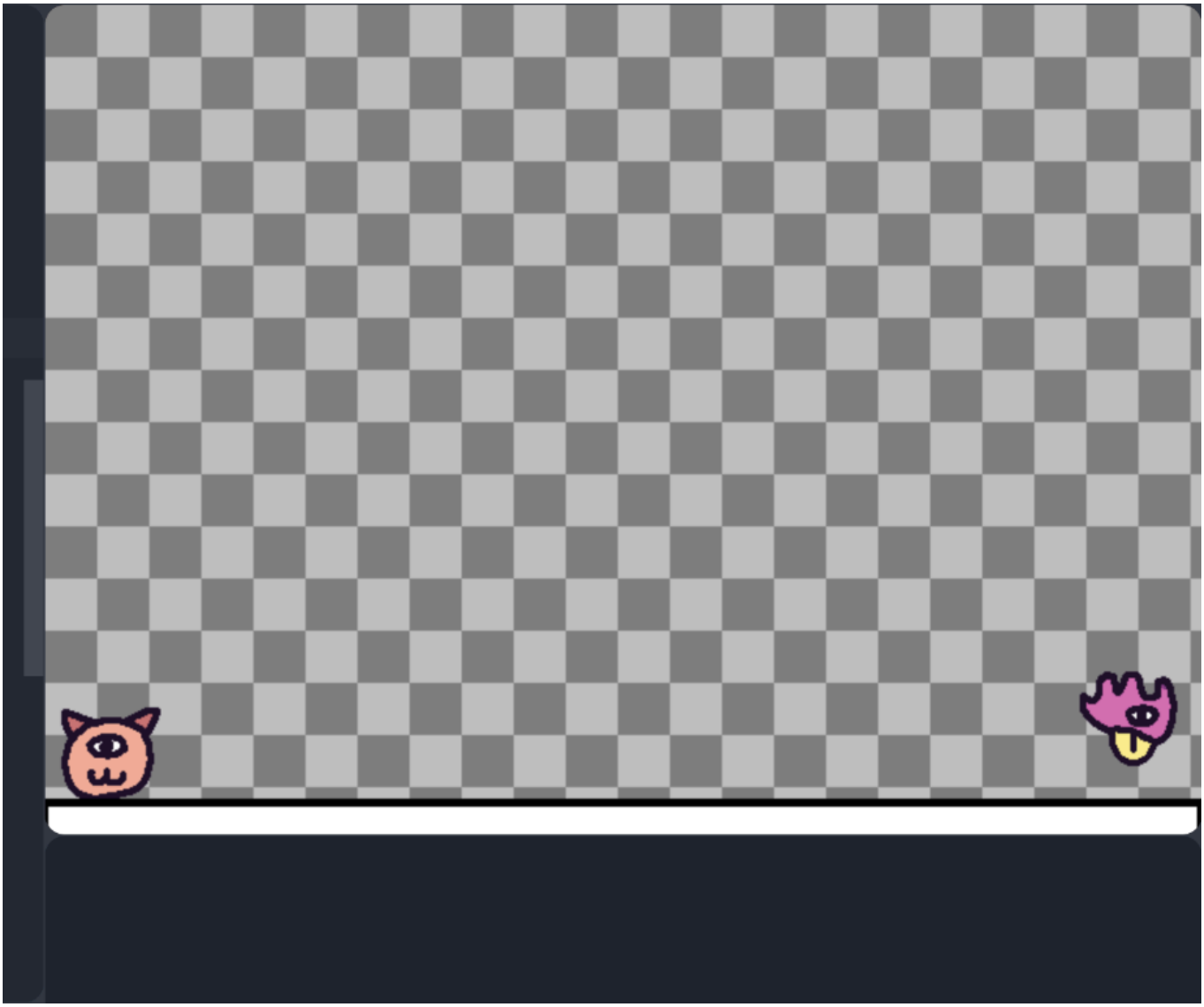
```
loadSprite("money_bag", "/crew/money_bag.png")
```

> **Why do we need this?**
> We need to tell the game which images to load before it can use them. The game wouldn't know what to show when we called `sprite("kat")` without it.

Like the other line we added for loading the "kat", we need another line to load the image for the goal (they named him "money_bag"). Then, let's change the sprite:

```
let goal = add([
    // changed "kat" to "money_bag"
    sprite("money_bag", { flipY: true }),
    area(),
    pos(width() - 75, height() - 100)
])
```

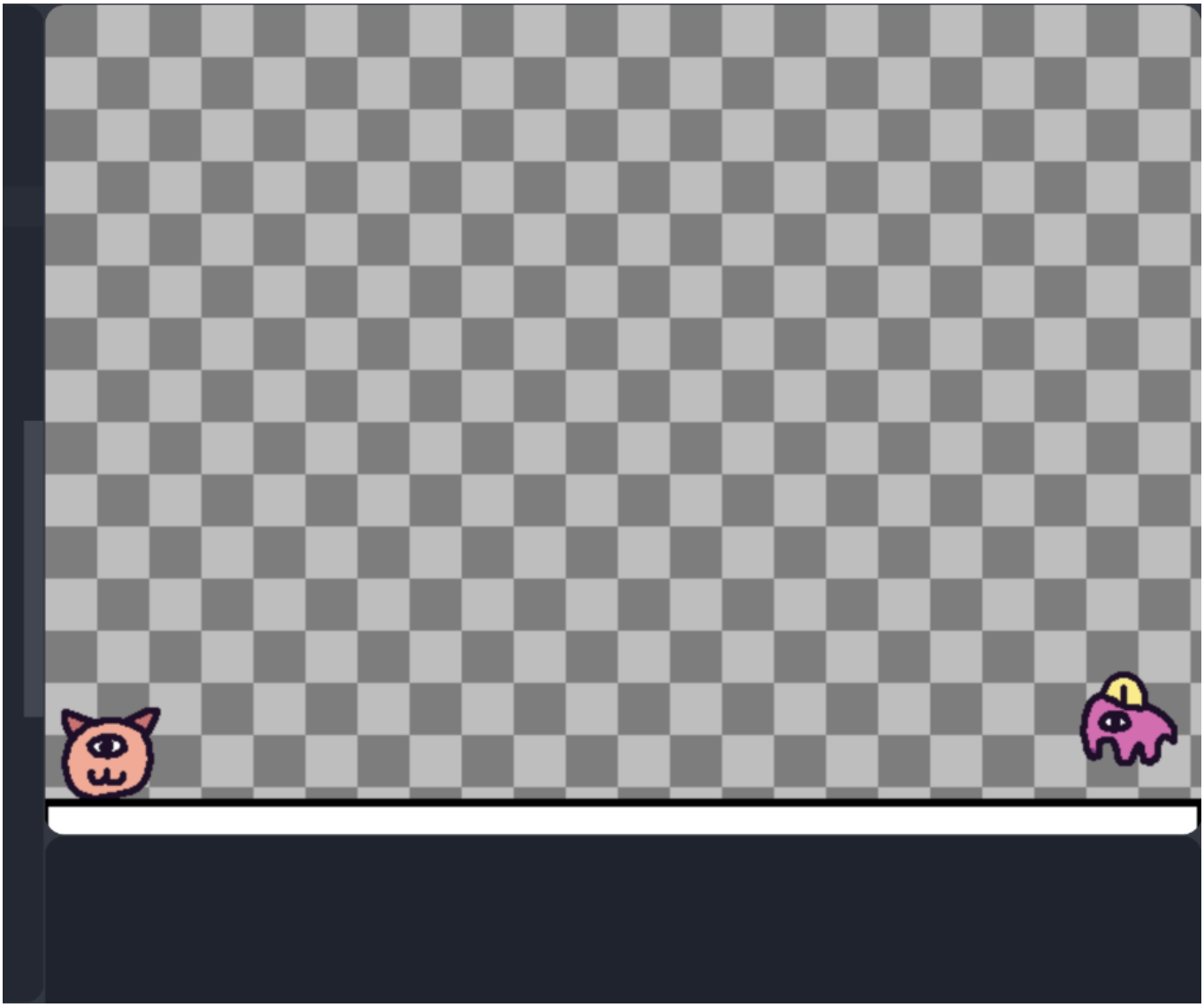
And that leaves us with:



Great, it's changed! But... he's not facing the right way. Let's get him right-side up and facing to the left:

```
let goal = add([
    // instead of "flipY", set "flipX"
    sprite("money_bag", { flipX: true }),
    area(),
    pos(width() - 75, height() - 100)
])
```

That leaves us with:



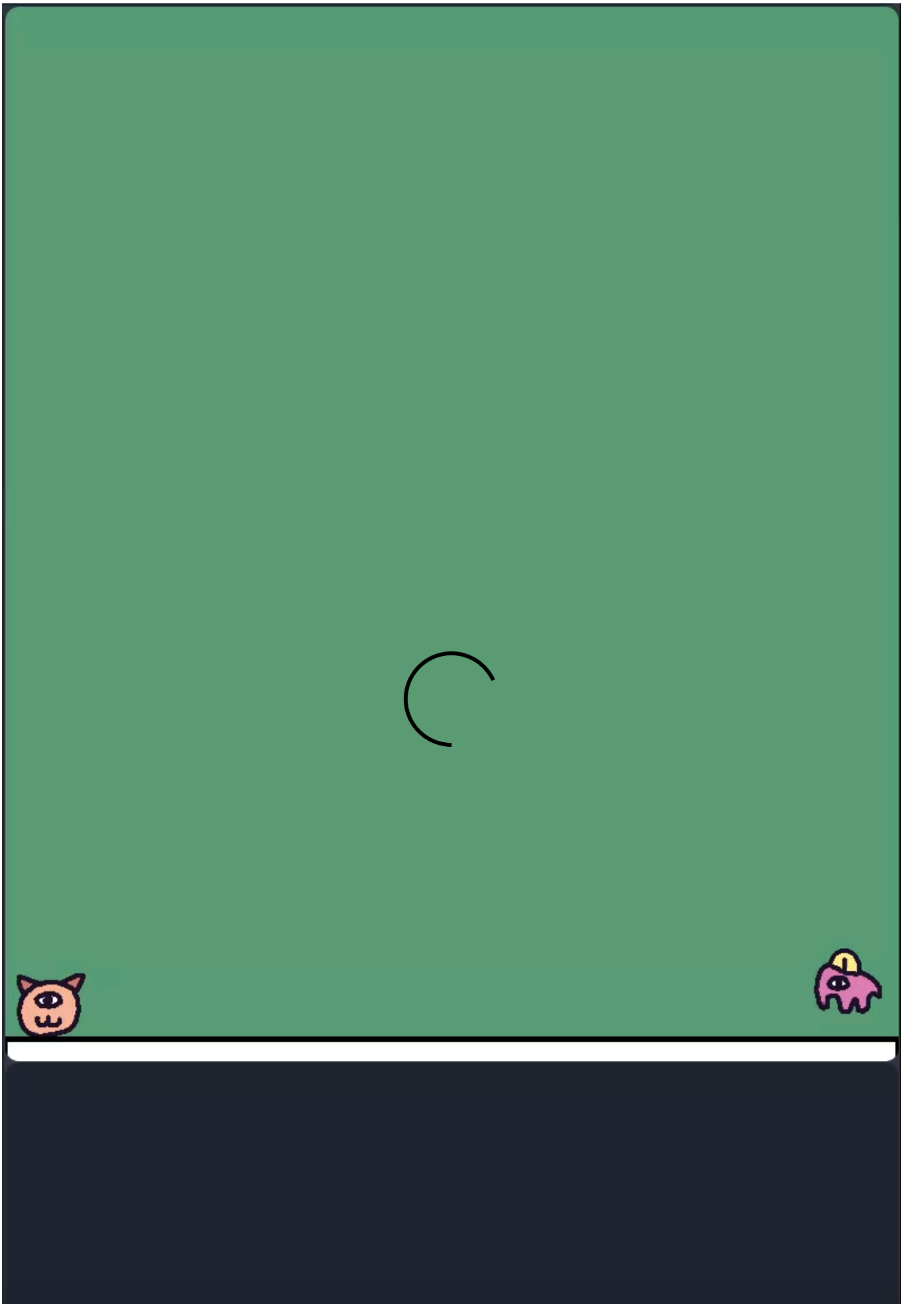
Perfect!

