ATMIYA UNIVERSITY

RAJKOT

A

Report On

2D ADVENTURE GAME

Under subject of

MAJOR PROJECT

B. Tech. Semester– VII

(Computer Engineering)

Submitted By:

Siddhesh Puranik **190002090**

Prof. Nirali Borad

(Faculty Guide)

Prof. Tosal M. Bhalodia

(Head of the Department)

AcademicYear

(2022-23)

<u>CANDIDATE'S DECLARATION</u>

I hereby declare that the work presented in this project entitled "**2D Adventure Game"** submitted towards completion of the project in **7th Semester** of B.Tech. (Computer Engineering) is an authentic record of my original work carried out under the guidance of "**Prof. Nirali Borad".**

I have not submitted the matter embodied in this project for the award of any other degree.

Semester: 7th
Place: Rajkot

Signature:
Siddhesh Puranik (190002090)

ATMIYA UNIVERSITY

RAJKOT

CERTIFICATE

Date:

This is to certify that the “**2D Adventure Game**” has been carried out by **Siddhesh Puranik** under my guidance in fulfillment of the subject Project in COMPUTER ENGINEERING (7th Semester) of Atmiya University, Rajkot, during the academic year 2022-23.

Prof. Nirali Borad	Prof. Tosal M. Bhalodia
(Project Guide)	**(Head of the Department)**

ACKNOWLEDGEMENT

I have put a lot of effort into this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend our sincere thanks to all of them.

I am highly indebted to Prof. Nirali Borad for their guidance and constant supervision as well as for providing necessary information regarding the Major Project titled, "**2D ADVENTURE GAME**". I would like to express my gratitude towards staff members of the Computer Engineering Department, Atmiya University, for their kind cooperation and encouragement which helped me in completion of this project.

I even thank and appreciate my colleagues in developing the project and people who have willingly helped me out with their abilities.

Siddhesh Puranik (190002090)

ABSTRACT

Little Town is an adventure game in which players control a child in an idyllic small town. By following clues, players must find the right items and bring them to one of the town's three characters - the Baker, Teacher and Grocer.

This game contains:

- 4-Directional Player Movement
- State-Based Animation
- Items you can pick and carry
- Interactable NPCs
- Do missions for NPCs
- Finish game by completing all NPC missions
- Sequence animations

INDEX

Sr. No.	Title	Page No.
	Candidate’s Declaration	**2**
	Certificate	**3**
	Acknowledgement	**4**
	Abstract	**5**
	Index	**6**
1	**Introduction**	
1.1	Introduction	8
1.2	Purpose	8
1.3	Scope	8
2	**Hardware & Software Requirements**	
2.1	Minimum Hardware Requirements	10
2.2	Minimum Software Requirements	10
3	**Steps**	
3.1	Starting New Project in GameMaker	11
3.2	Creating Objects	12
3.3	Creating Sprites	13
3.4	Creating Rooms	15
3.5	Background	17
3.6	Events	17
4	**GML**	**19**
5	**Events**	**20**
5.1	Keyboard Events to Move Player	20

5.2	Step Events	21
5.3	Vertical Movement	23
5.4	Changing all movement Events to code	25
5.5	Changing Player Sprites	26
6	**Game’s Town**	
6.1	Preparing Tile Sets	30
6.2	Designing The Town	32
6.3	Placing Tiles	33
6.4	Adding Objects To The Town	35
7	**NPCs**	
7.1	The Baker NPC	42
7.2	The Teacher NPC	43
7.3	The Grocer NPC	45
8	**Creating The Title Screen**	**47**
	Conclusion	**59**

1. Introduction

1.1 Introduction

Little Town is an adventure game in which players control a child in an idyllic small town. By following clues, players must find the right items and bring them to one of the town’s three characters - the Baker, Teacher and Grocer.

This game contains:

- 4-Directional Player Movement
- State-Based Animation
- Items you can pick and carry
- Interactable NPCs
- Do missions for NPCs
- Finish game by completing all NPC missions
- Sequence animations

1.2 Purpose

Adventure games have been around since the dawn of gaming. Though the genre has evolved, moving from text to fully-fledged 3D experiences, it's unlikely to go away permanently.

People crave a good narrative; you can't beat a well-told story. And if there's one thing that adventure games do best, it's suck you into their fictional worlds and let your imagination soar.

1.3 Scope

- **General features:**
  - GameMaker Studio 2’s Integrated Development Environment (IDE) and user interface (UI)
  - Creating and editing Objects, Sprites and Sounds
  - Coding player movement
  - Using Alarms to time actions
  - Using Draw and Draw GUI for animating and displaying graphics
  - Creating and editing Rooms
  - Designing levels with Tiles, Sprites and Objects
  - Creating and editing Cameras and Viewports
  - Playing sound effects and music
  - Creating 3D audio with Emitters and Listeners
  - Tracking collision between Objects

- **Coding (GML) specifics:**
  - Variables, global variables and booleans
  - Simple arrays for storing data
  - For loops, switch statements and with statements
  - Functions and Script Assets
  - States and enums

- **Sequences:**
  - UI and functionality overview
  - Creative use of the timeline
  - Using Broadcast Messages to communicate with other assets in the game
  - Using a control Object to listen to and affect Sequences

2. Hardware & Software Requirements

2.1 Minimum Hardware Requirements

Table 2.1.1 Hardware Requirements

Number	Description
1	Dual Core CPU
2	2 GB RAM
3	OpenGL 4-compliant onboard graphics or Dedicated GPU

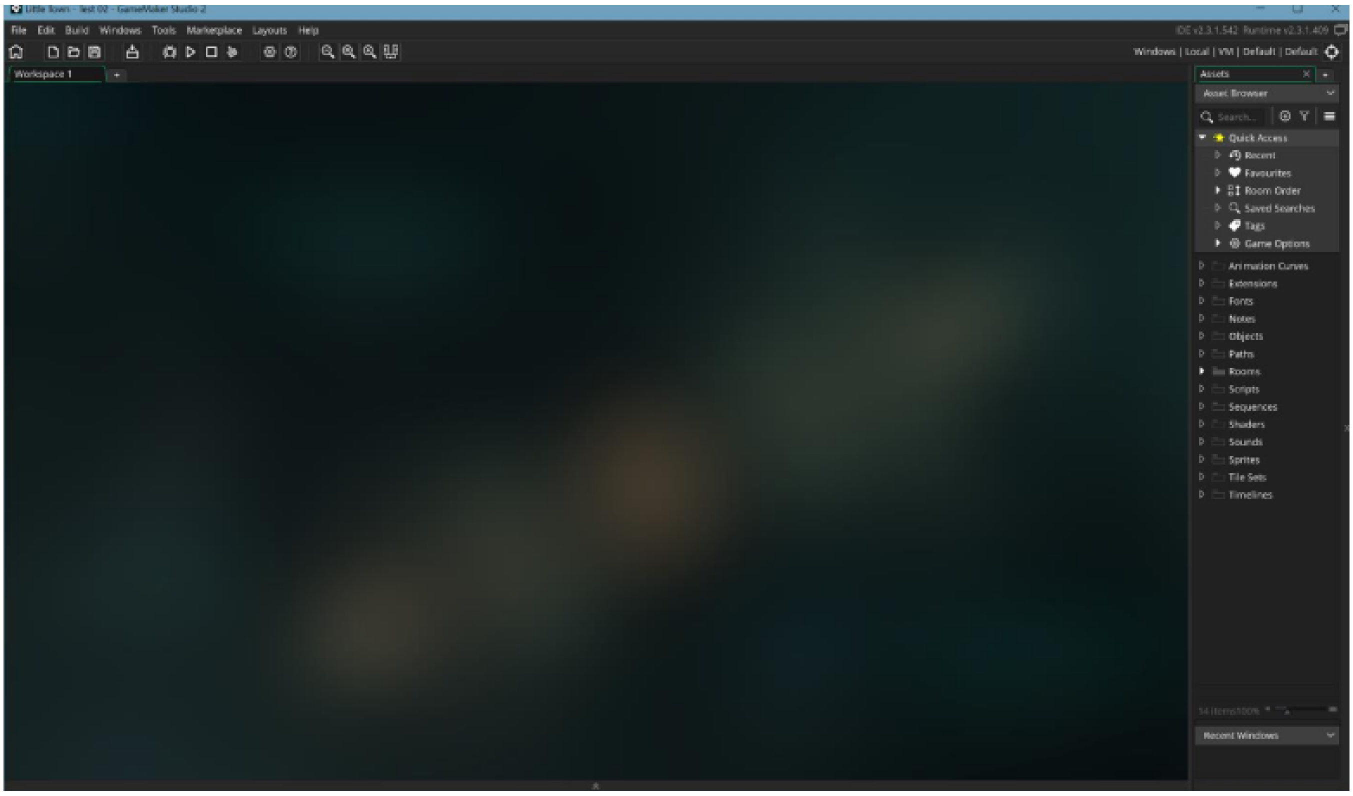
2.2 Minimum Software Requirements

Table 2.2.1 Software Requirements

Number	Description	Type
1	Operating System	Windows 7 with SP1 or macOS Mojave
2	Language	GML
3	IDE	GameMaker Studio

3. Steps

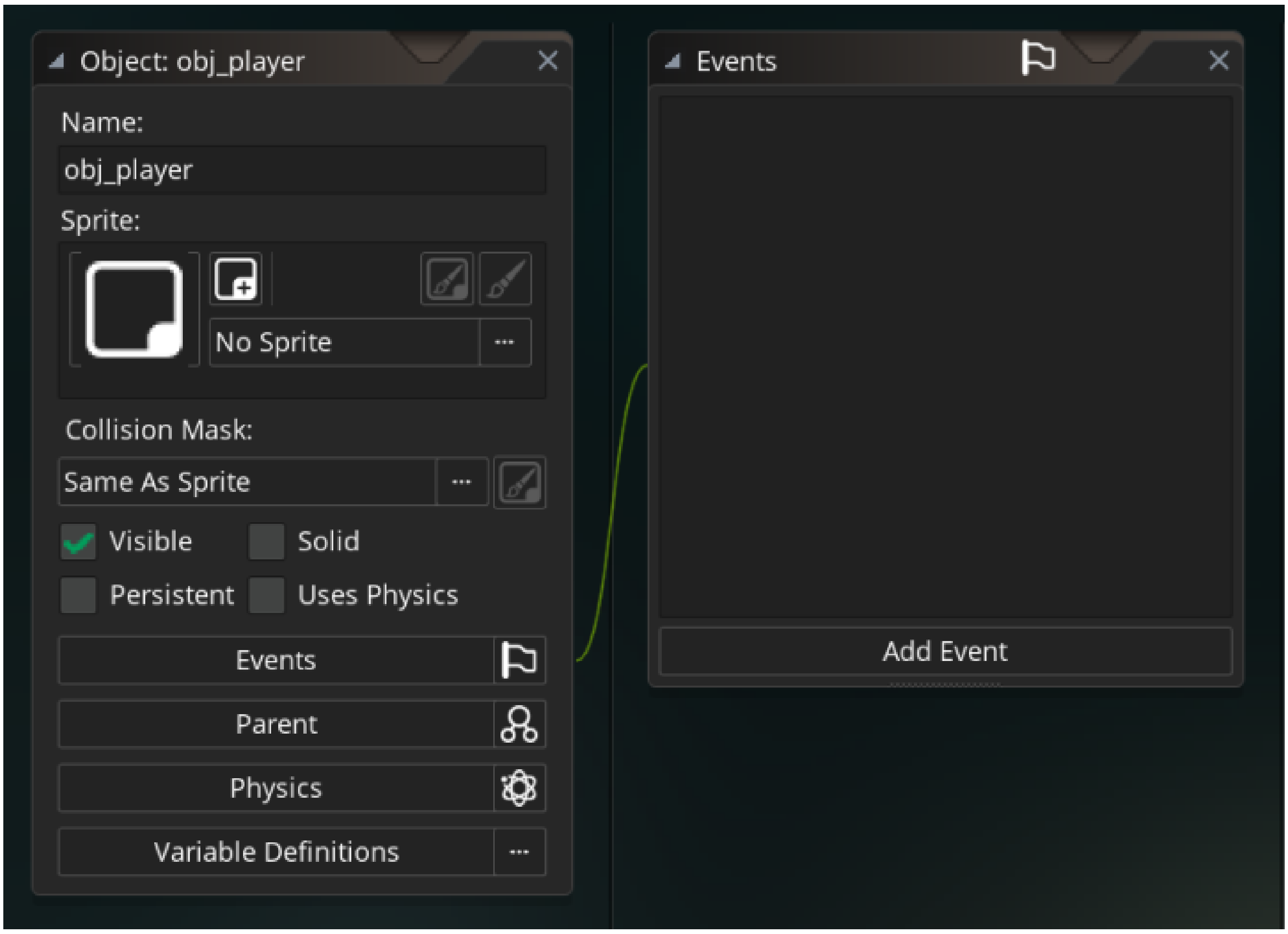
3.1 Starting New Project in Gamemaker



The GameMaker Studio 2 Integrated Development Environment (IDE). The blank area is the Workspace and the list on the right is the Asset Browser.

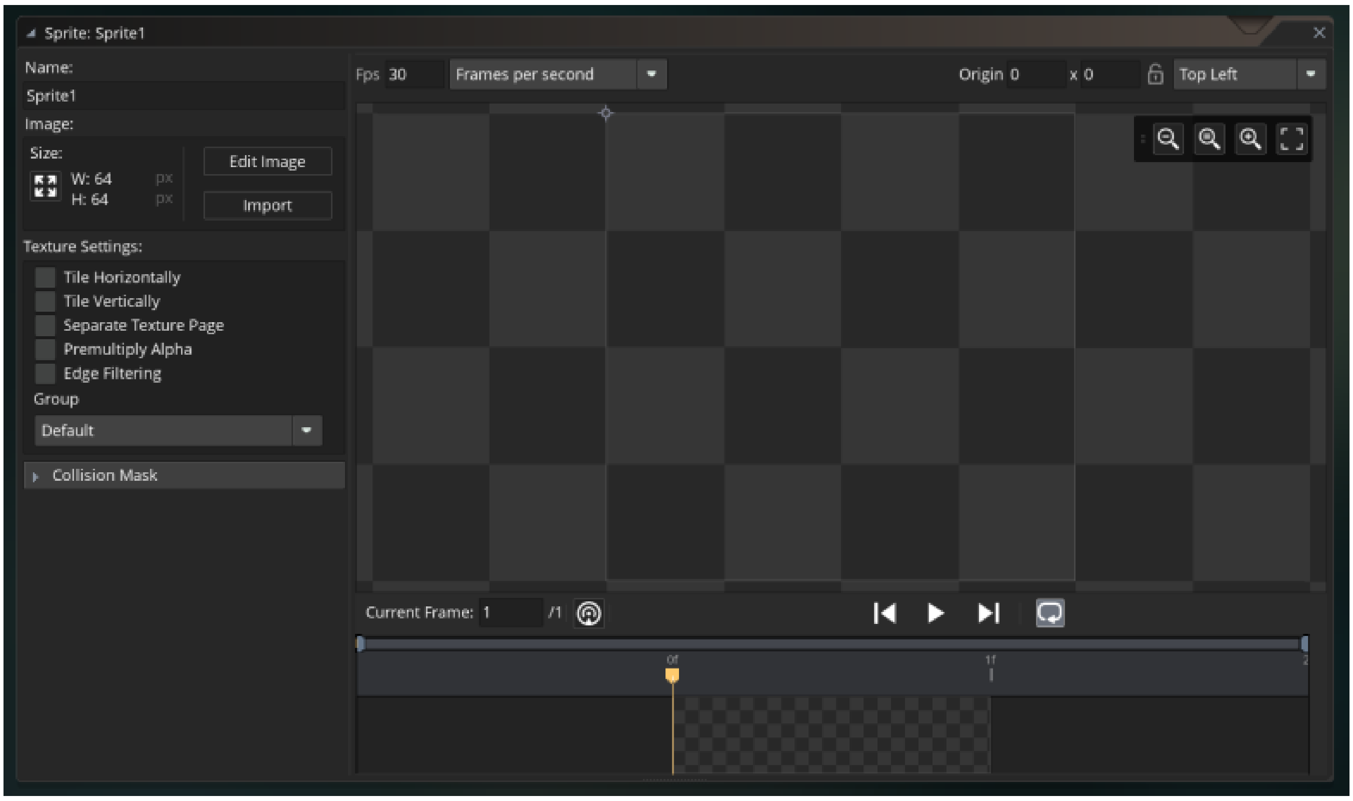
You may see by default that your Asset Browser has several folder icons with labels. These folders are called Groups, and they exist to help you organize commonly used assets (Sprites, Sounds, Objects, etc.). You can also add new groups as you need them or remove ones you don't.