Coins Falling From the Sky

Next, we want to make coins fall from the sky. Let's start with adding coins every second. We can do this using the `loop` function:

```
loop(1, function () {
    console.log("once a second!")
})
```

Add this code and let's see what happens:



Once every second, `loop` calls the function we passed in. Since we told it to print to the console, it does just that. Let's have it create our coin:

```
loop(1, function () {
    add([
        sprite("coin"),
        area(),
        body(),
        pos(width() / 2, 0)
    ])
})
```

Here, we are:

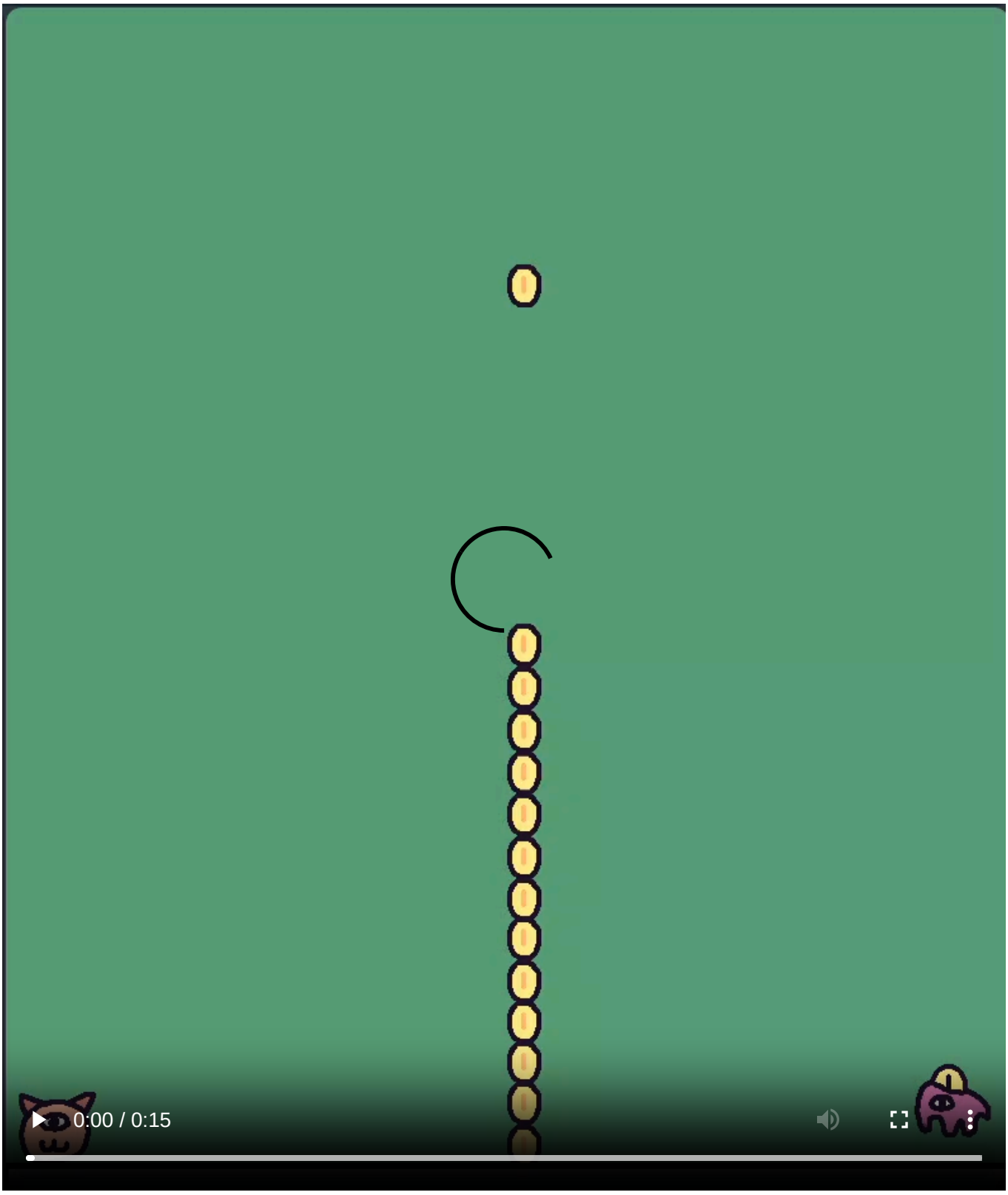
1. Adding another object to our game
2. Giving it a hitbox with `area()` (so it can collide with the player and the ground)
3. Applying gravity to it with `body()`
4. Telling it to appear at the top of the screen (the `0` in the vertical position) and in the center (the `width() / 2` in the horizontal position)

Let's see what this does:

Oh, right, we need to load the coin sprite first. Let's add it really quickly:

```
loadSprite("kat", "/crew/kat.png")
loadSprite("money_bag", "/crew/money_bag.png")
// new line here
loadSprite("coin", "/crew/coin.png")
```

Let's run the game again and see if this works:



Great! We have coins falling from the sky, but they keep stacking on top of each other. They also only fall from one place, which isn't that fun. Let's tackle the first one next.

Deleting the Coins

We need to remove the coins from the screen when the following things happens:

1. The coins touch the ground, or
2. The coins touch the player

Deleting on Collide with the Ground

Remember the code for colliding with the goal? It looks like this:

```
goal.onCollide("player", function () {
    // do stuff
})
```

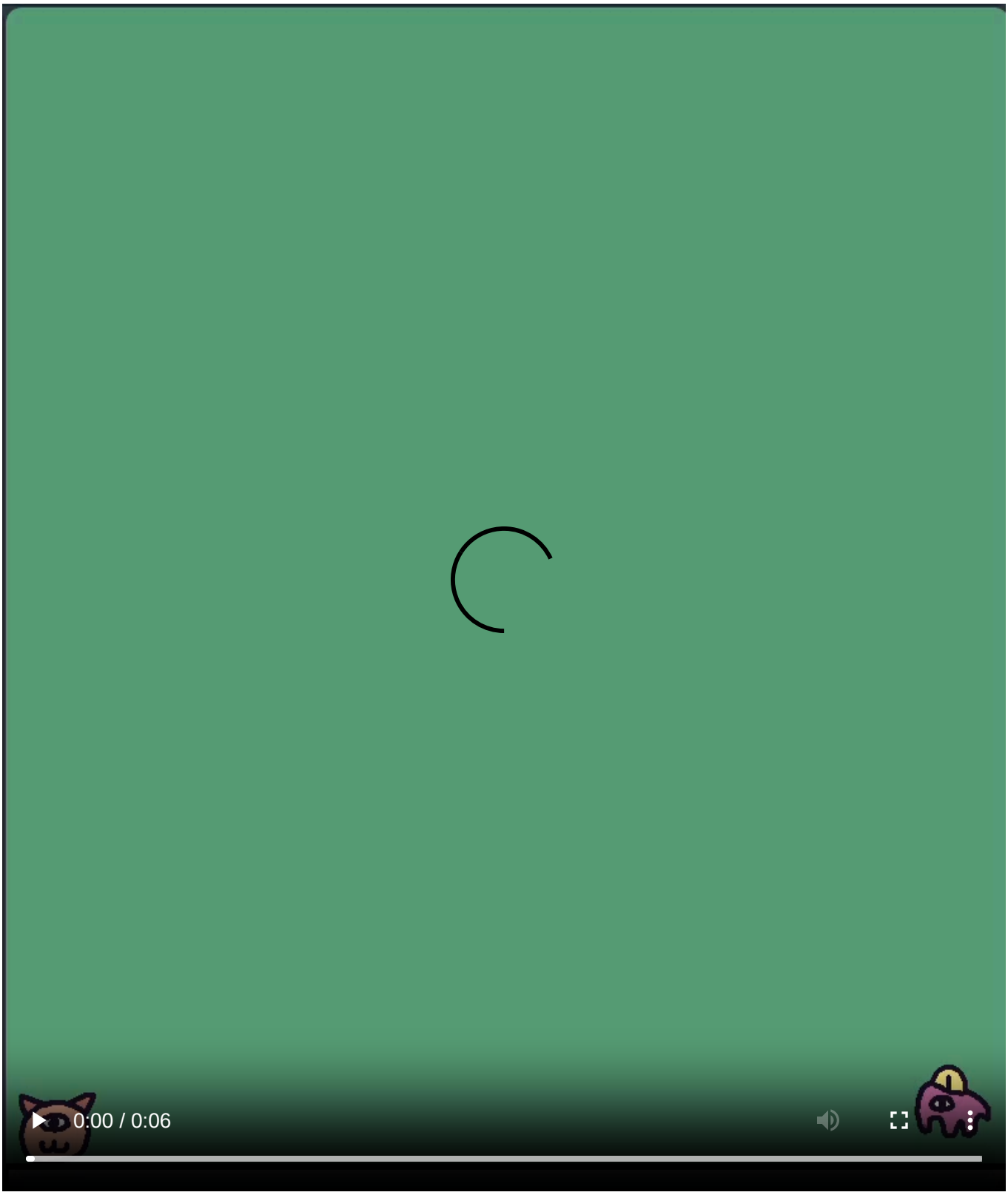
We're going to do the same thing with the `platform` and it will look very similar. Let's start by adding the collision handler:

```
platform.onCollide("coin", function(coin) {
    coin.destroy()
})
```

Let's break this down:

- `platform.onCollide` will run the function we give it when it collides with an object with the `coin` attribute
- Unlike before, our function takes a parameter: the coin that the platform collided with
- In that function, we "destroy" or remove the object by calling `coin.destroy()`

Let's run our game to see what happens:

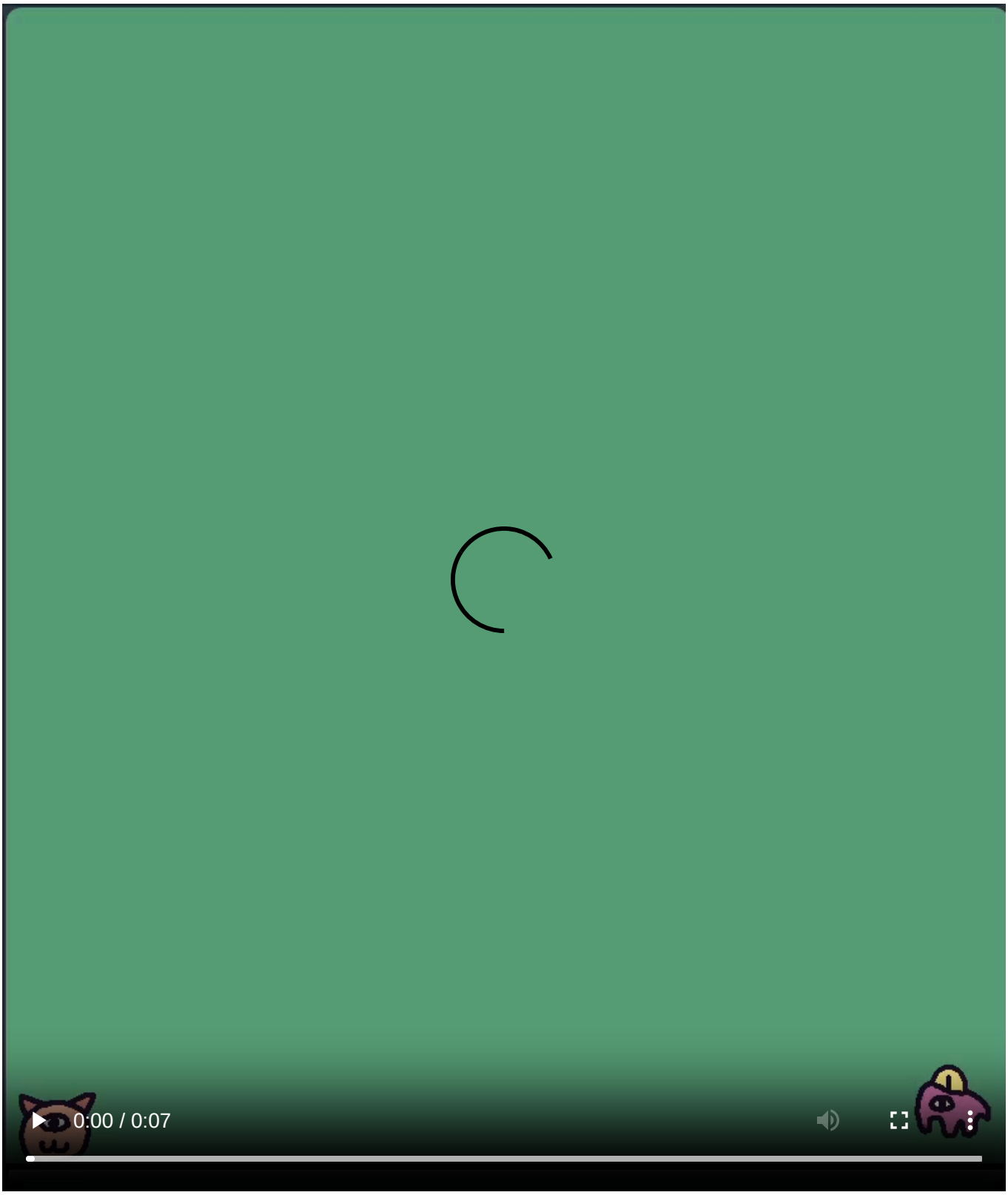


Oh, right. We didn't give the coin the `"coin"` attribute. This is exactly what we did with the player earlier. Let's add that to the coin:

```
loop(1, function () {
    add([
        sprite("coin"),
```

```
        area(),
        body(),
        pos(width() / 2, 0),
        // new code below
        "coin"
    ])
})
```

And now it works!



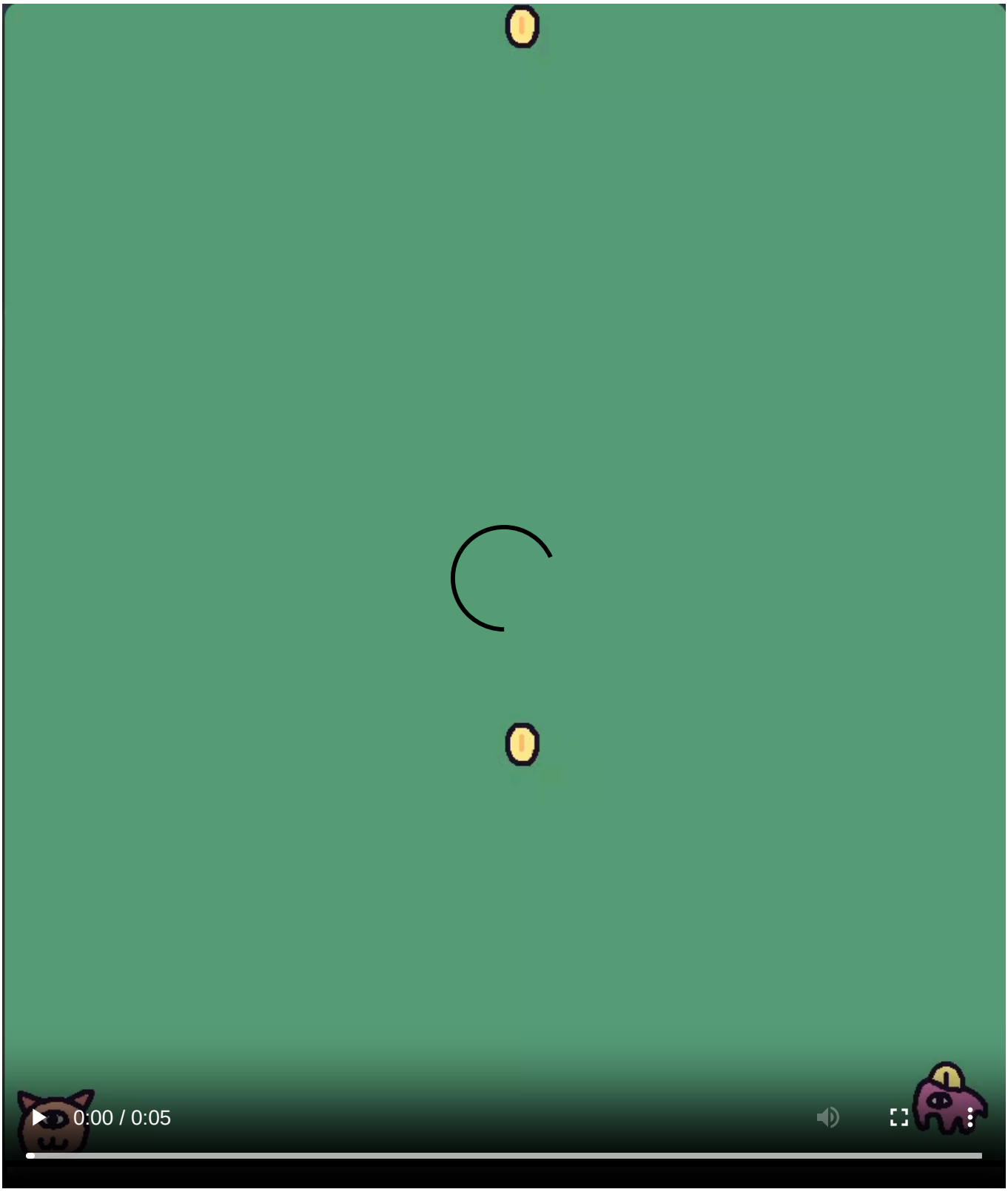
Deleting on Collide with the Player

Let's do the same thing with the player:

```
// at the end of the code so far
kat.onCollide("coin", function(coin) {
```

```
    coin.destroy()
})
```

And let's see how it works:



Perfect!

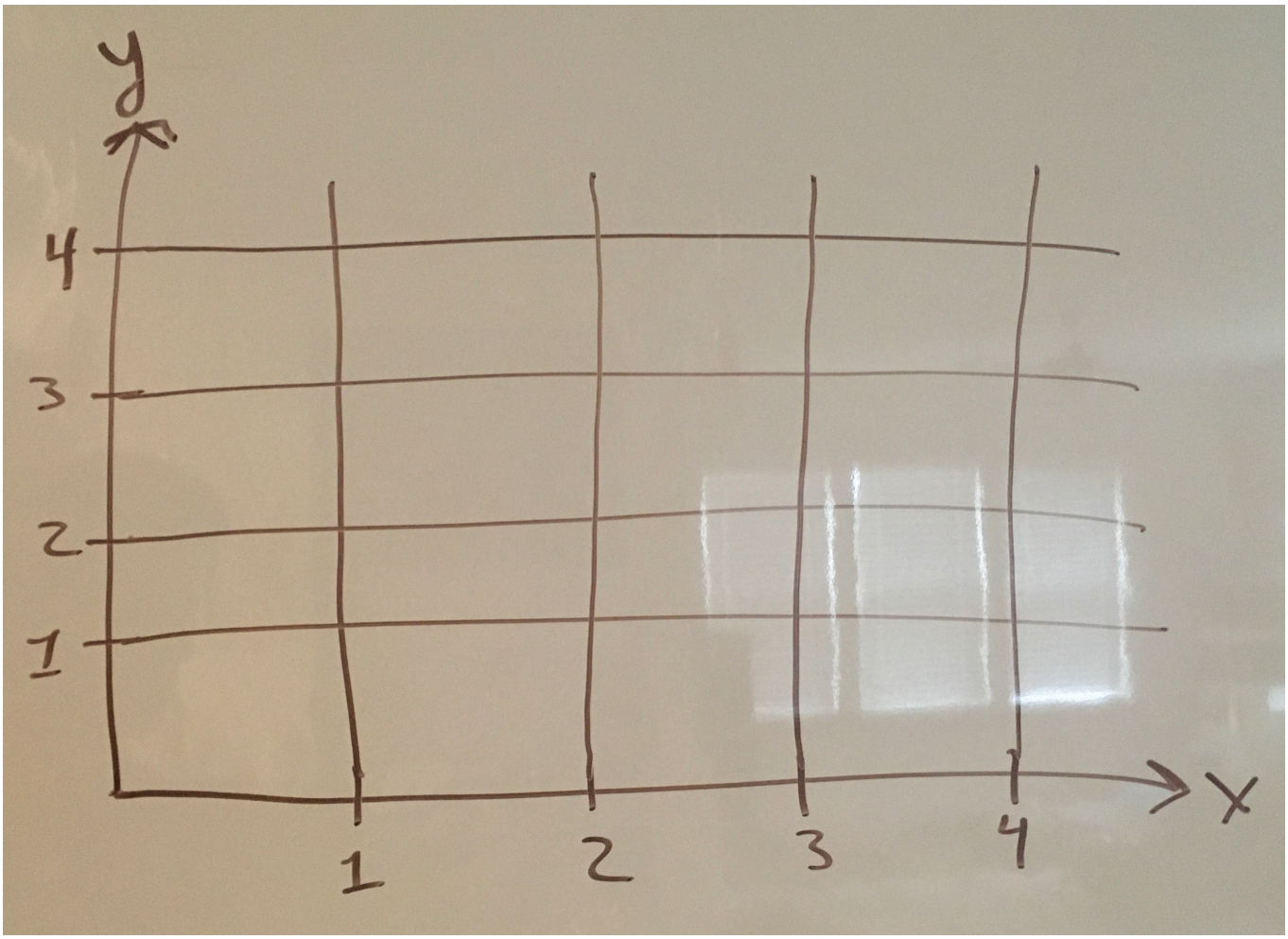
Making it Random

The coins only fall from a single spot on the screen. This isn't fun at all! Let's tweak it so the coins fall at random parts of the screen.

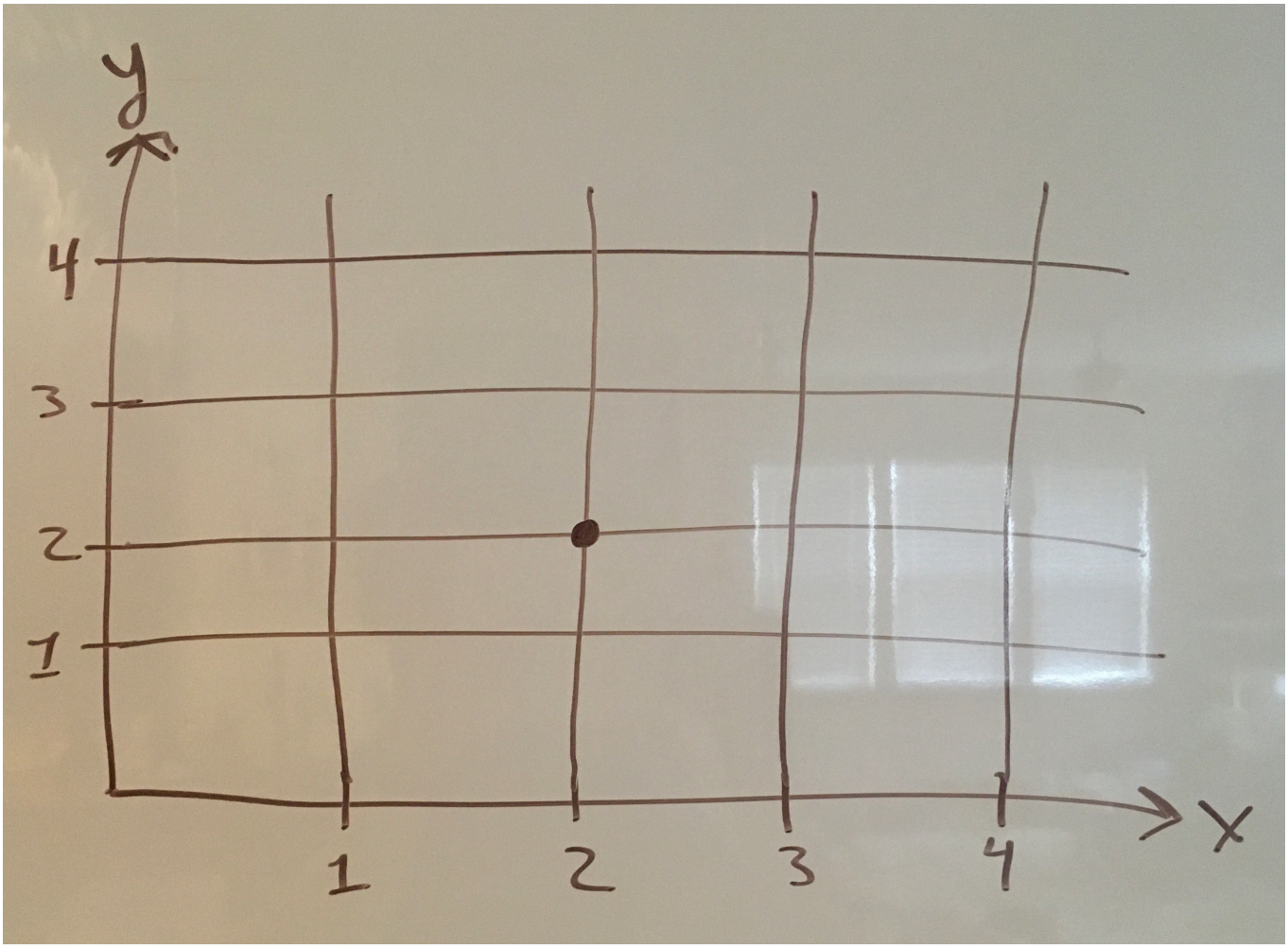
Aside: Coordinates

"Normal" Coordinate Grid

Before we talk about that, we need to talk about coordinates. Who here has seen this in their math classes before:



This is a coordinate grid, or a graph. It has an x-axis (the line at the bottom going horizontally) and a y-axis (the line of the left going vertically). If I put a point on this graph like this:



> Ask them: what would the position be?

We write positions on a coordinate grid like this in the format `(x, y)`. `x` is the number on the x-axis it's on, and `y` is the number of the y-axis. So, in this example, the coordinates for this point are `(2, 2)`.

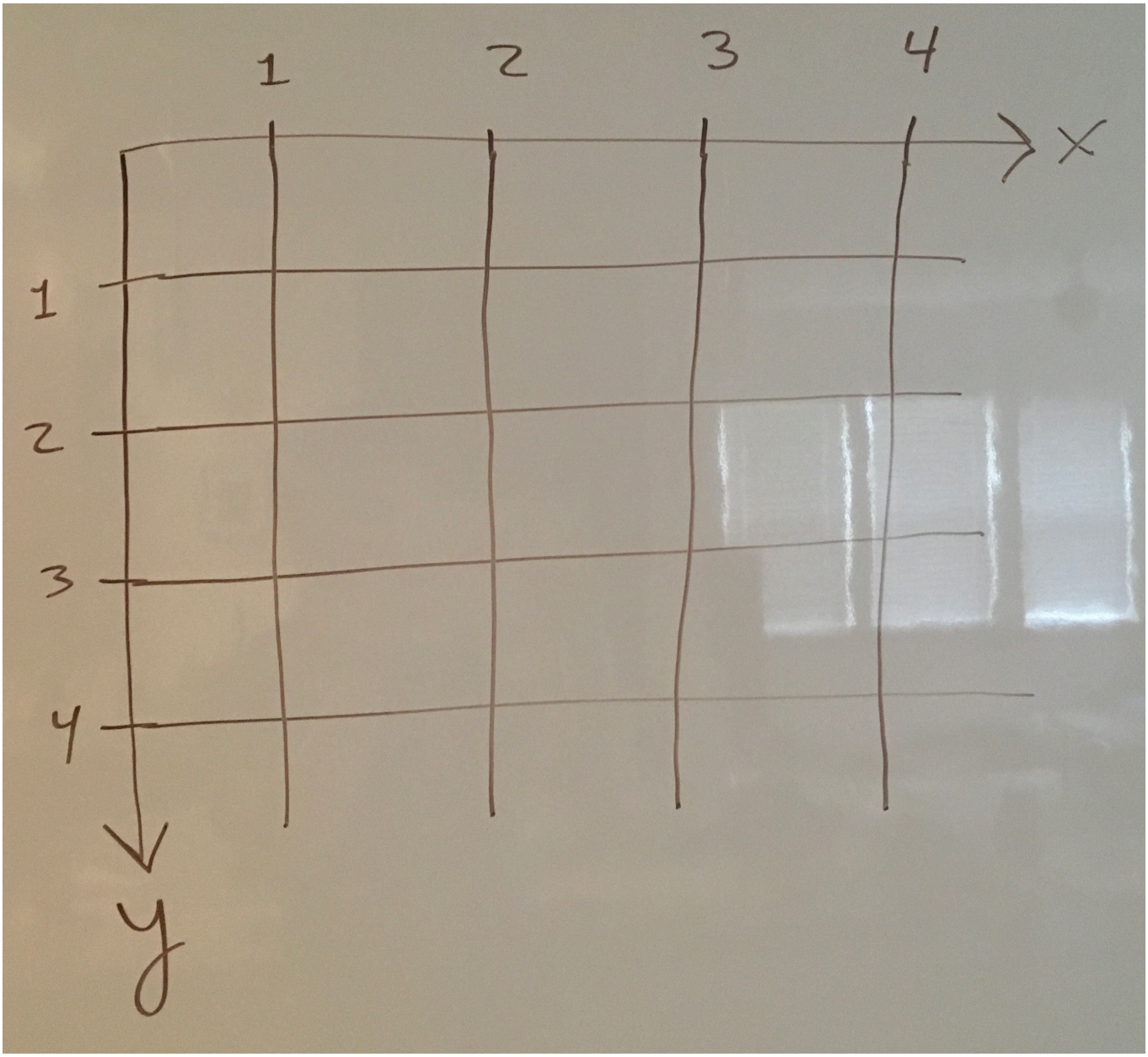
> What would we need to change for the point to move up? What about moving down? Left and right?

- Up: add to `y`
- Down: subtract from `y`
- Left: subtract from `x`
- Right: add to `x`

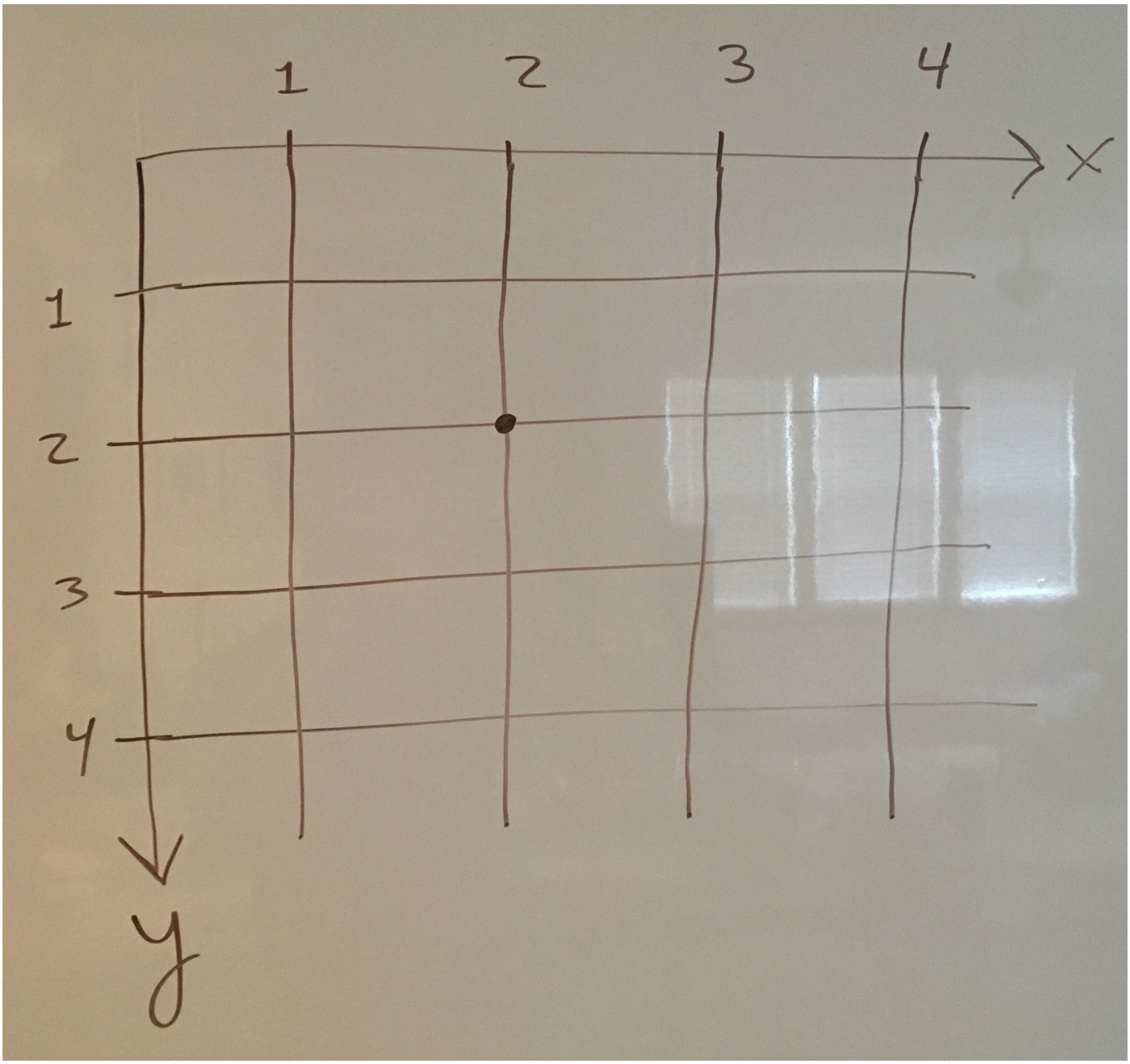
> Show a couple demonstrations of this on the whiteboard.