3.2 Creating Objects



Objects are the heart of GameMaker Studio 2. They can be players, enemies, items, obstacles, but they can also be health bars, textboxes, title screens and even invisible stuff that keeps track of our game behind the scenes. In short, if something must do something, it's probably going to be an Object.

3.3 Creating Sprites


Sprite: Sprite1

Name: Sprite1

Size: W: 64 px H: 64 px

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

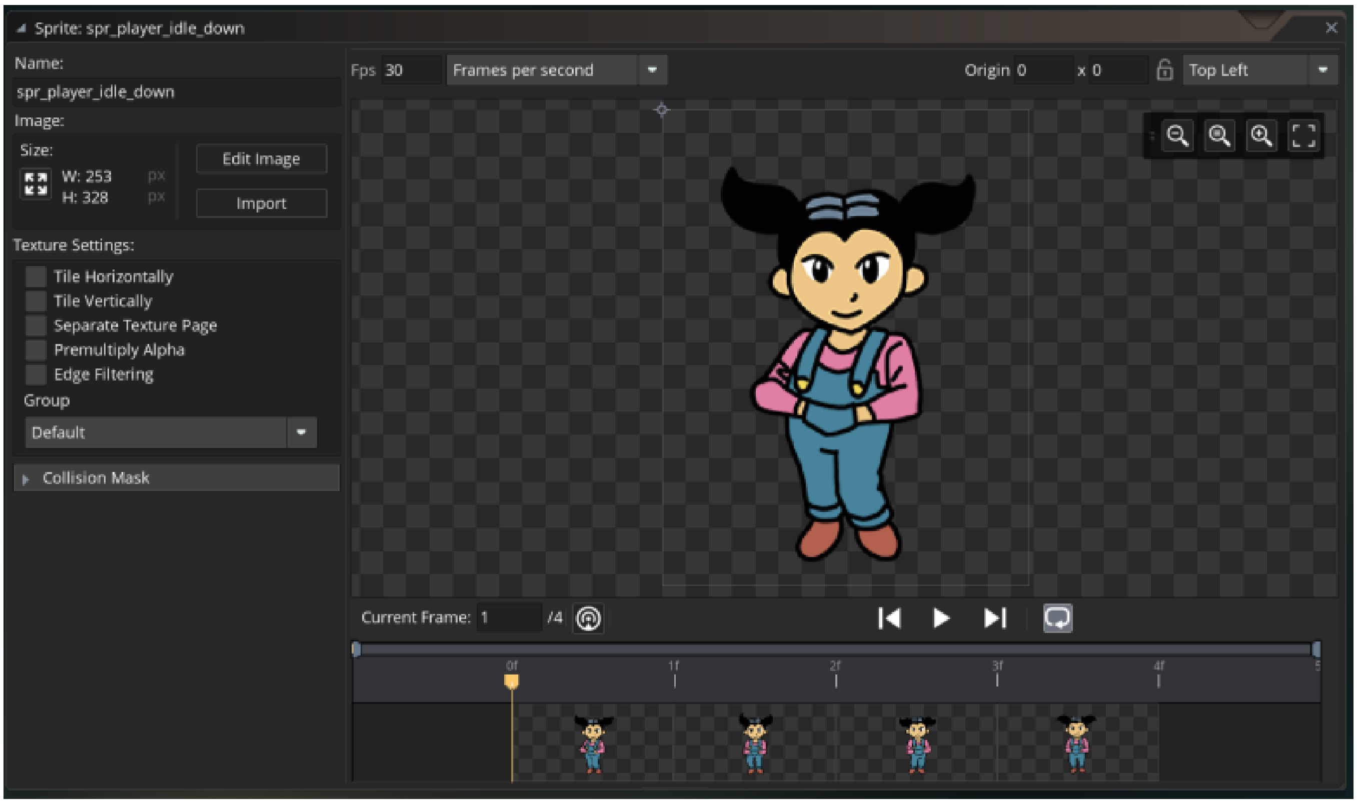
Group: Default

Fps 30 Frames per second

Origin 0 x 0 Top Left

Current Frame: 1 /1

Sprite in the Sprite Editor


Sprite: spr_player_idle_down

Name: spr_player_idle_down

Size: W: 253 px H: 328 px

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

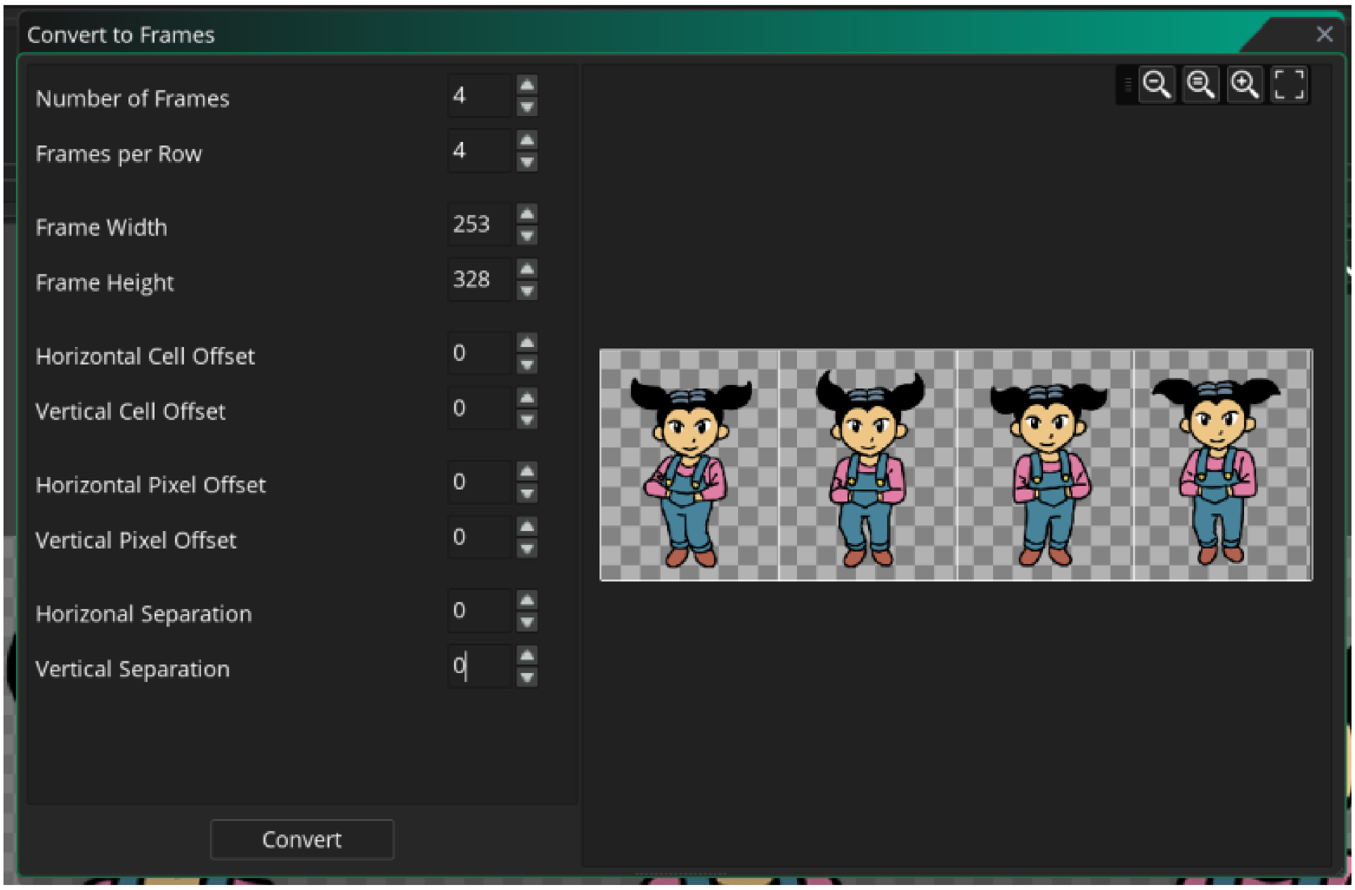
Group: Default

Fps 30 Frames per second

Origin 0 x 0 Top Left

Current Frame: 1 /4

In the Sprite Editor, click the Edit Image button. You'll be taken to a large version of your Sprite with several drawing and selection tools. Note that doing this has opened a new tab at the top of the IDE. You can click the X on this tab to close this at any time, or you can click on the "Workspace" tab to return to where we were. In the menu bar at the top of the IDE, click Image > Convert to Frames. This new window will allow us to "slice" our Sprite into individual frames. To do so, change the values like so:


Convert to Frames

Number of Frames	4
Frames per Row	4
Frame Width	253
Frame Height	328
Horizontal Cell Offset	0
Vertical Cell Offset	0
Horizontal Pixel Offset	0
Vertical Pixel Offset	0
Horizonal Separation	0
Vertical Separation	0

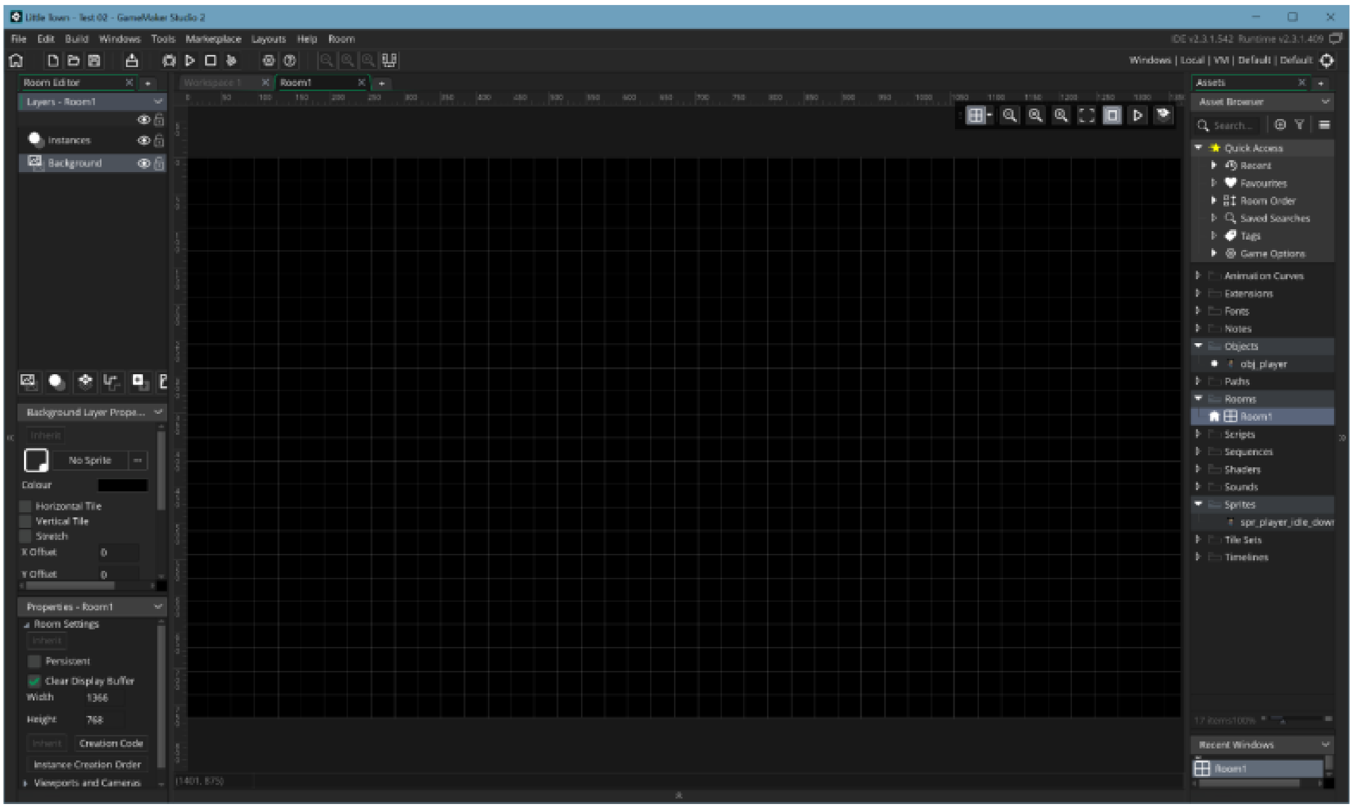
3.4 Creating Rooms

Now that we have a player object we can see, let’s put it in a Room! Luckily, we already have one to use because GameMaker Studio 2 always creates one Room for us by default. In the Asset Browser, open the folder called Rooms and double-click Room1. It will open our first Room, which is, pretty boring.

A Room in GameMaker Studio 2 is like a level or scene in your game. But it’s more than that; a Room can be a title screen, a loading screen, a credits screen, or almost anything else. But like the name implies, it’s a place where stuff happens. Right now, though, nothing of interest is happening. So, let’s change a few things. On the left is your Room Editor, which has all sorts of important things.

From top to bottom you’ll see:

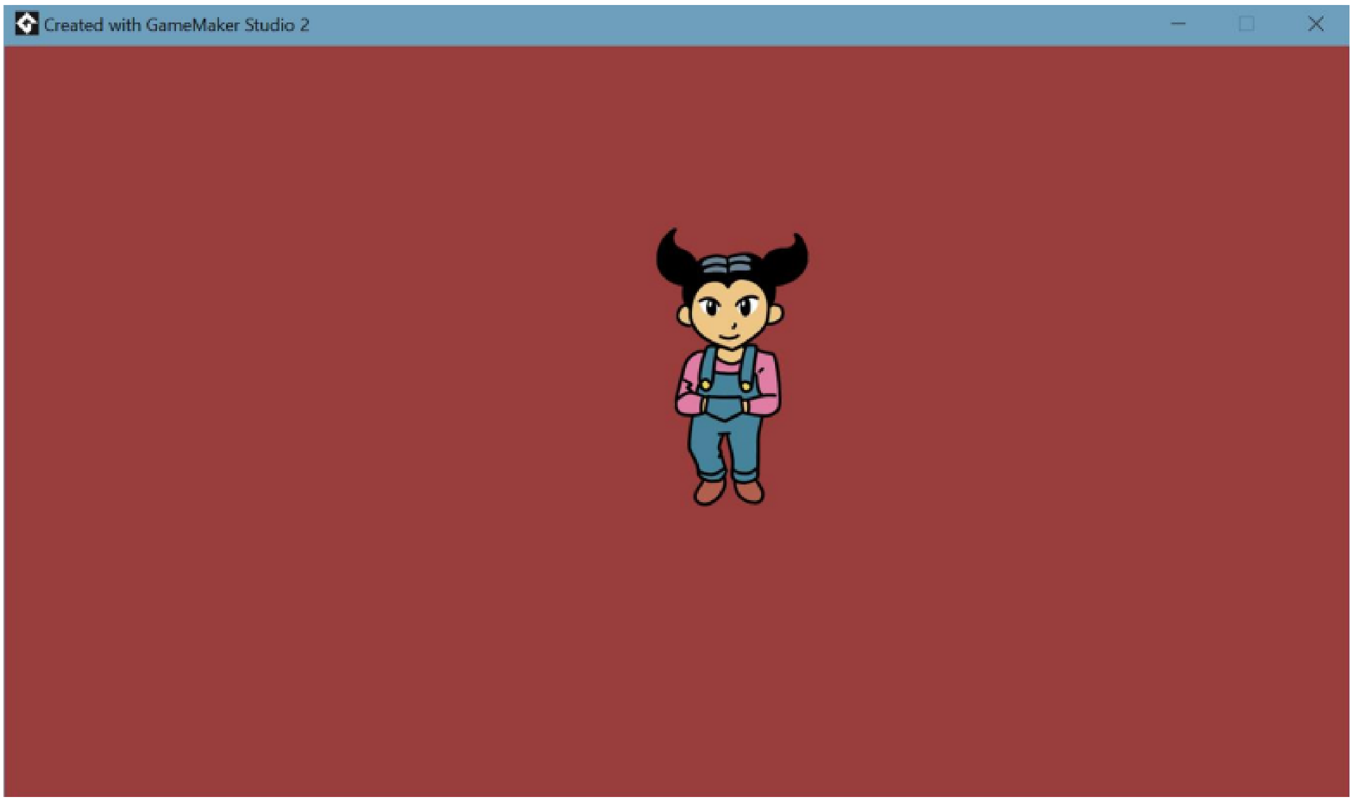
- Room Layers: this is where we’ll place Objects, background images and more
- Layer Properties: this lets you change all sorts of stuff for a particular layer
- Room Settings: here we can change the Room size and edit Cameras, which we’ll do later