How the Game's Grid Works

The game's coordinate grid is *similar*, but there's a difference: it's flipped vertically. You can visualize it like this:



Notice how the y-axis is pointing *down* this time. This means that going down *increases* the y-value and going up *decreases* it. Let's try the same exercise as before:



> Show the same `(x, y)` notation and put a point on `(2, 2)`. Ask again: how would I move this up, down, left, and right?

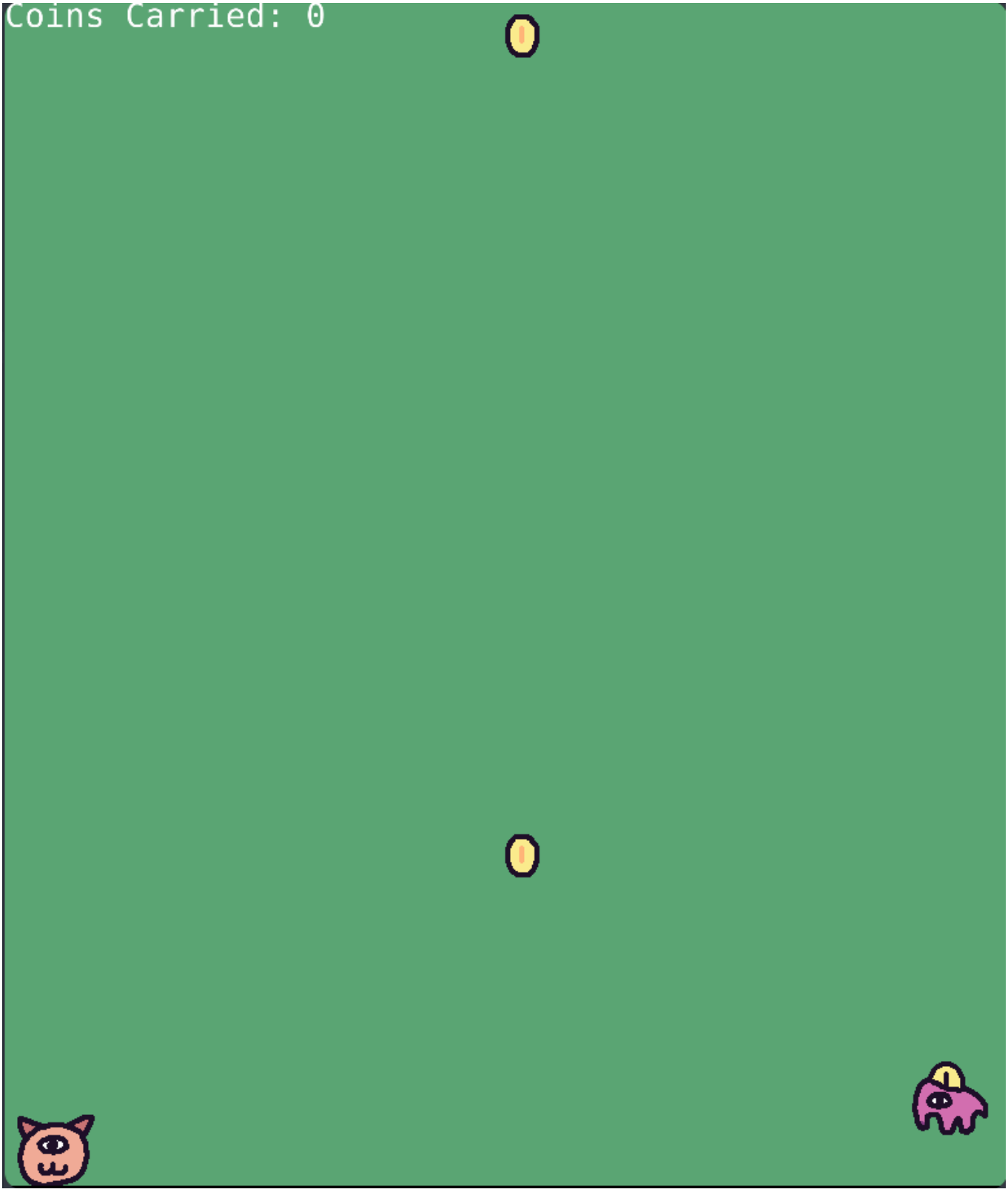
- Up: *subtract* from `y`
- Down: *add to* `y`
- Left: subtract from `x`
- Right: add to `x`

This is how the coordinates in the game work. If we look back at code we wrote, we can see this. Look at the `pos` function for the `platform`:

```
let platform = add([
    rect(width(), height()),
```

```
    pos(0, height() - 20),
    area(),
    body({ isStatic: true }),
    outline(5),
])
```

It's position is `(0, height() - 20)`. Position is determined by the upper-left hand point of the object. In this case, we want it to *start* at `0` on the x-axis (all the way to the left). For the y-axis, we want it to appear at the bottom, but still show it a little bit. If the height of my screen is `300` for example, we want it to show up at `280`. If we showed it at `height()` without subtracting, it wouldn't be visible:



The player is still standing, but you don't *see* it. That's because it's *off-screen*.

Back to Making it Random

With all that said, let's look again at the code for our coin:

```
loop(1, function () {
    add([
        sprite("coin"),
        area(),
        body(),
        pos(width() / 2, 0),
        "coin"
    ])
})
```

> Ask them: why does it appear at the top of the screen? Why does it appear in the middle?
>
> To make it random, what should we change about it?

To make it random, we'll use a function KAPLAY provides: `randi`. It works like this:

```
randi(1, 100)
```

> Aside: `randi` stands for "random integer." This means it will only give you whole numbers and never a number with a decimal point. "Integer" is another word for "whole number."

This will give you a random number between 1 and 99. The upper bound is not inclusive, meaning the number will never be 100. If you want numbers between (and including) 1 and 100, you would need to call:

```
randi(1, 101)
```

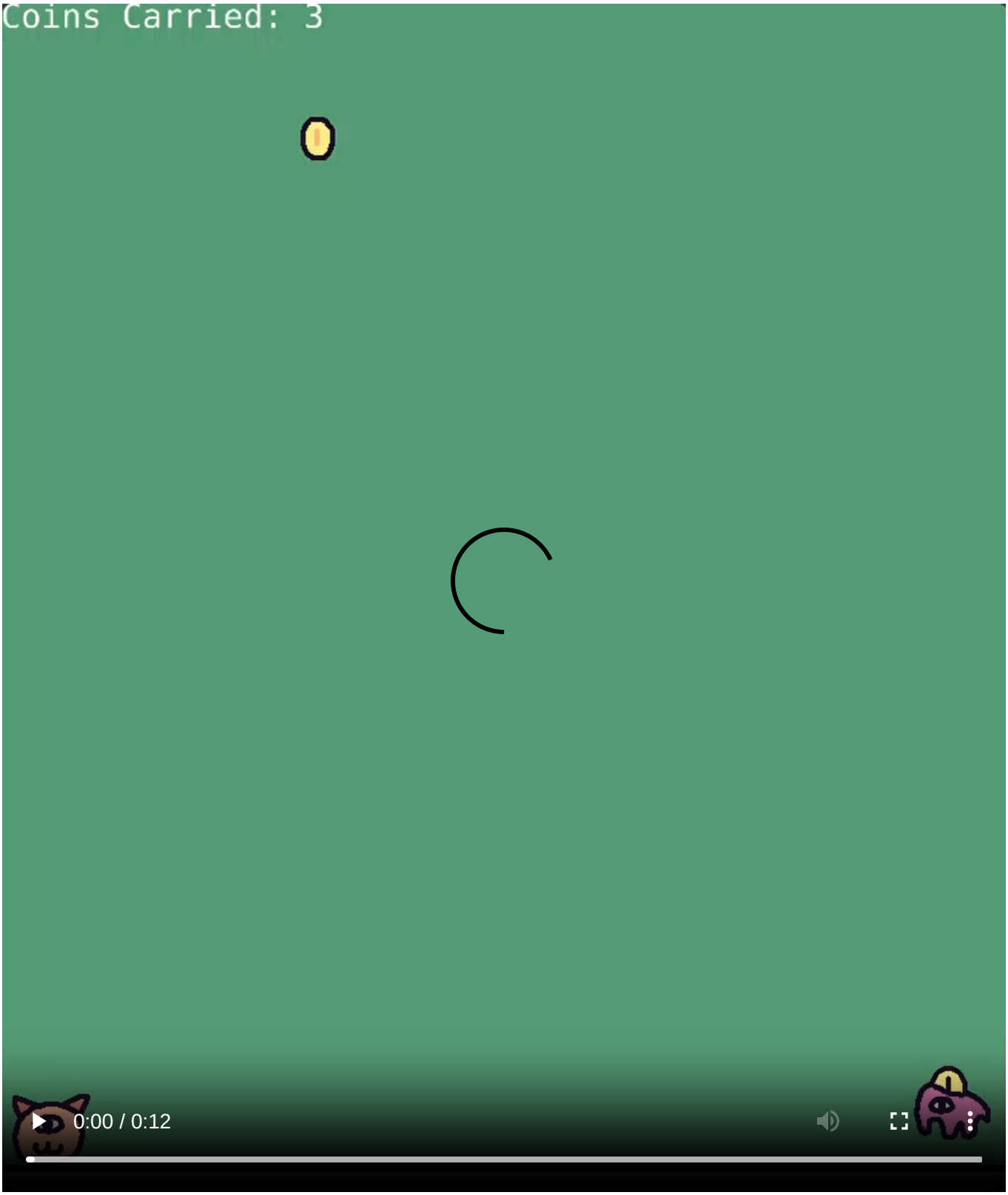
We know we want to keep the y-coordinate at 0, so we'll only change the x-coordinate.