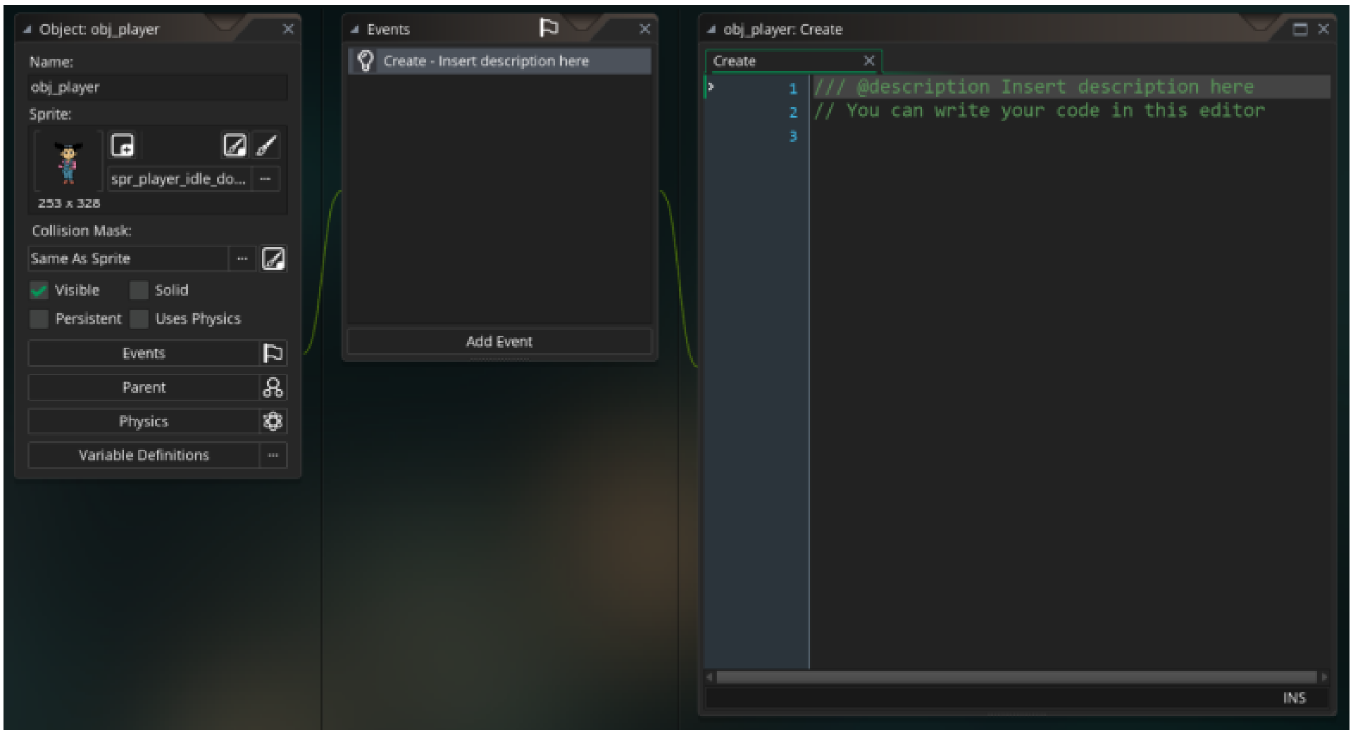
For now, click on the Background layer and look at its Properties in the section on the lower left. (If things look squished or out of the way, you can adjust the size of the different sections here.) Click on that black void beside Colour and pick any colour you like.

Now, click on the Instances layer in the Layers section. Then look at your Asset Browser (it should still be on the right of the IDE). Find our obj_player Object and drag it onto the room to place it.

3.5 Background



3.6 Events



```
/// @description Insert description here
// You can write your code in this editor
```

The Create Event happens when an Object is “created” within the game — in other words, when it first appears. Think of a space shooter where mashing a button shoots little bullets; each of those bullets is an Object and as soon as they appear, they each run their Create Event.

Since we’ve placed an instance of obj_player into our first Room, this Create Event will happen as soon as that Object appears in the Room — which is to say, as soon as we see the Room in our game.

At the top of each Event is a line marked /// @description.

You can add something after this, like “Code for this Event” to make titles. These titles appear on the left in the Events list for that Object and can help you stay organized.

4. GML

GameMaker Language Code

GameMaker Studio 2 uses its own programming language called GameMaker Language, or GML.

Enter our first code in obj_player's new Create Event, underneath the @description line:

```
// Variables

walkSpeed = 16;
vx = 0;
vy = 0;
dir = 3;
moveRight = 0;
moveLeft = 0;
moveUp = 0;
moveDown = 0;
```

These are all variables. A variable is a way to store information, called a value. GameMaker Studio 2 has built-in variables (i.e., things that always exist), but we can also make our own, like magic.

Here, we're creating several variables and assigning values to them for later use.

A line that starts with 2 or more forward slashes (such as // This is a comment) is "commented out," meaning it is not code that will run.

5. Events

5.1 Keyboard Events to Move Player

The first thing we want to do is be able to move our player Object around by using the keyboard, so let’s add another Event.

With obj_player open in the Object Editor, click Add Event and choose Key Down > Right.

EVENT	**WHAT IT MEANS**
Key Down	Occurs if that key is being held down
Key Pressed	Occurs only when that key is first pressed down
Key Up	Occurs when that key is released

In the new Key Down – Right Event, write the following code:

```
moveRight = 1;
```

As you can see, we’re changing the value of the variable moveRight to 1.

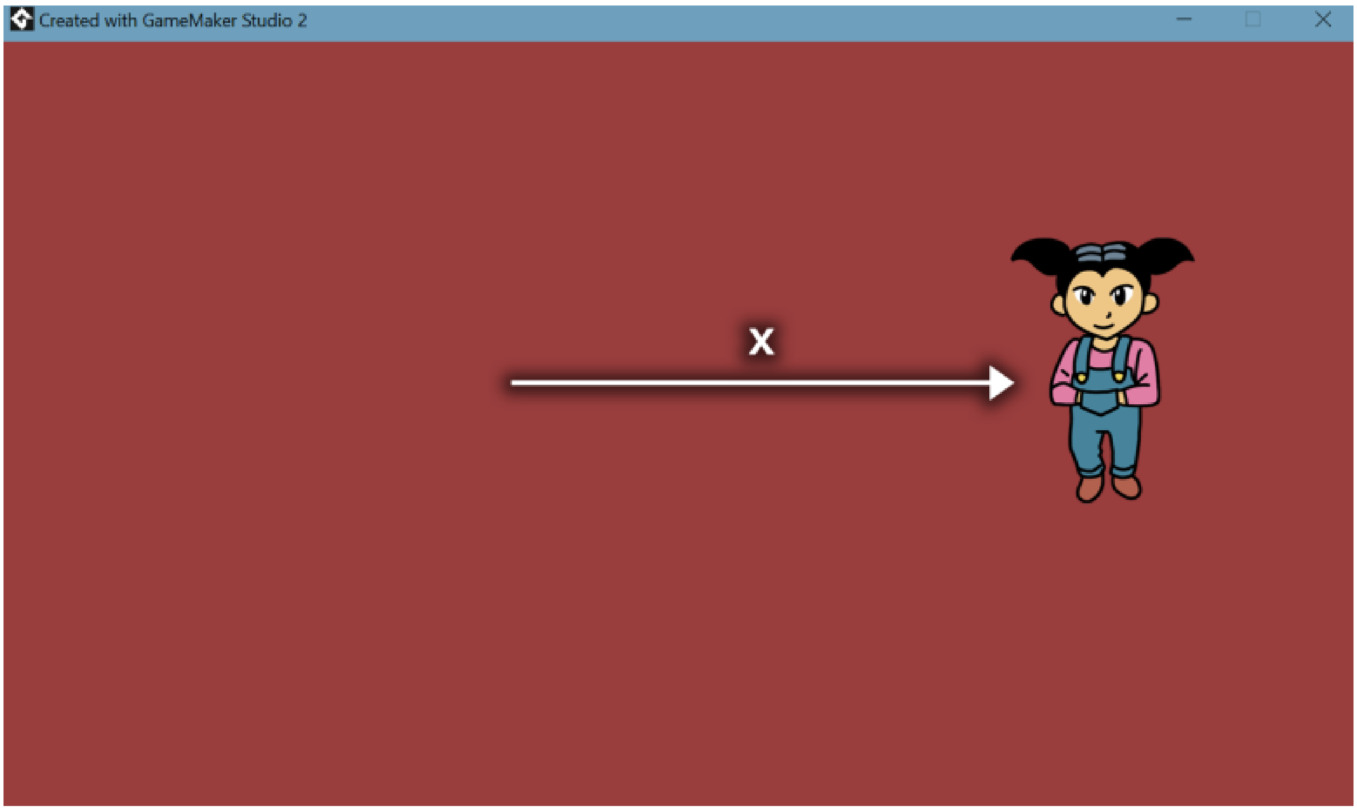
In the Object Editor, click “Add Event” again and choose Key Up > Right.

In this new Key Up – Right Event, add this line of code:

```
moveRight = 0;
```

So here we’re resetting moveRight back to 0 when we release the right arrow key.

5.2 Step Events



In the Object Editor, click Add Event and choose Step > Step. Note that there are multiple Step Events, but we want the one just called Step.

The Step Event happens every single frame of the game, so we must use it wisely. If our game is 60 FPS, then our player Object is going to run whatever code is in here 60 times every second.

In this new Event, add the following code:

```
// Calculate movement
vx = (moveRight * walkSpeed);
// If Idle if (vx == 0) {
        // do nothing for now }
// If moving
if (vx != 0) {
        x += vx; }
```

Code	**Working**

// Calculate movement
vx = (moveRight * walkSpeed);
	We're calculating the value of vx based on two other values: moveRight and walkSpeed:
	- We set moveRight to 1 when we're holding down the right arrow key (in the Key Down – Right Event)
	- We reset moveRight to 0 when we release the right arrow key (in the Key Up – Right Event)
	- In obj_player's Create Event, we set walkSpeed to 16
	Therefore, vx can be either 16 (1 x 16) or 0 (0 x 16)

// If Idle
 if (vx == 0) {
// do nothing for now
}
	If vx is equal to 0, do something. (Right now, we're not doing anything in particular, so we just have a comment here)

// If moving
 if (vx != 0) {
 x += vx;
 }
	If vx is not equal to 0, then add the value of vx to obj_player's x position.
	
	(In GML, += means "add to". Not unexpectedly, -= means "subtract from.")

When your game window opens, press the right arrow key; the player should move to the right.

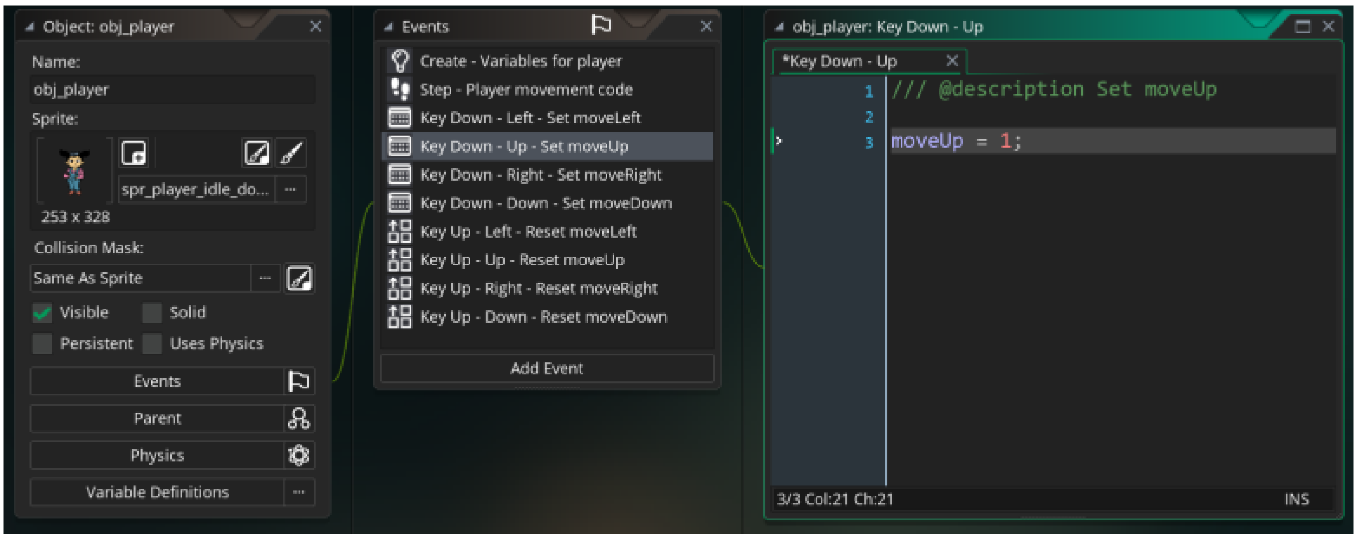
Release the right arrow key, and the player should stop moving.

5.3 Vertical Movement

Event	Code to type
`Key Down - Down`	`moveDown = 1;`
`Key Up - Down`	`moveDown = 0;`
`Key Down - Up`	`moveUp = 1;`
`Key Up - Up`	`moveUp = 0;`

Now that we have these principles in place, we can make our player Object move up and down too.

With obj_player open, use the Add Event button to add the following Events, and enter the code shown here for each.


Object: obj_player

Name: obj_player

Sprite: spr_player_idle_do...

253 x 328

Collision Mask: Same As Sprite

✓ Visible ☐ Solid

☐ Persistent ☐ Uses Physics

Events

- Create - Variables for player
- Step - Player movement code
- Key Down - Left - Set moveLeft
- Key Down - Up - Set moveUp
- Key Down - Right - Set moveRight
- Key Down - Down - Set moveDown
- Key Up - Left - Reset moveLeft
- Key Up - Up - Reset moveUp
- Key Up - Right - Reset moveRight
- Key Up - Down - Reset moveDown

obj_player: Key Down - Up

*Key Down - Up

```
/// @description Set moveUp

moveUp = 1;
```

3/3 Col:21 Ch:21 INS

When you have these Events in place, open the Step event again, and edit the three code blocks we've written there like so:

```
// Calculate movement
vx = ((moveRight - moveLeft) * walkSpeed);
vy = ((moveDown - moveUp) * walkSpeed);

// If Idle
if (vx == 0 && vy == 0) {
   // do nothing for now
   }

// If moving
if (vx != 0 || vy != 0) {
   x += vx;
   y += vy;
   }
```

You can see we've added vertical movement to our player with these additions. We've also added checks for vy in the second and third blocks.

You should be able to move the player Object around in all four directions!

5.4 Changing all movement Events to code

open obj_player and its Step event. Enter the following new code above the // Calculate movement code block:

```
// Check keys for movement
moveRight = keyboard_check(vk_right);
moveUp = keyboard_check(vk_up);
moveLeft = keyboard_check(vk_left);
moveDown = keyboard_check(vk_down);
```

Now, in the Object Editor, we're going to delete all eight of those Key Events, because we don't need them anymore.

You can right-click on each Key Down and Key Up event and choose Delete Event.

You can also shift-click a group of Events all at once and use the same right-click menu to delete them all in one go.