> Ask them what they think the upper and lower bounds should be before filling them out.

Here's the code for that:

```
loop(1, function () {
    add([
        sprite("coin"),
        area(),
        body(),
        pos(randi(0, width()), 0),
        "coin"
    ])
})
```

Let's see this in action:



They're showing up at random spots every second! This works, but if you let it play long enough, it might show coins off screen on the right-hand side. Let's change the bounds of our `randi` function.

> Again, ask them: how should we change the bounds?
> **Answer:** subtract from the upper bound (50 is a good number to try)

```
loop(1, function () {
    add([
        sprite("coin"),
        area(),
        body(),
        pos(randi(0, width() - 50), 0),
        "coin"
    ])
})
```

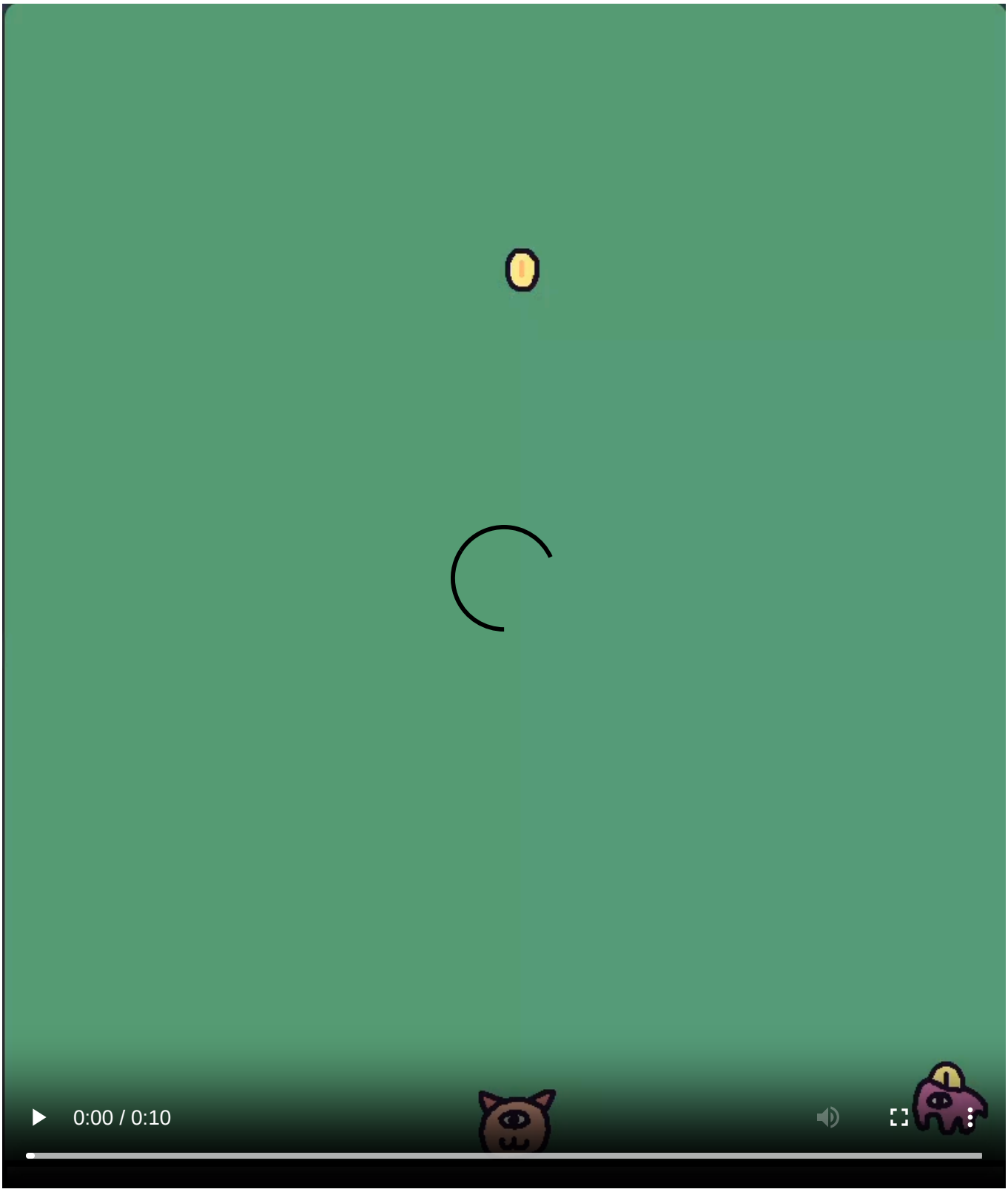
Adding Scoring

Add Coins Collected

We need to keep track of how many coins the player has. Let's create a variable to hold that:

```
// new code
let coinsCarried = 0

kat.onCollide("coin", function(coin) {
    // new code
    coinsCarried = coinsCarried + 1
    coin.destroy()
})
```

Let's see this in action:



Uhhh, I *guess* it's working? We can't really see how many coins we have. They are being removed, but is it increasing our score?

Showing Coins Collected on the Screen

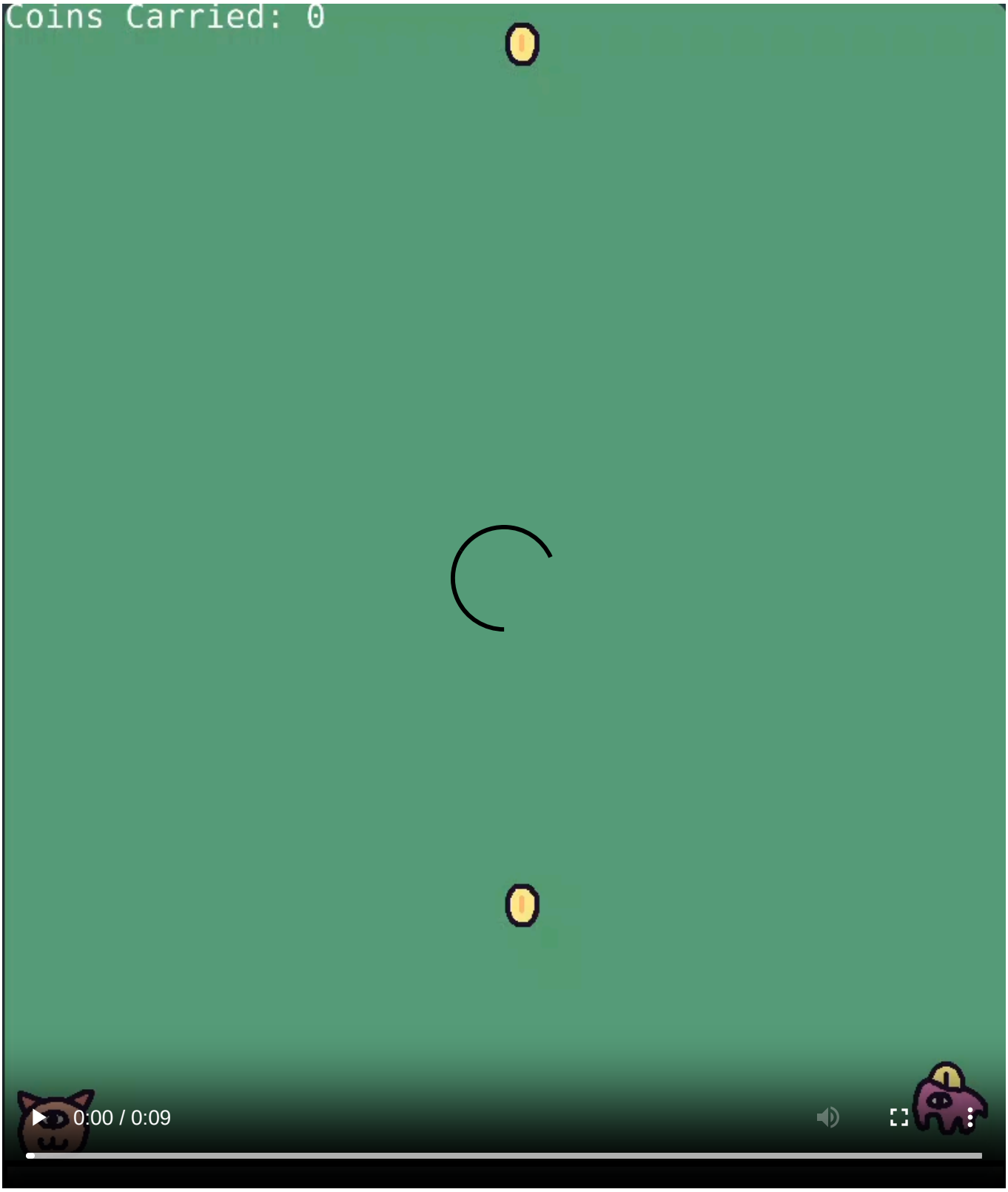
For our sake (and for the player), let's add some text on the screen. Right below your variable, we're going to add some code:

```
let coinsCarried = 0

// new code here
onUpdate(function () {
    drawText({
        text: "Coins Carried: " + coinsCarried,
        size: 26
    })
})

kat.onCollide("coin", function(coin) {
    coinsCarried = coinsCarried + 1
    coin.destroy()
})
```

Let's break this down:

- `onUpdate` is another event handler that runs on every *frame* of the game
  - When you play a game, what's the unit of measurement used to determine how fast your game is (or how smooth the graphics look)? **Frames per second**.
  - If your game is running at 60 frames per second, it's drawing an image to the screen 60 times in a single second
  - So, `onUpdate` gets called 60 times every second!
- `drawText` will show some text on the screen
  - It takes an object (indicated by the curly braces `{}`)
  - It takes a `text` field that indicates what text to show
  - It takes a `size` field that tells it how big to make the text
  - `"Coins Carried: " + coinsCarried` is combining the string with the number to create "Coins Carried: 0"

And now, we'll see if it works:



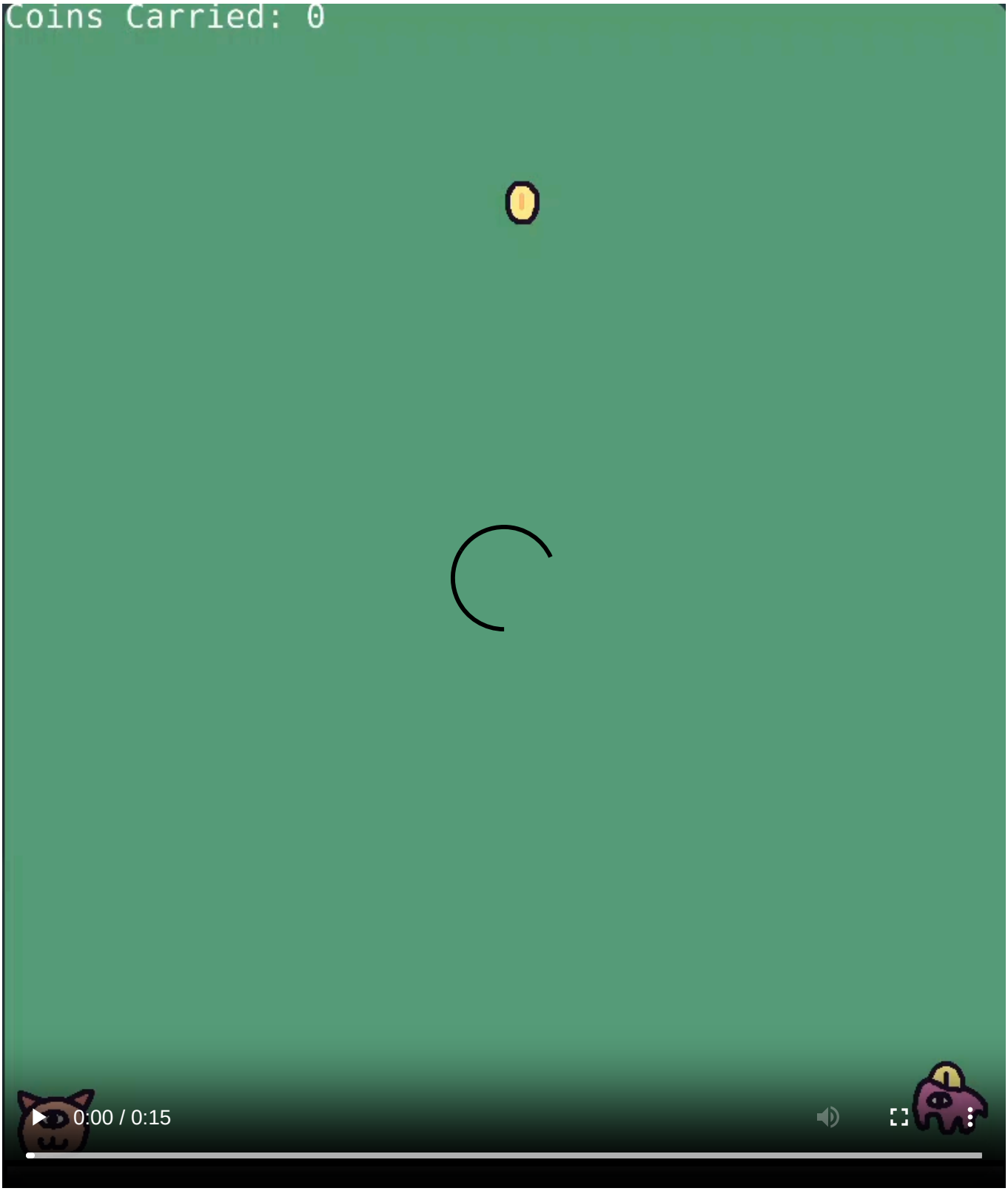
It's working!

Add Coin Carrying Limit

Remember back to the demo I showed earlier. The player could only collect 10 coins. Let's add a limit to that:

```
let coinsCarried = 0
// new code
let maximumCoinsCarried = 10

// the onUpdate call, omitted for this

kat.onCollide("coin", function (coin) {
    // new code
    if (coinsCarried < maximumCoinsCarried) {
        coinsCarried = coinsCarried + 1
    }
    coin.destroy()
})
```

We added a new variable to hold the maximum number of coins, and then we modified the on collide function to only add to our coins if we have less than 10. Let's see if this works:



Great, it works perfectly! Our `if` statement prevents the number of coins from increasing if we already have 10 coins. We still destroy the coin because we don't want it to take up the entire screen.

Add Collection Mechanic

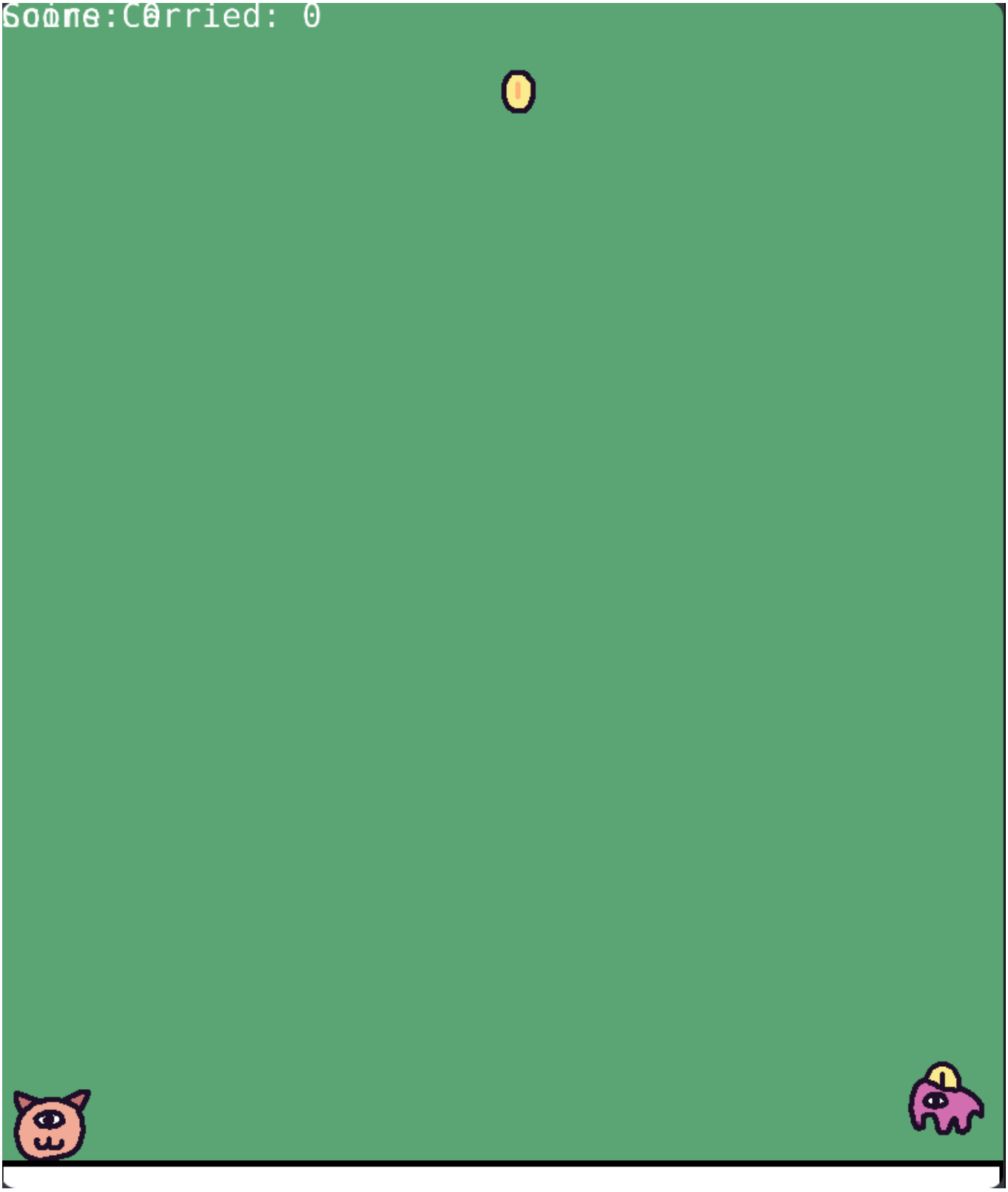
We can carry the coins, now we have to "drop" them off to add to our score. First, let's create a new variable for our score:

```
let coinsCarried = 0
let maximumCoinsCarried = 10
// new code below
let score = 0
```

Then, let's show it on the screen (so we can know it's working and the player knows too):

```
onUpdate(function () {
    drawText({
        text: "Coins Carried: " + coinsCarried,
        size: 26
    })
    // new code below
    drawText({
        text: "Score: " + score,
        size: 26
    })
})
```

And we'll see it on the screen:

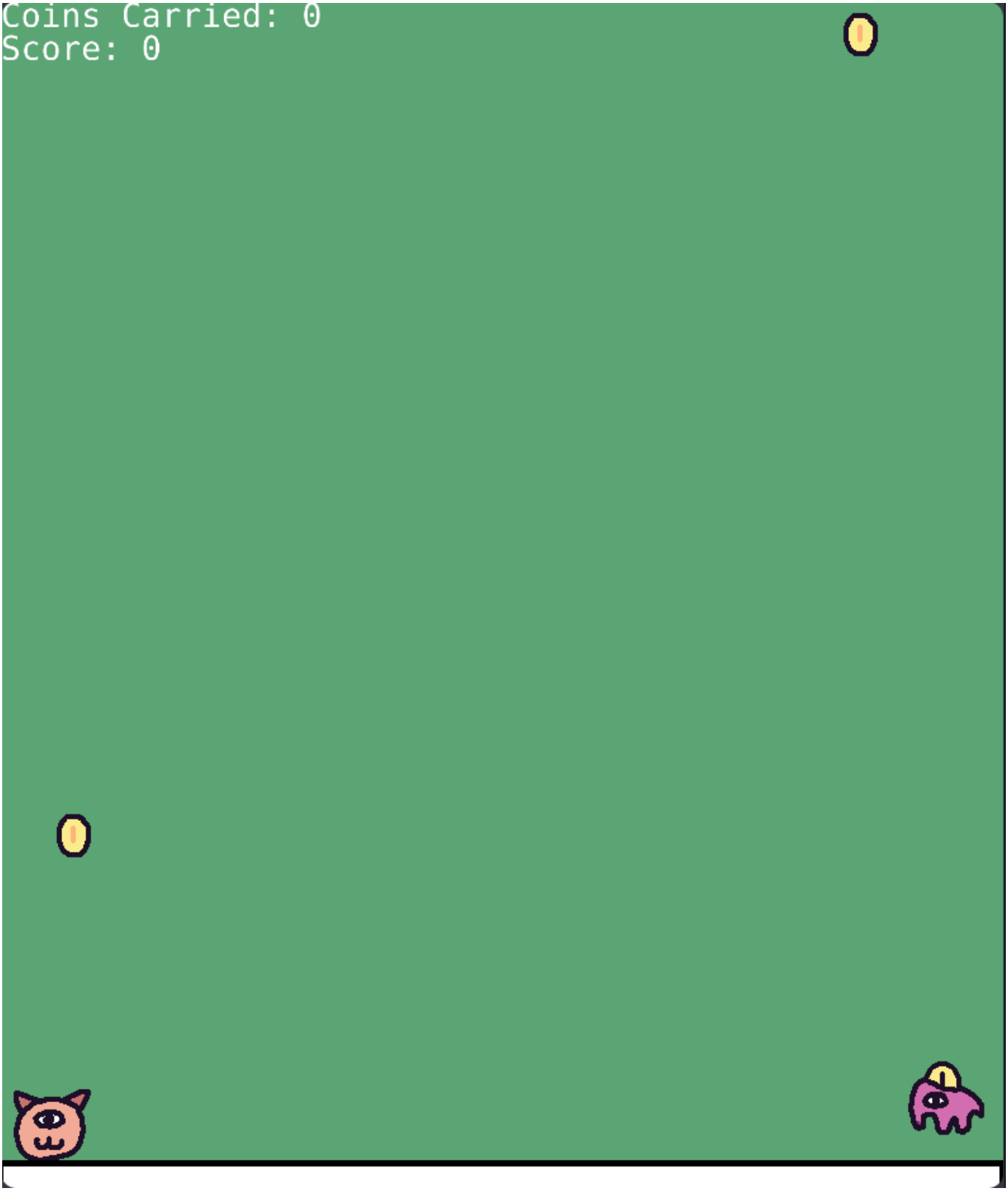


Oh, that doesn't look right. Both the "Coins Carried" and the "Score" text are drawn in the same place. Let's move the "Score" text down a little bit:

```
onUpdate(function () {
    drawText({
        text: "Coins Carried: " + coinsCarried,
```

```
        size: 26
    })
    drawText({
        text: "Score: " + score,
        size: 26,
        // new code below
        pos: vec2(0, 26)
    })
})
```