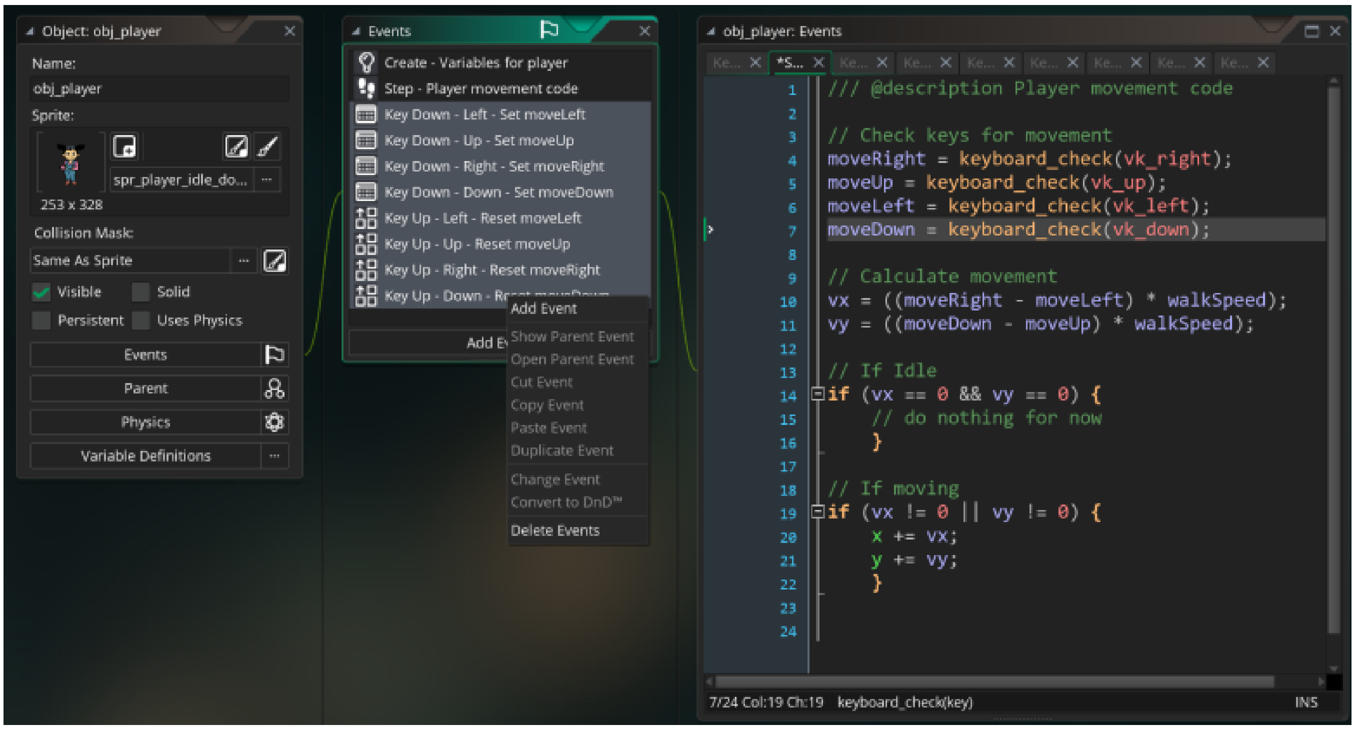
You'll receive a warning when you do this.Once you've deleted these Events, run your game again and try the arrow keys on the keyboard.

Our player Object should be moving just like before! When you're ready, close your running game and return to GameMaker Studio 2.

The function keyboard_check() requires a single argument — a detail it needs to do its job; in this case, the key we want our game to check.

Here, we're checking the right arrow key (called vk_right in GML). And, as it turns out, the keyboard_check() function does the exact same thing as the Key Down Events we used at first! So, keyboard_check(vk_right) is the same as creating a Key Down – Right Event.

`moveRight`	`=`	`keyboard_check(vk_right)`
Update the variable `moveRight`...		... with the result we get by checking if the player is pressing the right arrow key


Object: obj_player
Name: obj_player
Sprite: spr_player_idle_do...
253 x 328
Collision Mask: Same As Sprite
Visible ✓ Solid
Persistent Uses Physics

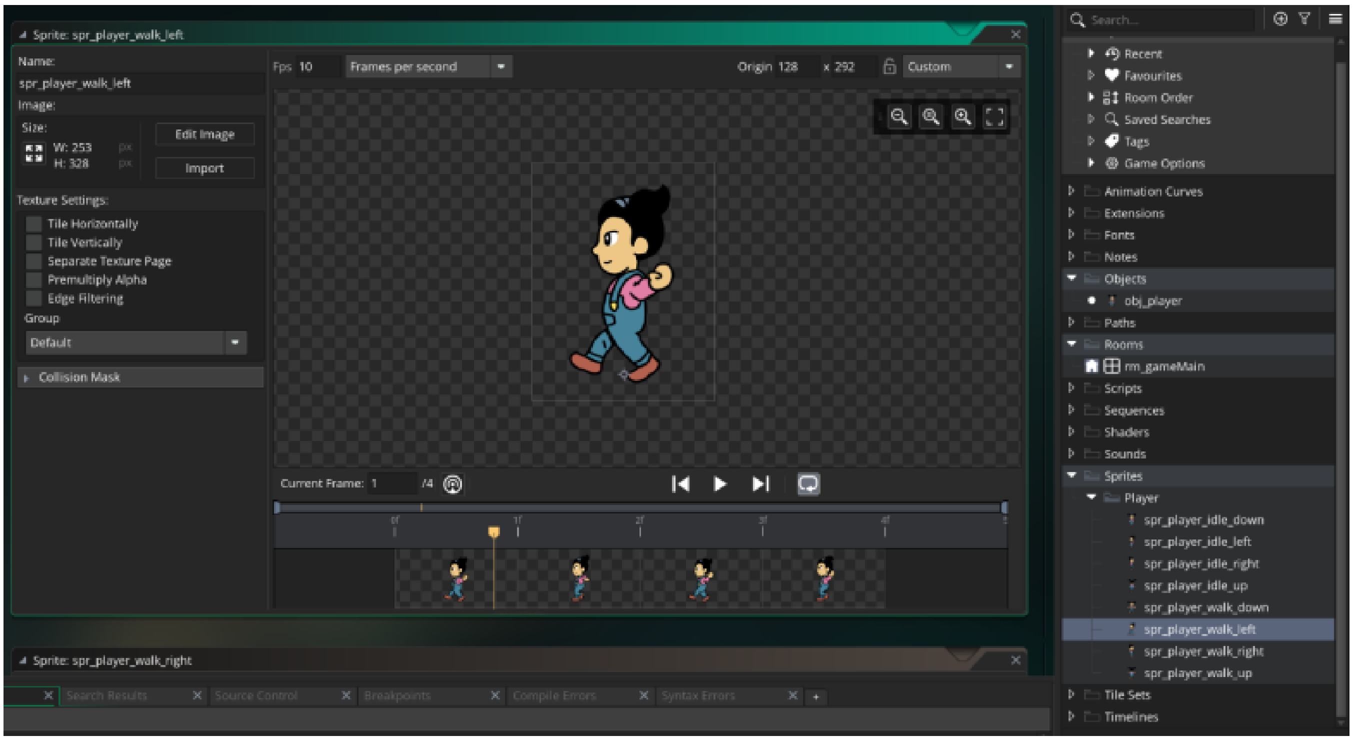
Events
- Create - Variables for player
- Step - Player movement code
- Key Down - Left - Set moveLeft
- Key Down - Up - Set moveUp
- Key Down - Right - Set moveRight
- Key Down - Down - Set moveDown
- Key Up - Left - Reset moveLeft
- Key Up - Up - Reset moveUp
- Key Up - Right - Reset moveRight
- Key Up - Down - Reset moveDown

obj_player: Events

```
/// @description Player movement code

// Check keys for movement
moveRight = keyboard_check(vk_right);
moveUp = keyboard_check(vk_up);
moveLeft = keyboard_check(vk_left);
moveDown = keyboard_check(vk_down);

// Calculate movement
vx = ((moveRight - moveLeft) * walkSpeed);
vy = ((moveDown - moveUp) * walkSpeed);

// If Idle
if (vx == 0 && vy == 0) {
    // do nothing for now
    }

// If moving
if (vx != 0 || vy != 0) {
    x += vx;
    y += vy;
    }
```

7/24 Col:19 Ch:19 keyboard_check(key) INS

5.5 Changing Player Sprites


Sprite: spr_player_walk_left
Name: spr_player_walk_left
Size: W: 253 H: 328
Fps 10 Frames per second
Origin 128 x 292 Custom
Current Frame: 1 /4

Sprites / Player:
- spr_player_idle_down
- spr_player_idle_left
- spr_player_idle_right
- spr_player_idle_up
- spr_player_walk_down
- spr_player_walk_left
- spr_player_walk_right
- spr_player_walk_up

Change the player Object's Sprite based on whether it's idle (standing still) or walking. Do so based on its direction (up, down, left, right).

First, let's get all our other assets ready. Using File Explorer (Windows) or Finder (Mac), navigate to the Assets folder provided with this course. Open the Sprites > Characters and Items folder and drag these Sprites into the Sprite group in GameMaker Studio 2's Asset Browser:

- sprPlayer_idle_up_strip04
- sprPlayer_idle_right_strip04
- sprPlayer_idle_left_strip04
- sprPlayer_walk_up_strip04
- sprPlayer_walk_right_strip04
- sprPlayer_walk_down_strip04
- sprPlayer_walk_left_strip04

Once each new Sprite Strip has correctly been converted into a Sprite with frames, you can remove its _stripXX suffix.

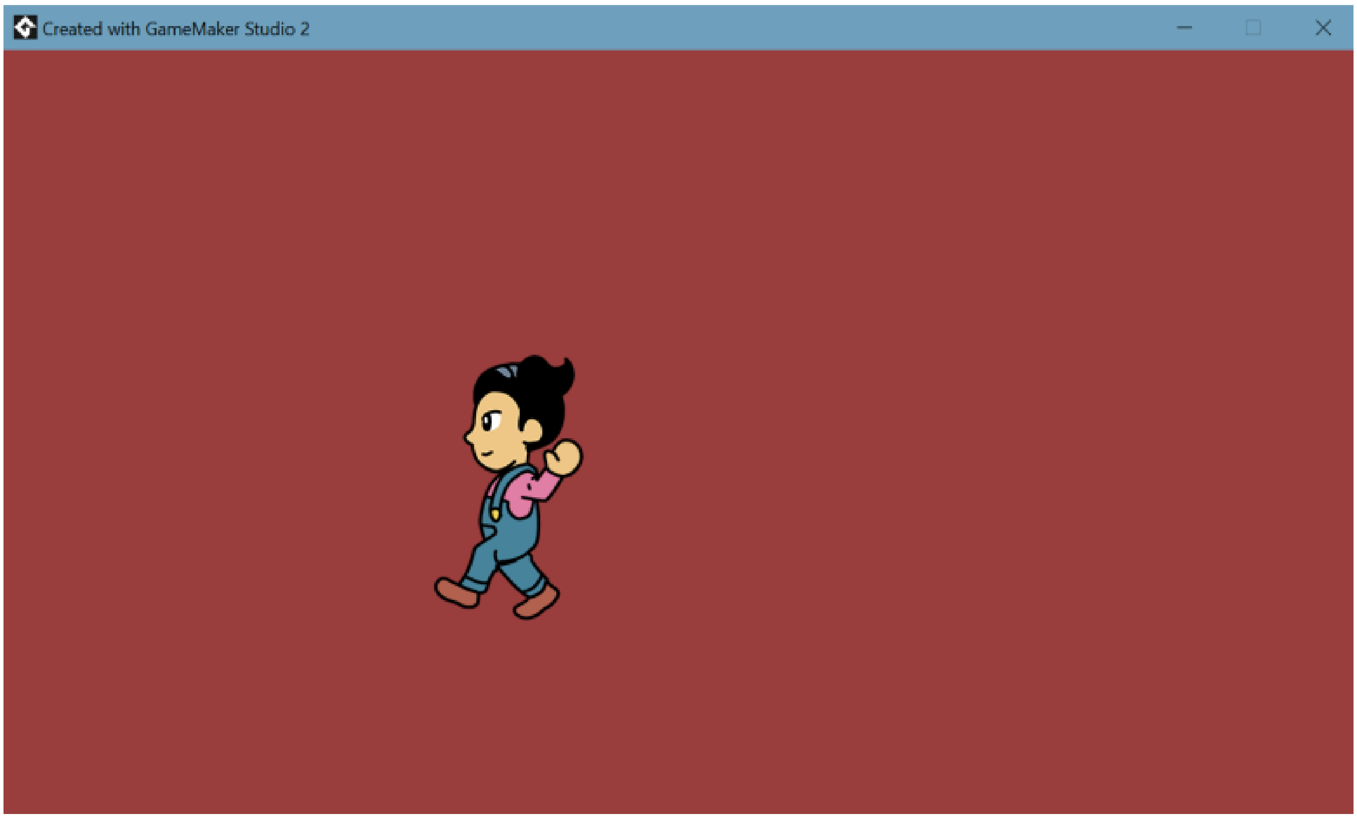
Make sure the origin point for each of these new Sprites is the same as what you else things will get weird. You can see that first Sprite (spr_player_idle_down)'s origin x and y values by double-clicking the Sprite in the Asset Browser; in the top-right of the Sprite Editor window are two Origin fields.

Here, we've converted one of the new player Sprites into frames; set its FPS to 10 to match the original sprite; changed its Origin to match as well; and organized these new player Sprite assets within the Asset Browser.
Once you have all the new Sprites dealt with, open obj_player and its Step Event again.
Update the // If moving code block like so:

```
// If moving
if (vx != 0 || vy != 0) {
   x += vx;
   y += vy;

   // Change walking Sprite based on direction
   if (vx > 0) {
      sprite_index = spr_player_walk_right;
      dir = 0;
```

```
        }
    if (vx < 0) {
        sprite_index = spr_player_walk_left;
        dir = 2;
        }
    if (vy > 0) {
        sprite_index = spr_player_walk_down;

        dir = 3;
        }
    if (vy < 0) {
        sprite_index = spr_player_walk_up;
        dir = 1;
        }
}
```

Since we're already using vx and vy for movement, we can check it here with a bunch of if statements. It's not the most efficient way to do this, but it totally works.



We also set dir to a different value in each case so we can use it in the next bit.

You should see your player moving in all four directions, with the correct Sprites for each direction.

Back in obj_player's Step Event, replace the "If idle" block with new code:

```
// If Idle
if (vx == 0 && vy == 0) {
    // Change idle Sprite based on last direction
    switch dir {
        case 0: sprite_index = spr_player_idle_right; break;
        case 1: sprite_index = spr_player_idle_up; break;
        case 2: sprite_index = spr_player_idle_left; break;
        case 3: sprite_index = spr_player_idle_down; break;
        }
    }
```

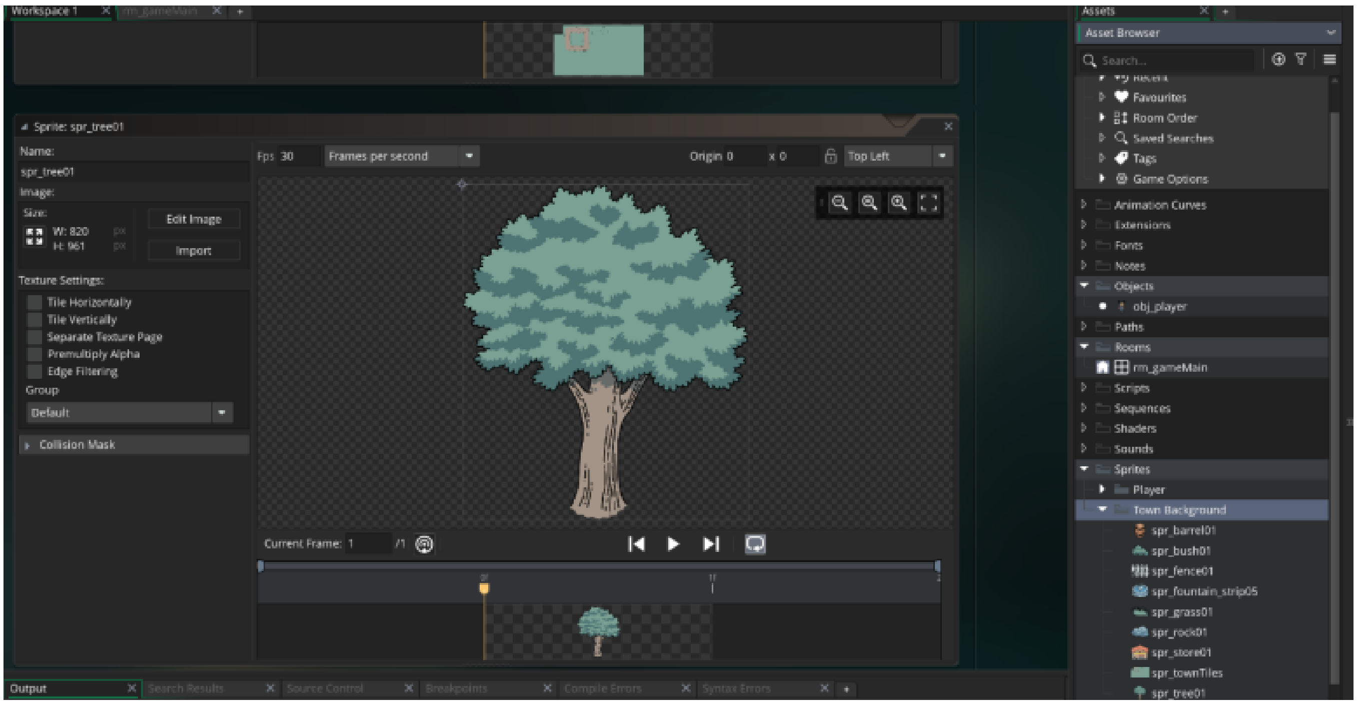
This new code is called a switch function. It's a handy way to combine a bunch of if statements into something easier to read.

Here, we're checking the dir variable (which we set whenever we're moving) and applying the correct idle Sprite for the last direction that obj_player was moving.

Run the game again and see the difference; now our player correctly moves and stops!

6. Game’s Town

6.1 Preparing Tile Sets



Using File Explorer (Windows) or Finder (Mac), navigate to the Assets folder provided with this course. Go to Sprites > Backgrounds and drag the files there into the Asset Browser. As mentioned before, you can organize these assets if you want to (for example, by putting them in a Town Background group within the Sprites group.).

You’ll see a bunch of Sprites for things like trees and bushes and a big image called spr_townTiles. Open this image from the Asset Browser.

Most games build levels and areas with reusable assets that designers can place and nudge to create a scene. In GameMaker Studio 2, we can use Tiles to do this. They are just as the name suggests; tiled graphics that we can use over and over. This file (spr_townTiles), is going to let us design a town by creating a Tile Set.

Tile Sets need two pieces: a Tile Set asset and a Sprite attached to that asset.

We already have the Sprite (spr_townTiles) so we just need to create the asset.

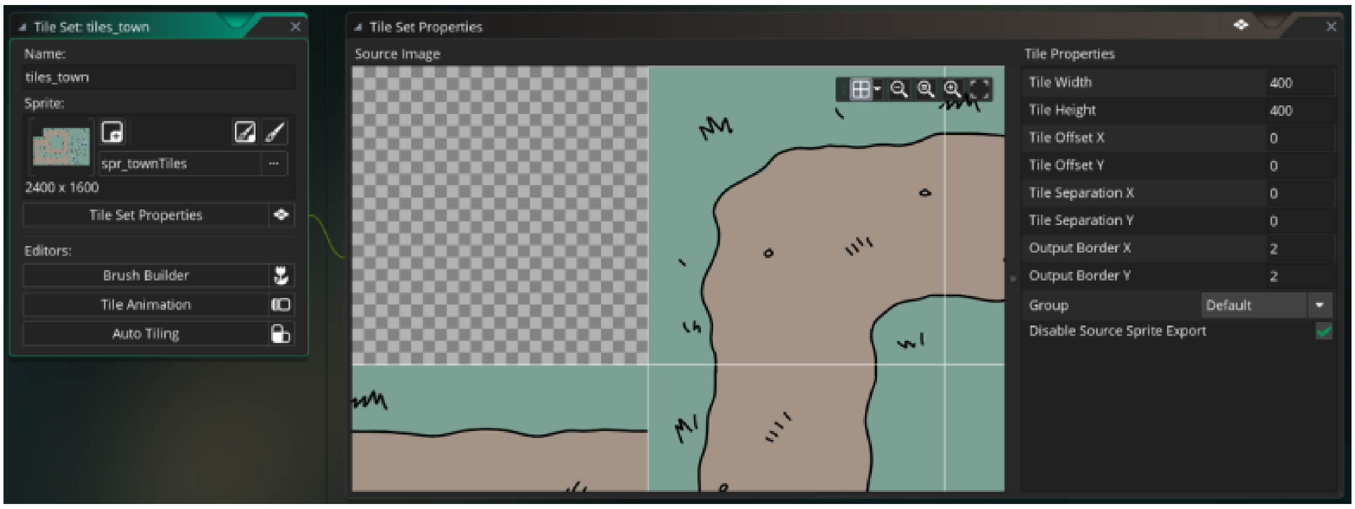
In the Asset Browser, you’ll see a pre-existing group called Tile Sets. You can right-click on this and choose Create > Tile Set.

In the Tile Set editor that pops up, name this tiles_town.

Click the Select Sprite menu to choose the Sprite you just imported (spr_townTiles). You’ll see your Tile Set now has a grid and probably doesn’t make much sense.

That grid is the tile grid and as you can see, it’s way too small. So, on the right side of the Tile Set editor, change both the Tile Width and Tile Height to 400.

Now the grid should look correct and you’ll see a blank space in the top-left.



Name: tiles_town

spr_townTiles

2400 x 1600

Tile Width	400
Tile Height	400
Tile Offset X	0
Tile Offset Y	0
Tile Separation X	0
Tile Separation Y	0
Output Border X	2
Output Border Y	2
Group	Default
Disable Source Sprite Export	✓

Our Tile Set with Sprite attached and Tile Width and Height set.

6.2 Designing The Town

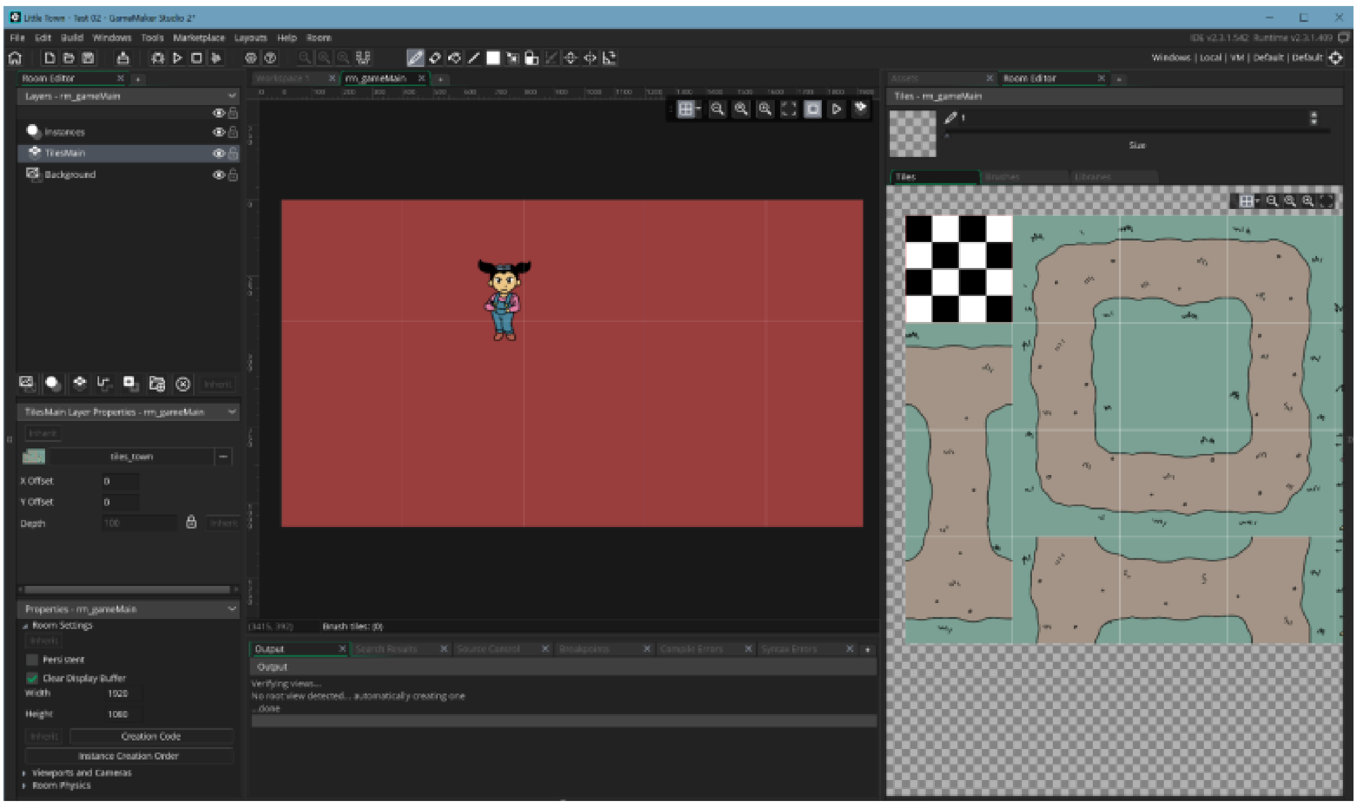
Open our one-and-only Room (rm_gameMain) and make sure you can see the Room Editor (the tab that shows us layers like Instances and Background. At the bottom of this tab are those layer buttons we mentioned; click on the Create New Tile Layer ( ) button.

This will make a new layer called Tiles_1. Right-click on the new layer and choose Rename; rename it to TilesMain.

Drag the layer to place it between Instances and Background.

Select TilesMain in the Layers section so we can edit it. Below, under Layer Properties, you'll see a menu that says, "No Tile Set." Click on this and choose the TilesTown Tile Set we just made.

You'll see a new tab open called Room Editor and it will show our new Tile Set. If it looks huge or you can't see it properly, you can use the little magnifying glass tools to zoom in and out.



TilesMain Layer Properties - rm_gameMain
tiles_town
X Offset 0
Y Offset 0
Depth 100

Properties - rm_gameMain
Room Settings
Persistent
Clear Display Buffer ✓
Width 1920
Height 1080
Creation Code
Instance Creation Order
Viewports and Cameras
Room Physics

```
Verifying views...
No root view detected... automatically creating one
...done
```

You might have noticed that our little Room is way too small to do much with, so let’s change that. In the Properties tab (by default on the lower left of the IDE), you’ll see a few sub-sections you can show and hide.

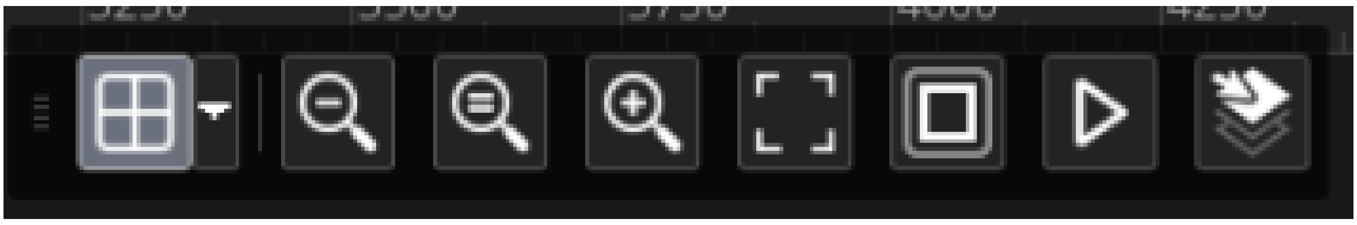
The first is Room Settings; if it’s hidden, click the little arrow to show its contents and change these values:

Width: 4000
Height: 2400

Now we’ve got some space to play in! In the main Room view, you can use the magnifying glass tools at the top to zoom in and out and you can hold the space bar while clicking and dragging with your mouse to pan around. Make sure you can see the whole Room.

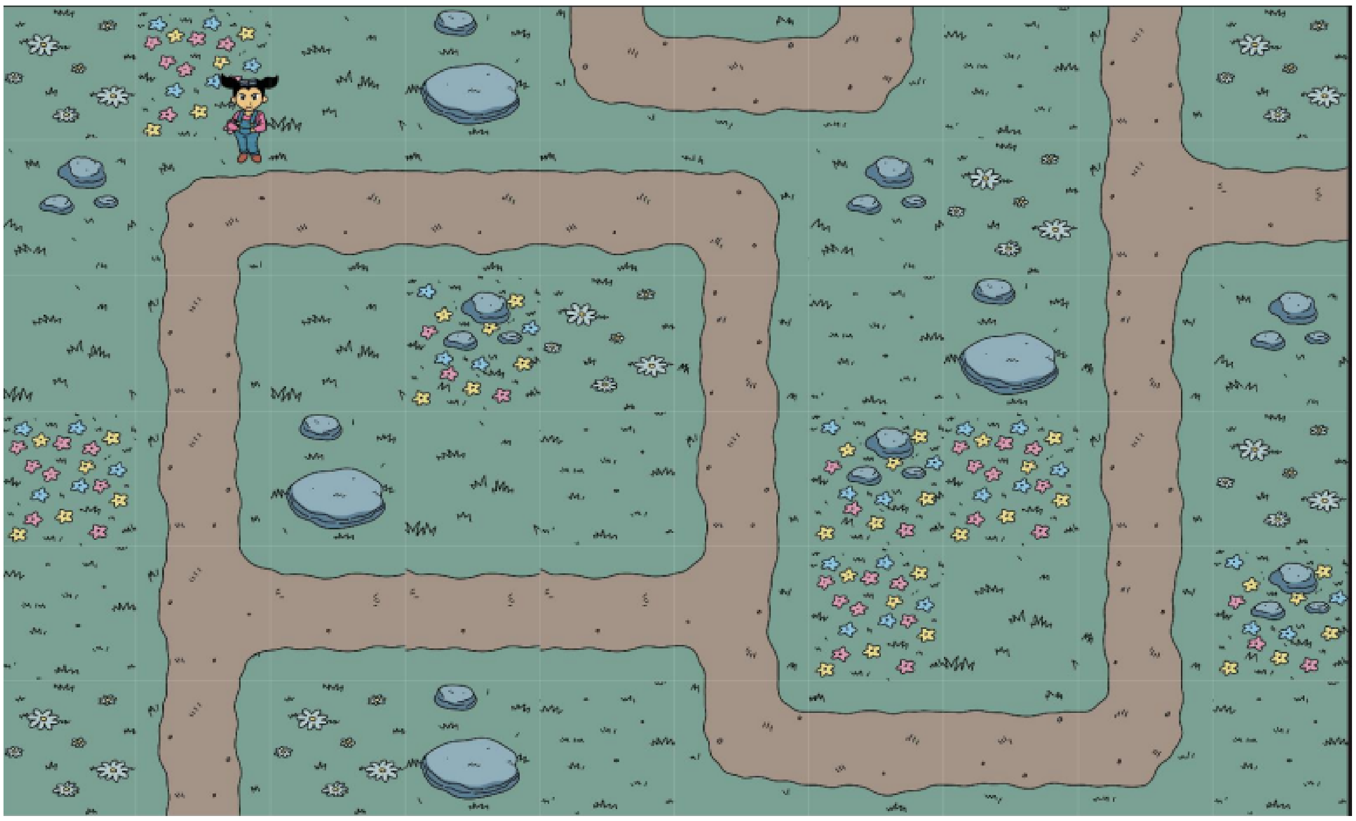
6.3 Placing Tiles

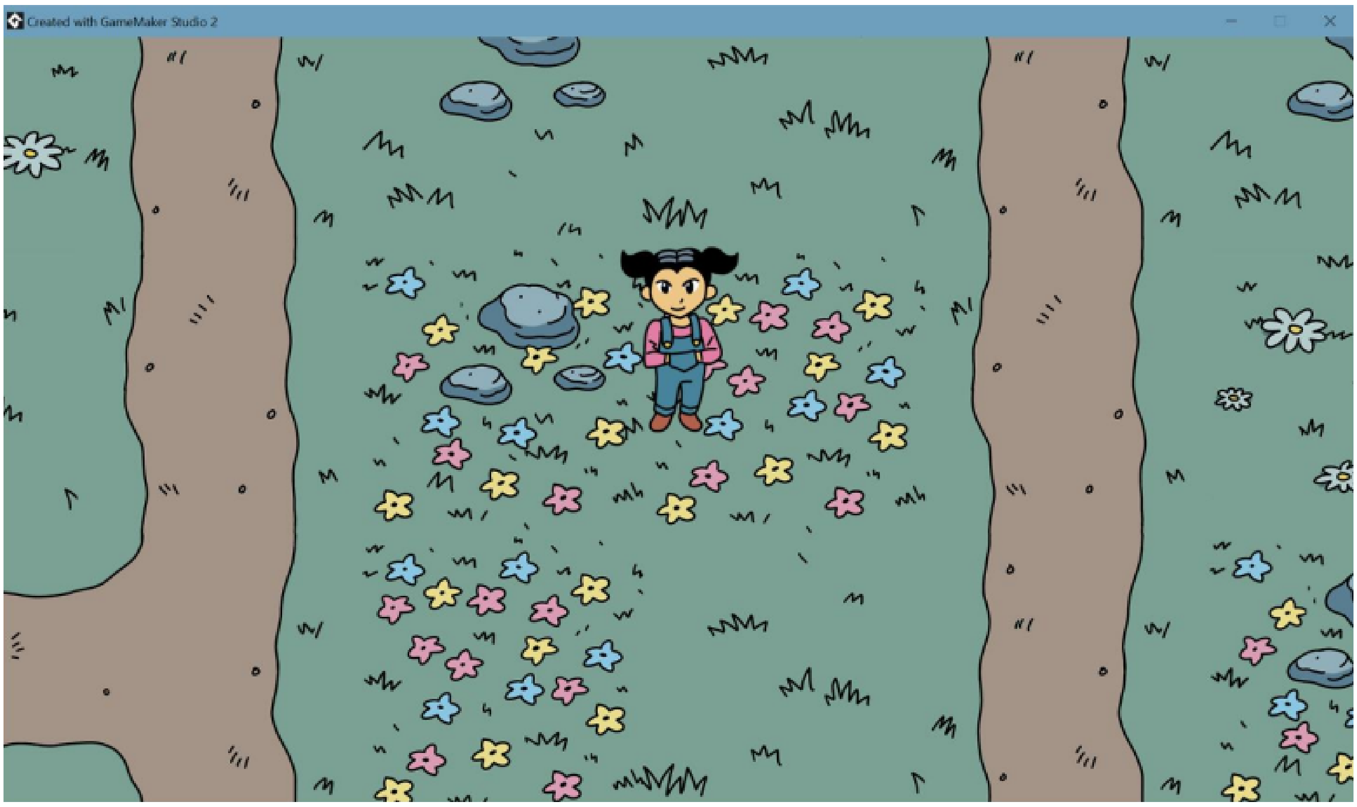
You’ll notice a grid in the Room that corresponds to the size of our Tile Set’s Tile Width and Tile Height (in this case, 400). If you click on the Grid icon at the top right on of the Room tab, you can turn the grid on and off and check the grid size.