`vec2` is another way to define a pair of coordinates (the x-coordinate and the y-coordinate). We want it to appear below the other text (which is rendered at `(0, 0)`), so we keep the x-coordinate the same and change the y-coordinate to be greater. Now, it should look like this:



Great, we have the score on the screen now. Next, remember the collision code with the goal?

```
// on code from before
goal.onCollide("player", function () {
    let bubble = add([
        anchor("center"),
        pos(center()),
```

```
        rect(400, 100, { radius: 8 }),
        outline(4, BLACK),
    ])
    bubble.add([
        anchor("center"),
        text("You did it!", {
            size: 26,
        }),
        color(BLACK),
    ])
})
```

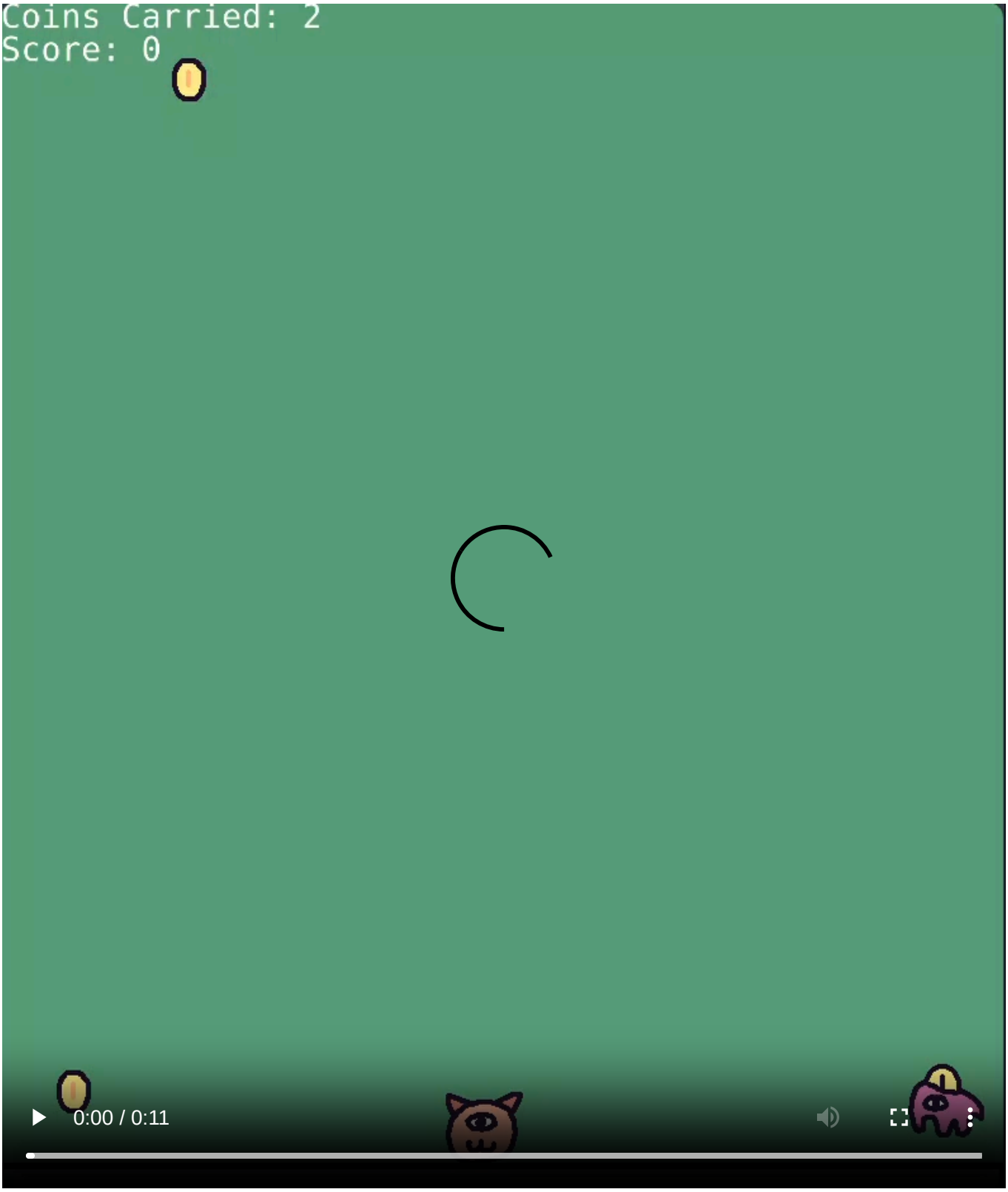
We haven't changed this yet: it will still show the "You did it" text:

Let's change this to update `score` and `coinsCarried`:

```
goal.onCollide("player", function () {
    // all new code in here
    score = score + coinsCarried
```

```
    coinsCarried = 0
})
```

Let's see it in action:



Notice how our "Score" updates when we drop off our coins? And that our "Coins Carried" resets to 0? It happens that way because we updated the variables to those values.

Add Dying

We're almost done! There's one last thing we need to update before this is ready for us to play-test: the losing condition. In the demo, if you try to collect more coins than you can carry, you lose! We'll implement this by removing the player from the screen (and showing a dramatic explosion just for fun).

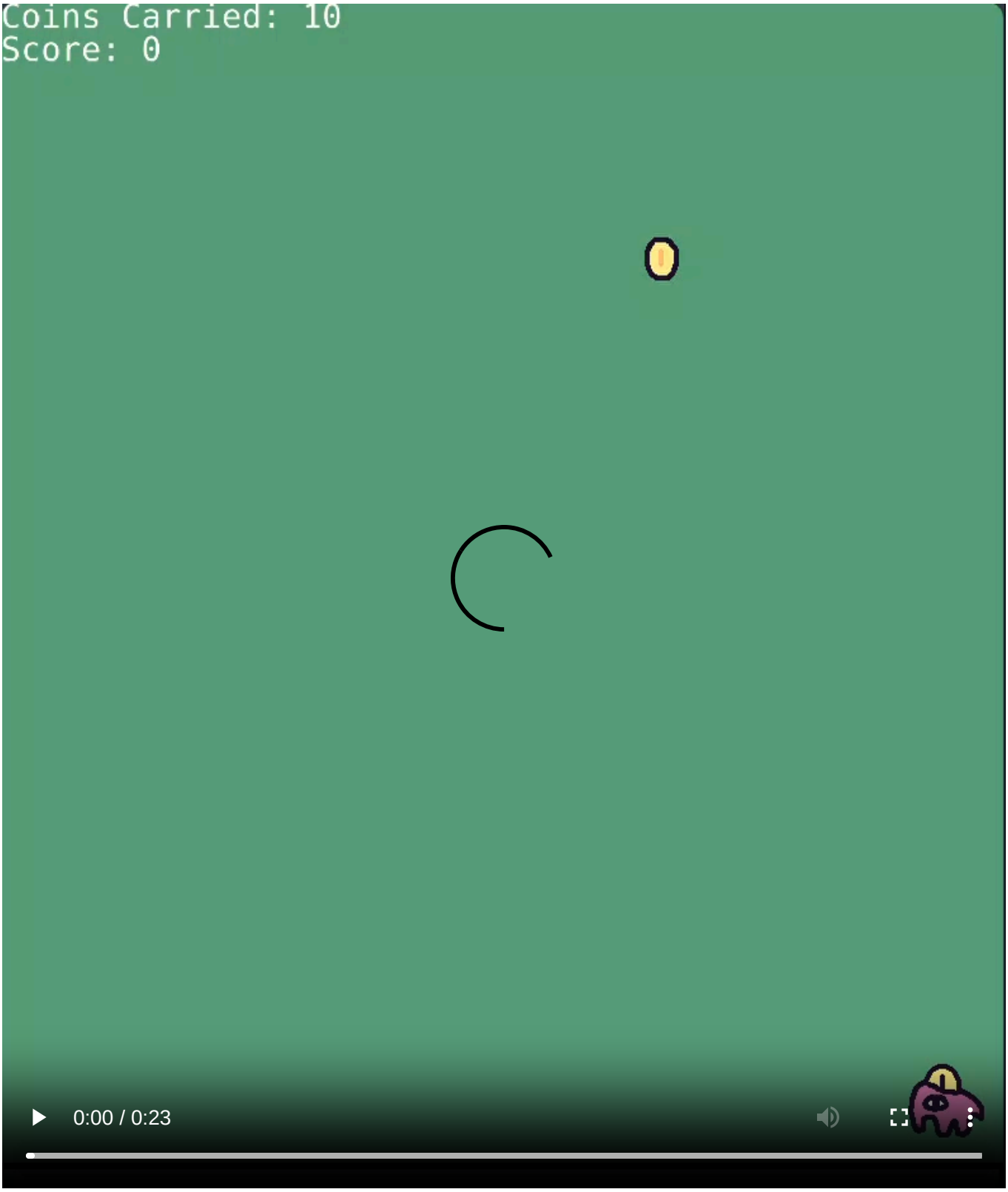
Let's change the player's `onCollide` function to add an `else` statement:

```
kat.onCollide("coin", function (coin) {
    if (coinsCarried < maximumCoinsCarried) {
        coinsCarried = coinsCarried + 1
        // new code below
    } else {
        kat.destroy()
        addKaboom(kat.pos)
        burp()
    }
    coin.destroy()
})
```

Let's break this down:

- The `else` block will be run if `coinsCarried` is equal to (or greater than) `maximumCoinsCarried`
- We call `kat.destroy()` to remove the player from the game
- We call `addKaboom` which... adds that little explosion animation
    - Pass it `kat.pos` so it shows up where the player was
- We also add a little burp sound effect using `burp()`

And...



It works! We've completed the basic functionality of the game!

Tweaking the Game

> Play the game a little bit! Are there any changes you'd want to make so it's more fun? What about easier, or harder?

> I have some ideas that I can suggest to get you thinking, but feel free to suggest some to change in front of the class. **There are no bad ideas,** but I might say no to some if it's too difficult to write the code for.

Some ideas:

- Increase speed
- Add sprinting
- Change the loop speed
- Add randomness to how often coins fall
- Make the character slower the more coins they have
- Lose points if a coin hits the ground
- Give bonus points if the player has 10 coins and puts them away
- Adding different kinds of coins that can fall
    - Maybe some give you more coins?
    - Maybe some make you lose coins?