For now, start placing Tiles in your Room! You can “stamp” them one by one by clicking on a tile in the Room Editor tab, or you can click and drag on the Tile Set to select and stamp a series of Tiles all at once.

Design the base for your town anyway you like. Fill up every tile space in the Room.




Created with GameMaker Studio 2

6.4 Adding Objects To The Town

With our town partially laid out, let's create an Object using some of the other assets we imported.

Close the game window and return to GameMaker Studio 2.

Go back to your Workspace and in the Asset Browser, find the group of town background Sprites we imported. Double-click on spr_barrel01 to open it in the Sprite Editor.

You can actually place just plain old Sprites on a layer in a Room but doing so means our player Object won't be able to interact with them.

So, we're going to make each of these environmental details into Objects.

First, let's take that origin point on our barrel Sprite and move it to the bottom center of the Sprite, so it lines up with the middle of the base of the barrel.

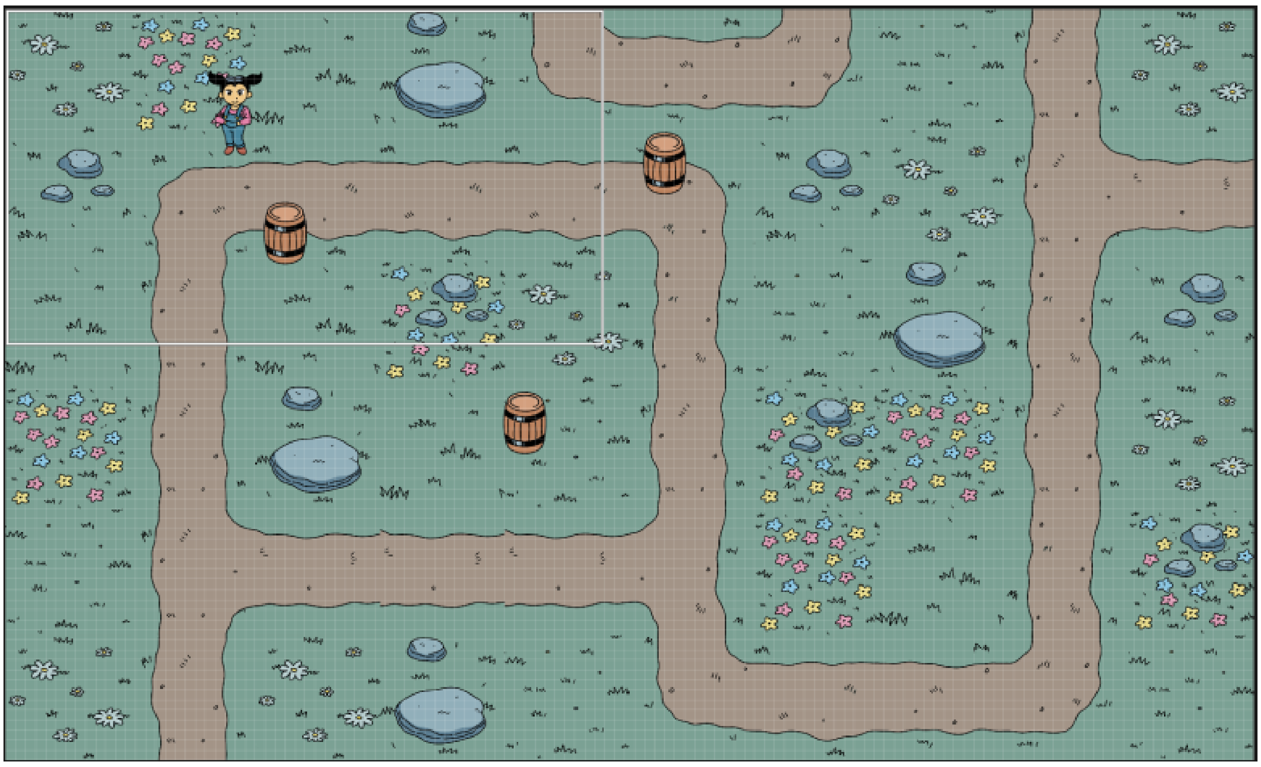
Now, in the Asset Browser, right-click and choose Create > Object. Name this new Object obj_barrel01. Click on the menu that says, "No Sprite," and choose spr_barrel01.

This is all good for now. Open our big room (rm_gameMain) again and click on the Instances layer to make sure it's selected.

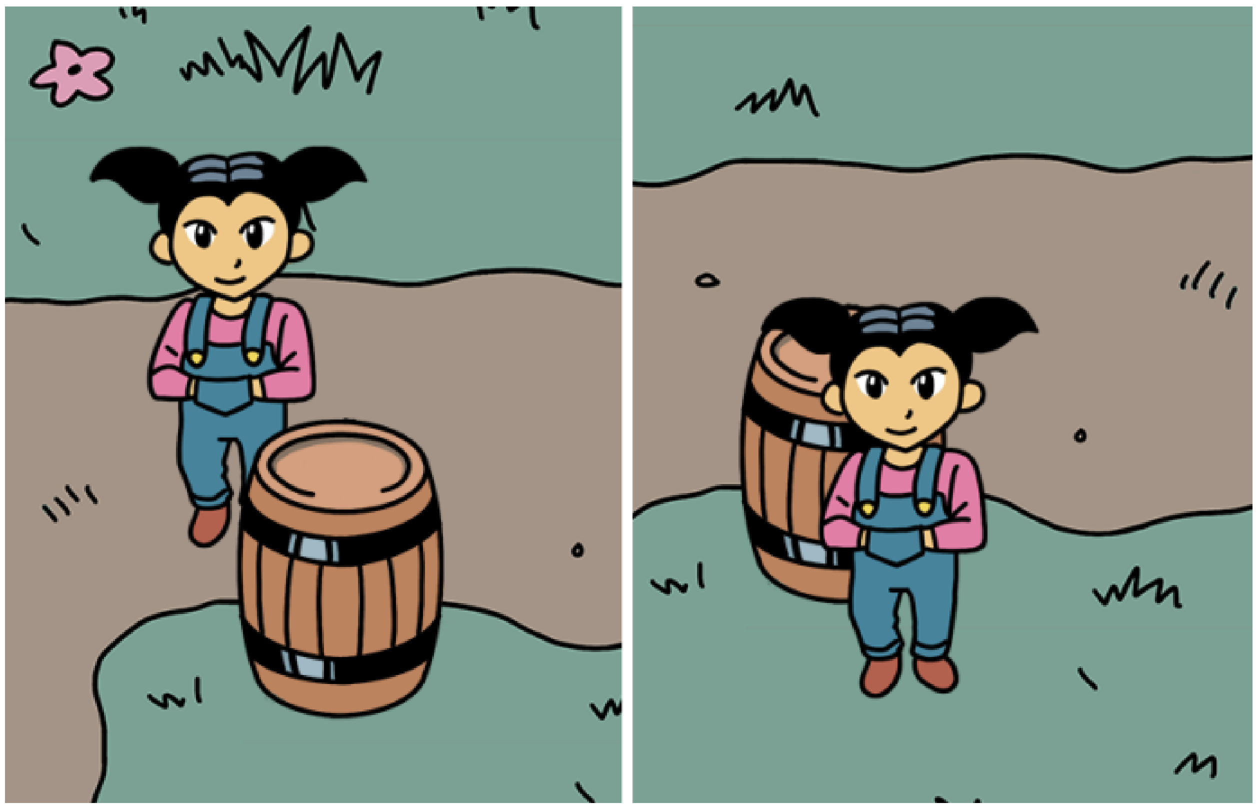
In the Asset Browser, find our new obj_barrel01 and drag it into the Room. (If you can't see your Asset Browser, look for the Assets tab; it might be paired with something else, or the Room Editor tab might still be in focus..)

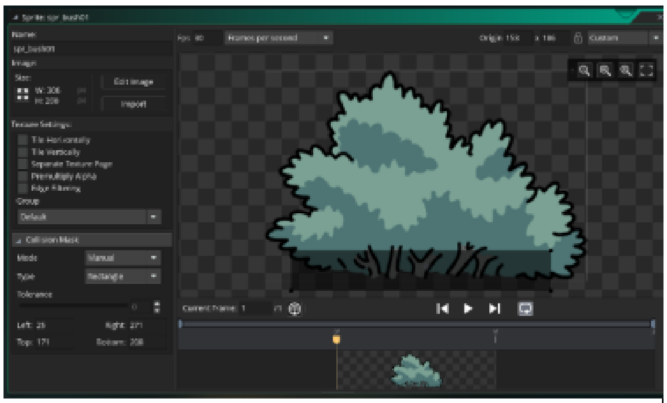
Dragging an Object into a Room creates an instance of that Object, so you can place as many as you like. If you update an Object, all its instances are automatically updated too.

Go ahead and drag one or two more barrels into the Room and position them wherever you like



When you're happy with your barrel placement, run the game again and walk around.You should have observed that the barrels appear "flat" — we don't seem to move in front or behind them correctly and 2e can just walk through the barrels




Sprite: spr_bush01

Name: spr_bush01

Size: W: 308 H: 208

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

Group: Default

Collision Mask

Mode: Manual

Type: Rectangle

Tolerance: 0

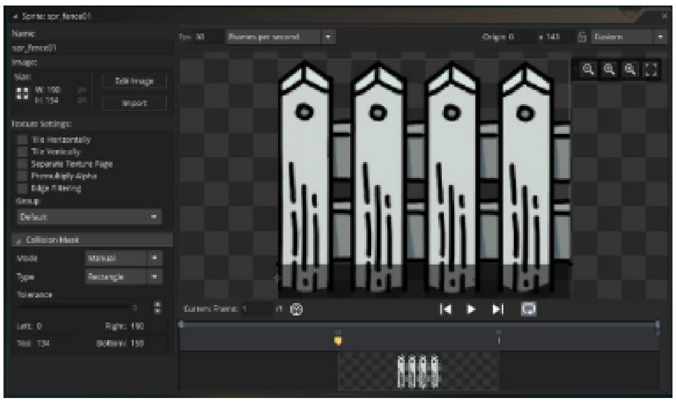
Left: 25 Right: 271

Top: 171 Bottom: 208

Fps: 30 Frames per second

Origin 153 x 186 Custom

Current Frame: 1 /1


Sprite: spr_fence01

Name: spr_fence01

Size: W: 190 H: 154

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

Group: Default

Collision Mask

Mode: Manual

Type: Rectangle

Tolerance: 0

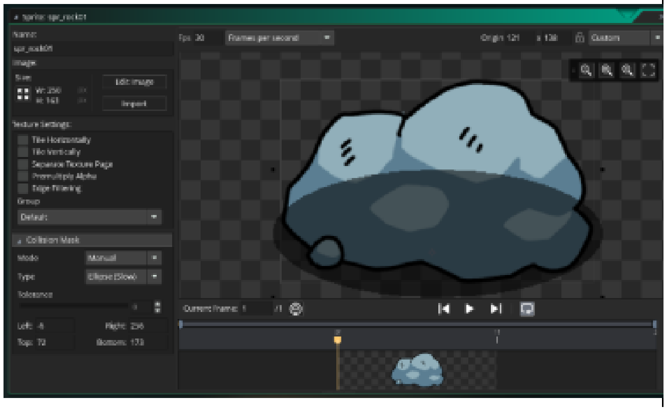
Left: 0 Right: 190

Top: 134 Bottom: 159

Fps: 30 Frames per second

Origin 0 x 143 Custom

Current Frame: 1 /1


Sprite: spr_rock01

Name: spr_rock01

Size: W: 250 H: 163

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

Group: Default

Collision Mask

Mode: Manual

Type: Ellipse (Slow)

Tolerance: 0

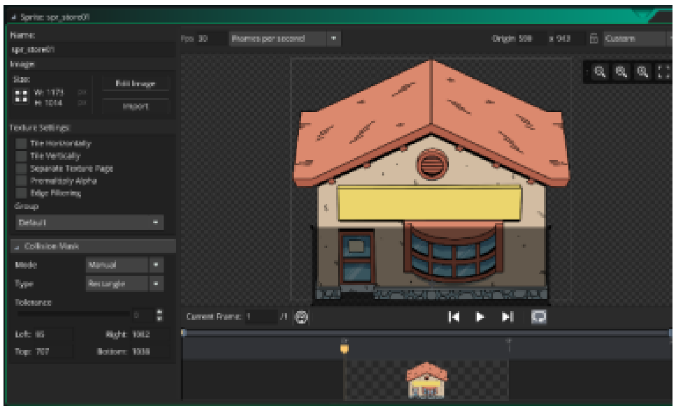
Left: 4 Right: 256

Top: 72 Bottom: 173

Fps: 30 Frames per second

Origin 121 x 138 Custom

Current Frame: 1 /1


Sprite: spr_store01

Name: spr_store01

Size: W: 1173 H: 1014

Texture Settings: Tile Horizontally, Tile Vertically, Separate Texture Page, Premultiply Alpha, Edge Filtering

Group: Default

Collision Mask

Mode: Manual

Type: Rectangle

Tolerance: 0

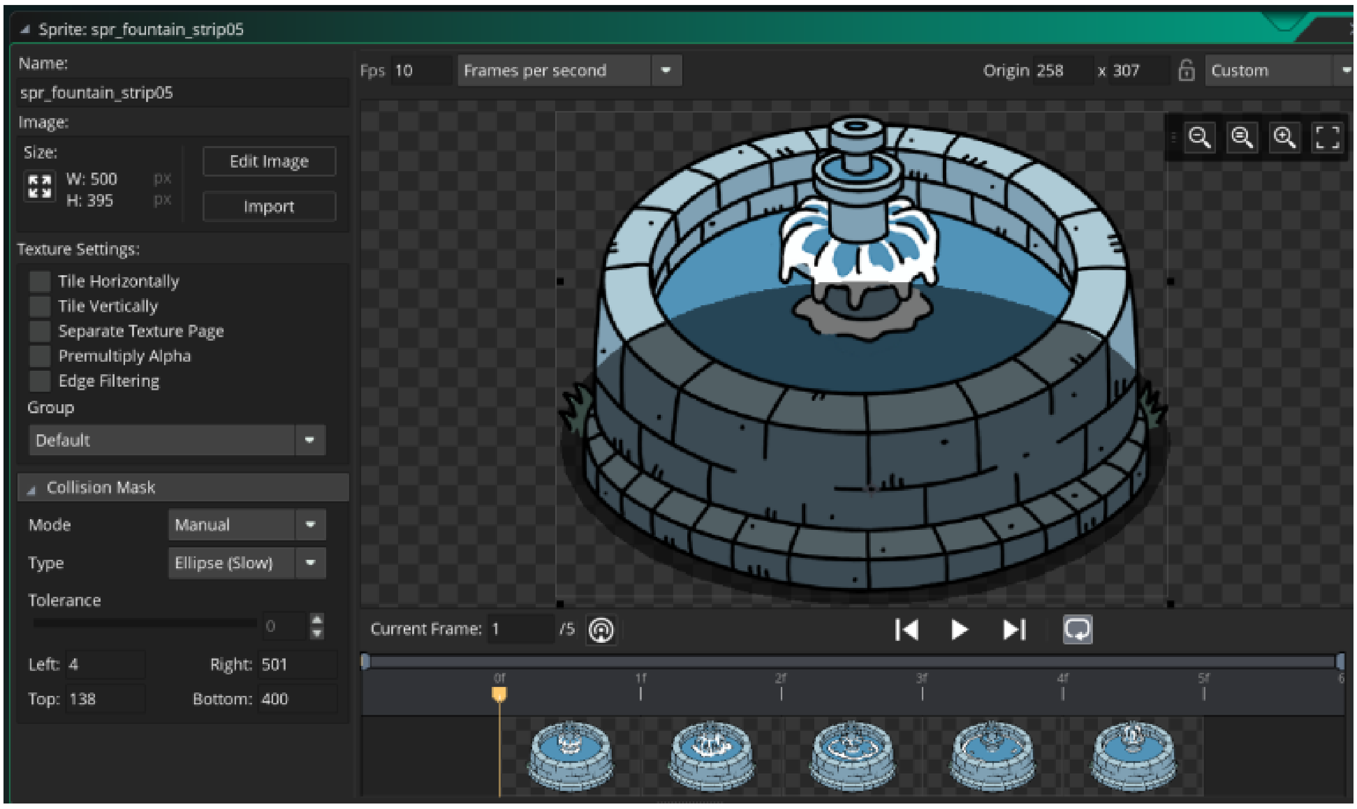
Left: 85 Right: 1082

Top: 707 Bottom: 1038

Fps: 30 Frames per second

Origin 599 x 913 Custom

Current Frame: 1 /1


Sprite: spr_fountain_strip05

Name:

spr_fountain_strip05

Image:

Size:

W: 500 px

H: 395 px

Edit Image

Import

Texture Settings:

Tile Horizontally

Tile Vertically

Separate Texture Page

Premultiply Alpha

Edge Filtering

Group

Default

Collision Mask

Mode: Manual

Type: Ellipse (Slow)

Tolerance: 0

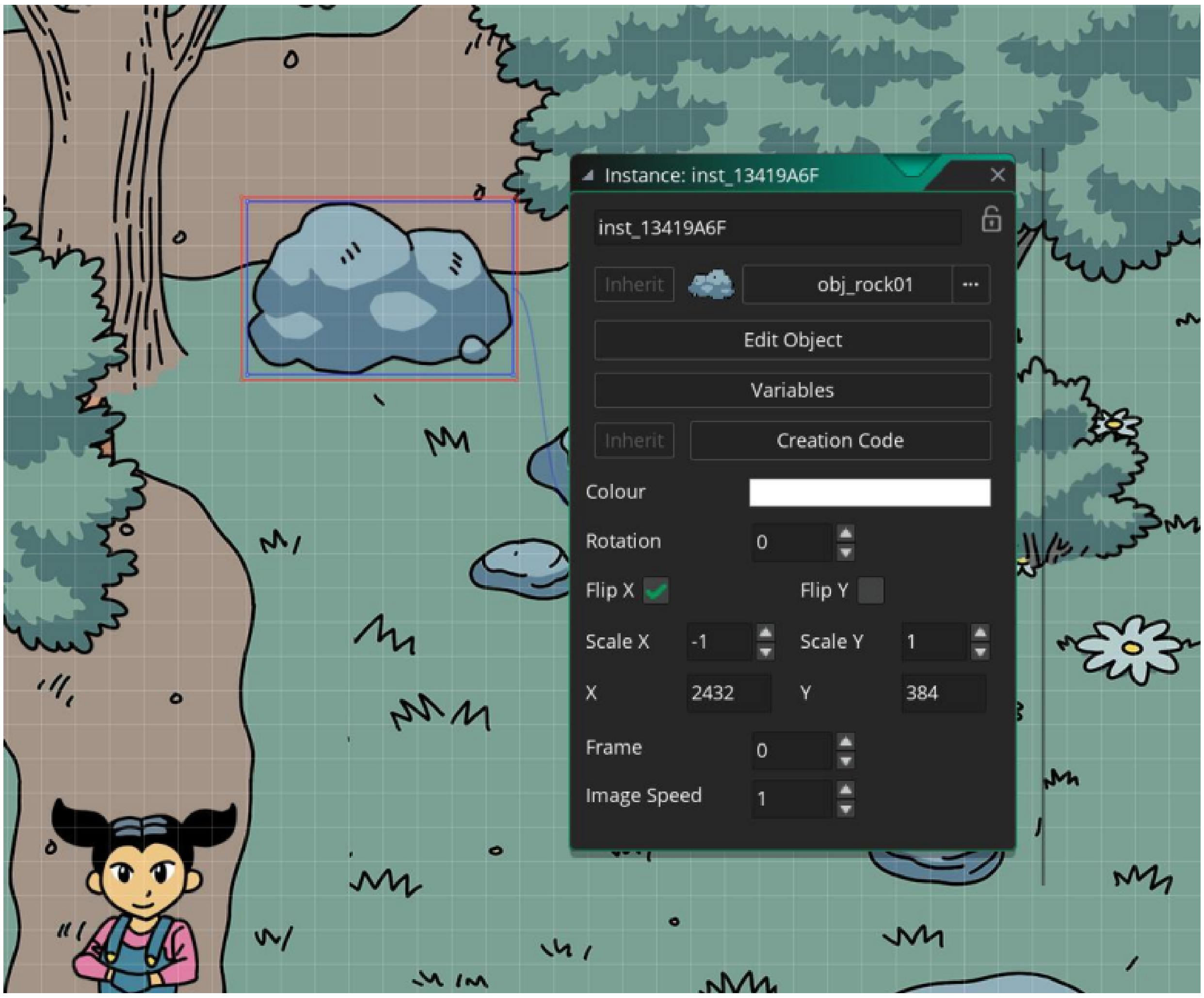
Left: 4 Right: 501

Top: 138 Bottom: 400

Fps 10 Frames per second

Origin 258 x 307 Custom

Current Frame: 1 /5


Instance: inst_13419A6F

inst_13419A6F

Inherit | obj_rock01

Edit Object

Variables

Inherit | Creation Code

Colour

Rotation 0

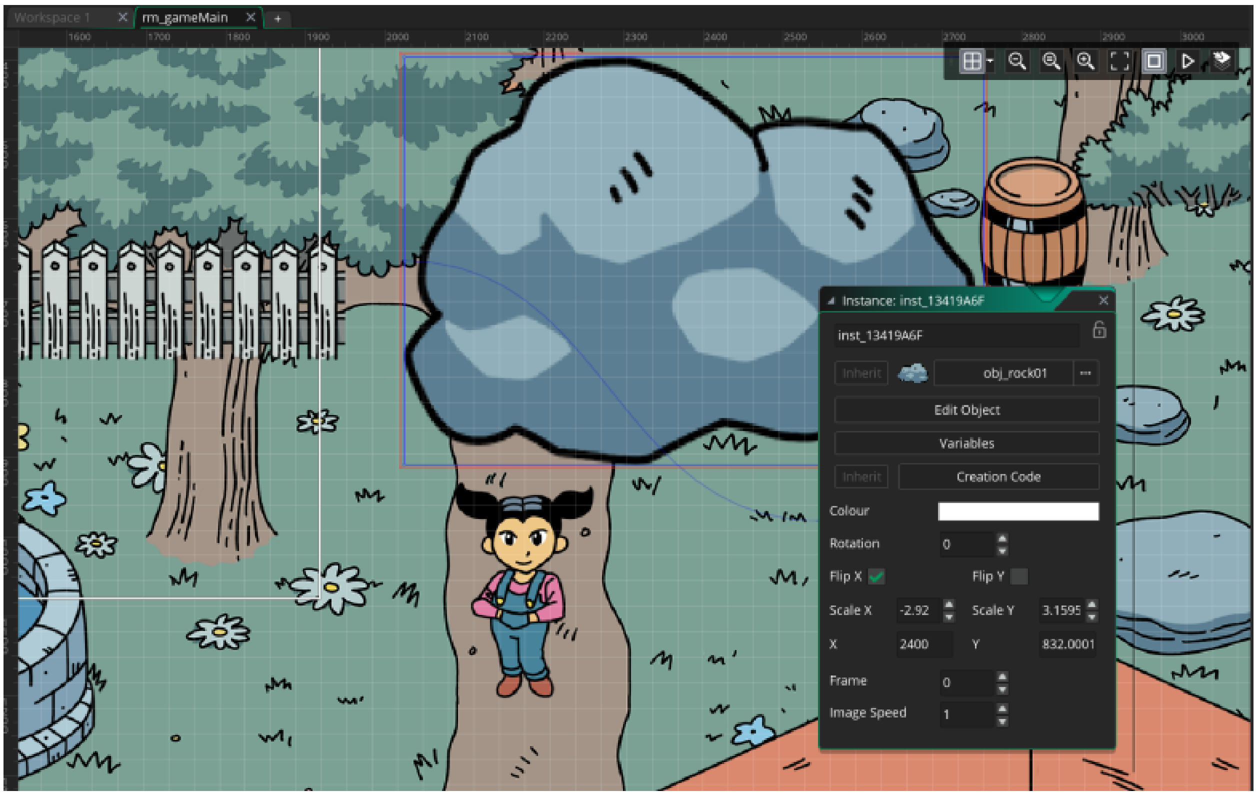
Flip X ✓ Flip Y

Scale X -1 Scale Y 1

X 2432 Y 384

Frame 0

Image Speed 1


Workspace 1 | rm_gameMain

Instance: inst_13419A6F

inst_13419A6F

Inherit | obj_rock01

Edit Object

Variables

Inherit | Creation Code

Colour

Rotation 0

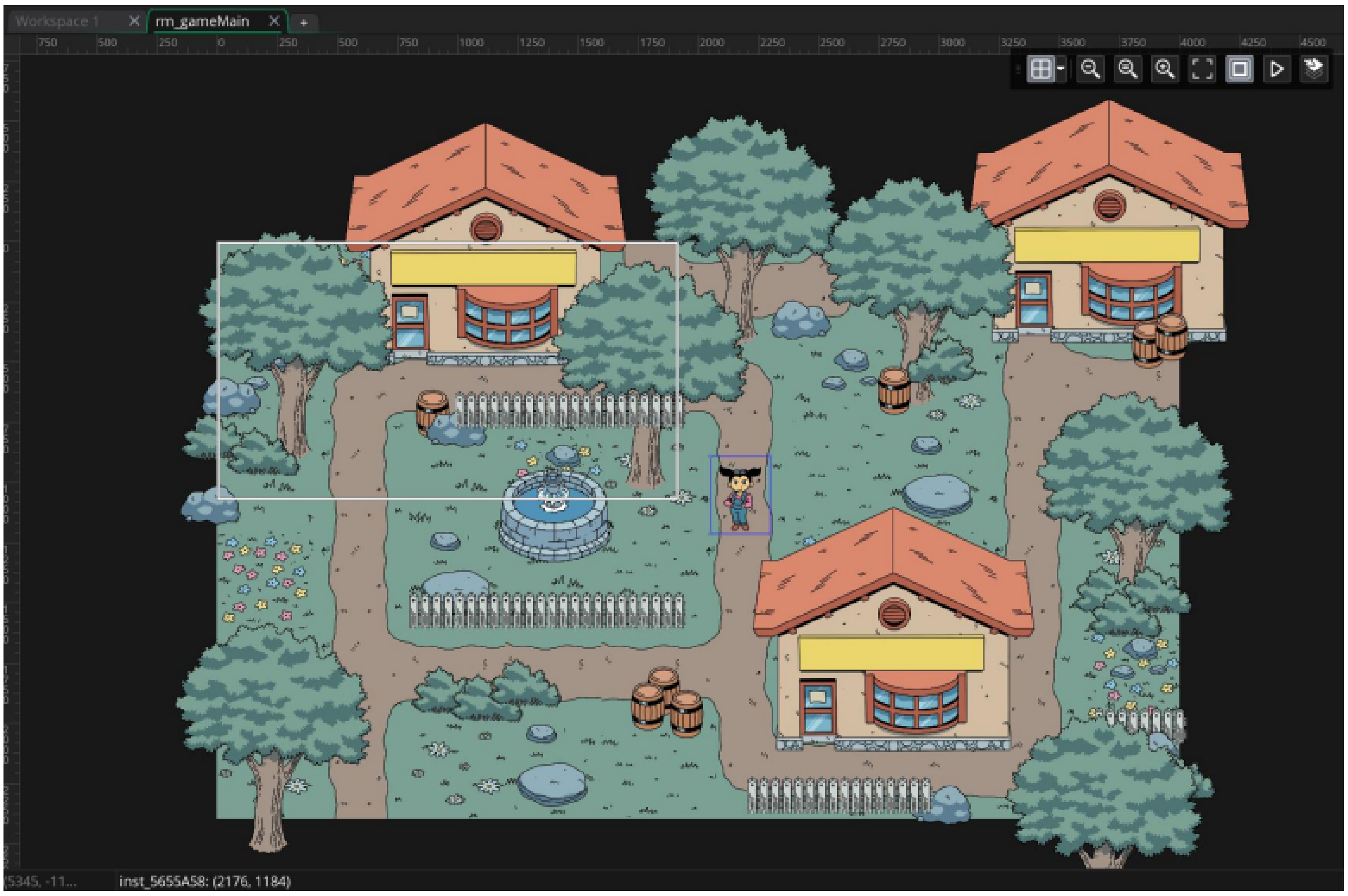
Flip X ✓ Flip Y

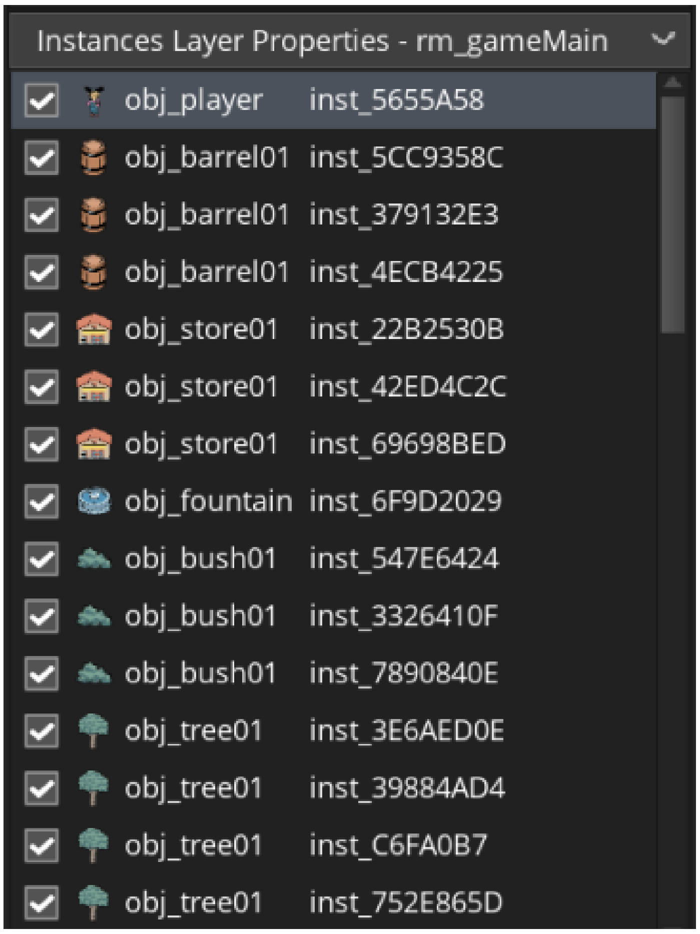
Scale X -2.92 Scale Y 3.1595

X 2400 Y 832.0001

Frame 0

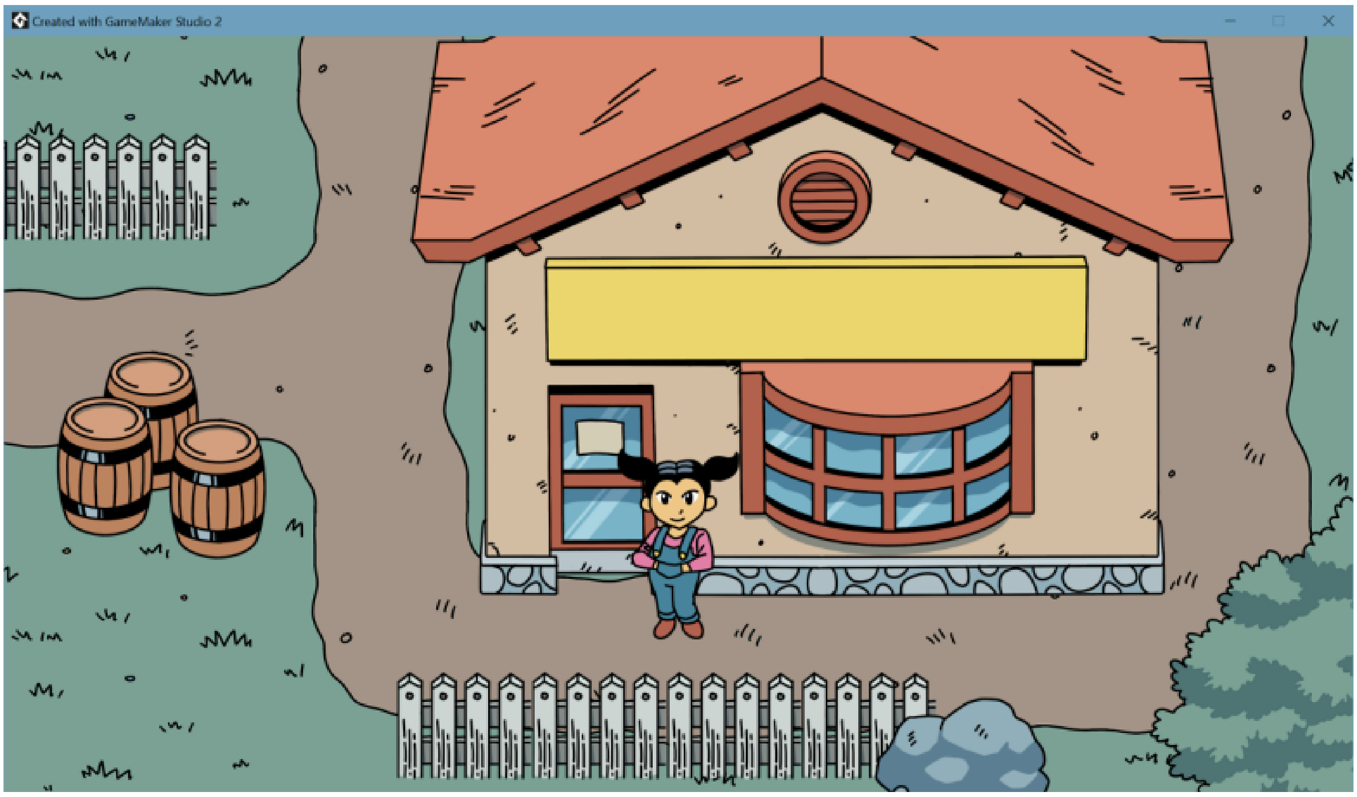
Image Speed 1

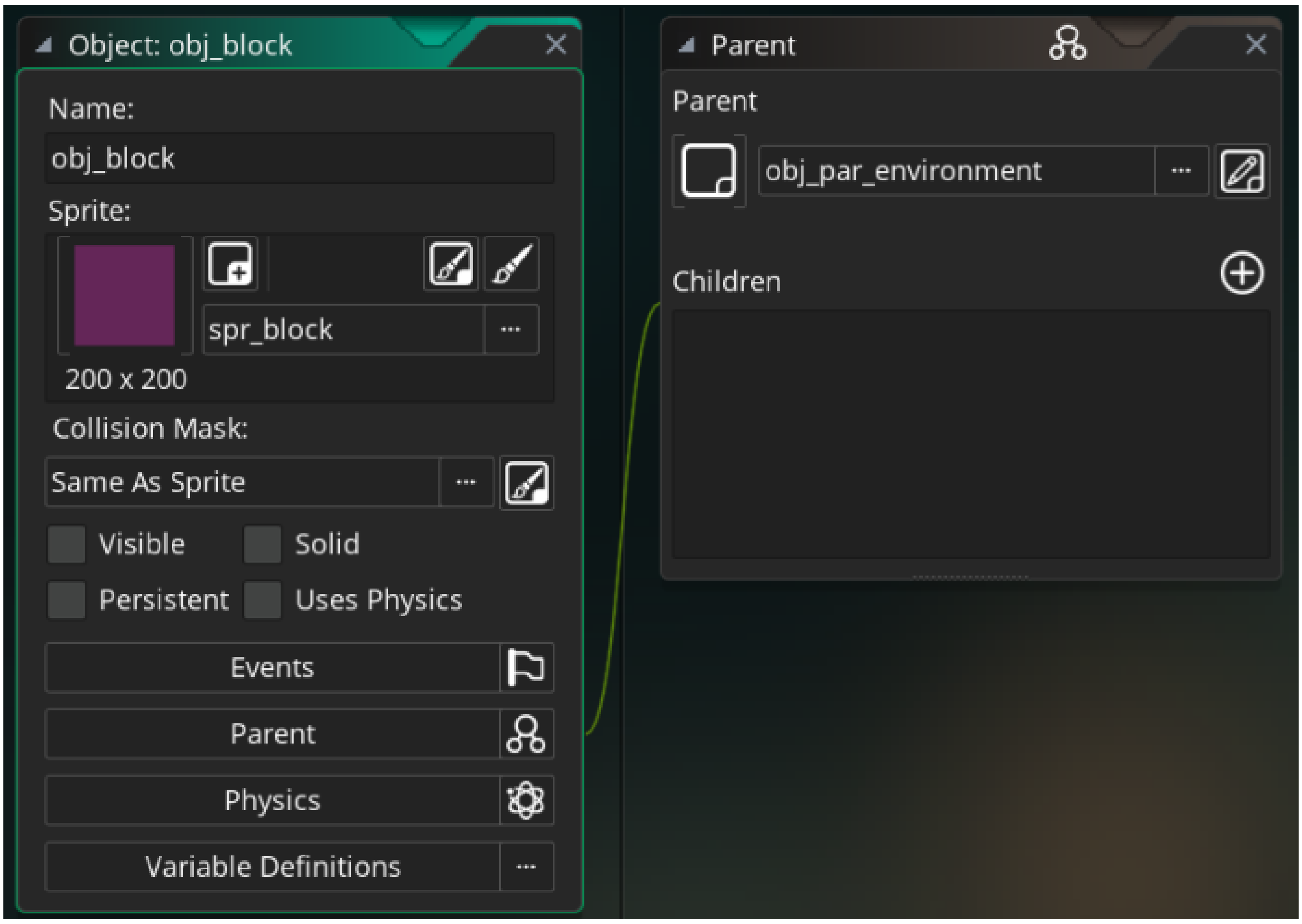




Instances Layer Properties - rm_gameMain

✓	obj_player	inst_5655A58
✓	obj_barrel01	inst_5CC9358C
✓	obj_barrel01	inst_379132E3
✓	obj_barrel01	inst_4ECB4225
✓	obj_store01	inst_22B2530B
✓	obj_store01	inst_42ED4C2C
✓	obj_store01	inst_69698BED
✓	obj_fountain	inst_6F9D2029
✓	obj_bush01	inst_547E6424
✓	obj_bush01	inst_3326410F
✓	obj_bush01	inst_7890840E
✓	obj_tree01	inst_3E6AED0E
✓	obj_tree01	inst_39884AD4
✓	obj_tree01	inst_C6FA0B7
✓	obj_tree01	inst_752E865D




Object: obj_block

Name: obj_block

Sprite: spr_block

200 x 200

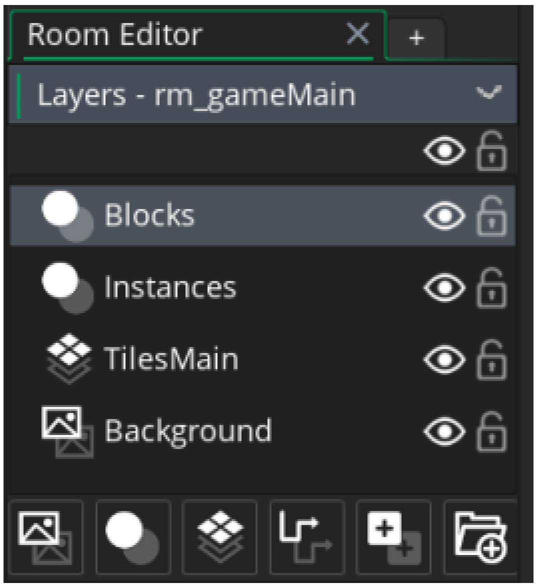
Collision Mask: Same As Sprite

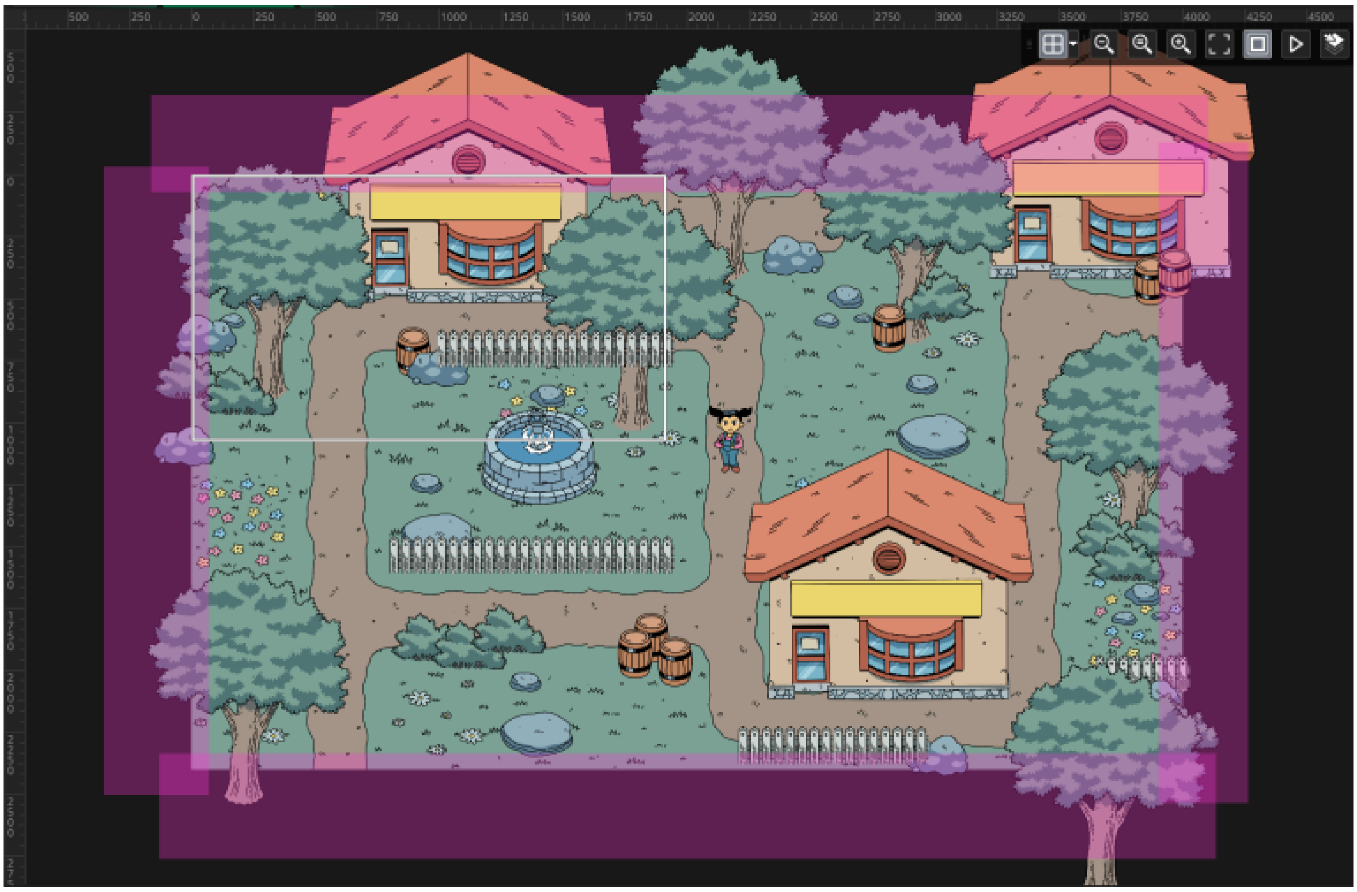
Visible
Solid
Persistent
Uses Physics

Parent

Parent: obj_par_environment

Children


Room Editor
+
Layers - rm_gameMain
Blocks
Instances
TilesMain
Background



7. NPCs

7.1 The Baker NPC

With our town designed, we can now turn our attention toward the other characters who will populate it.

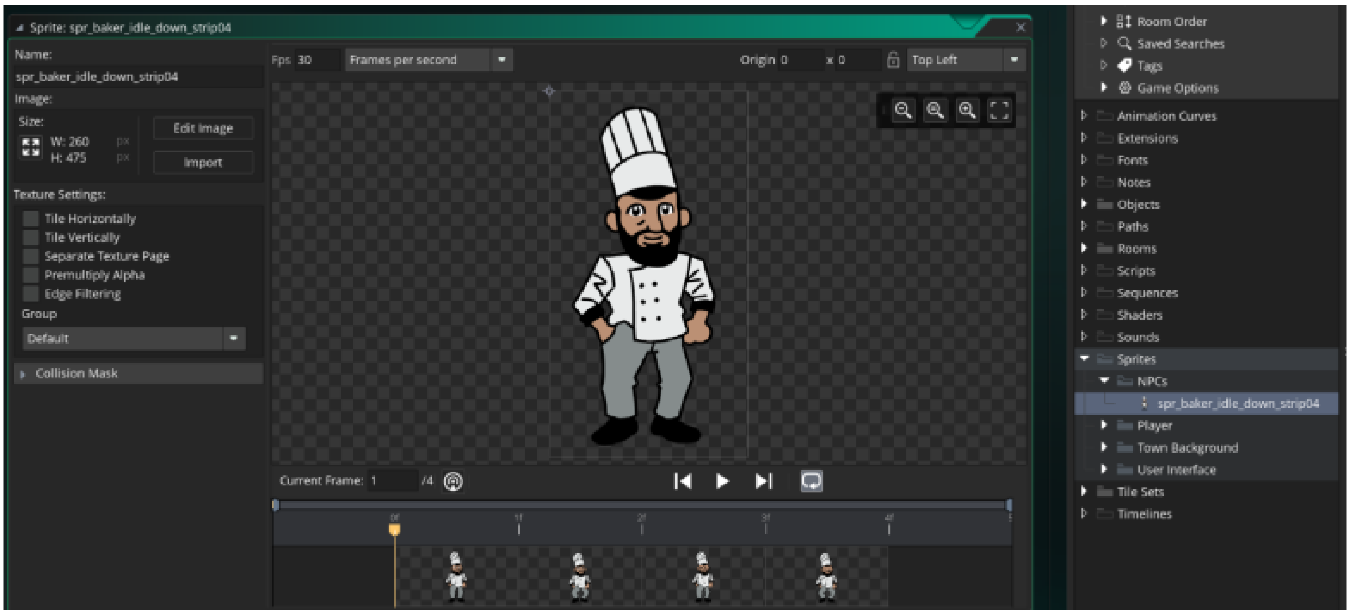
These characters are known as NPCs — non-playable characters — and our game will have three of them: the Baker, Teacher and Grocer.

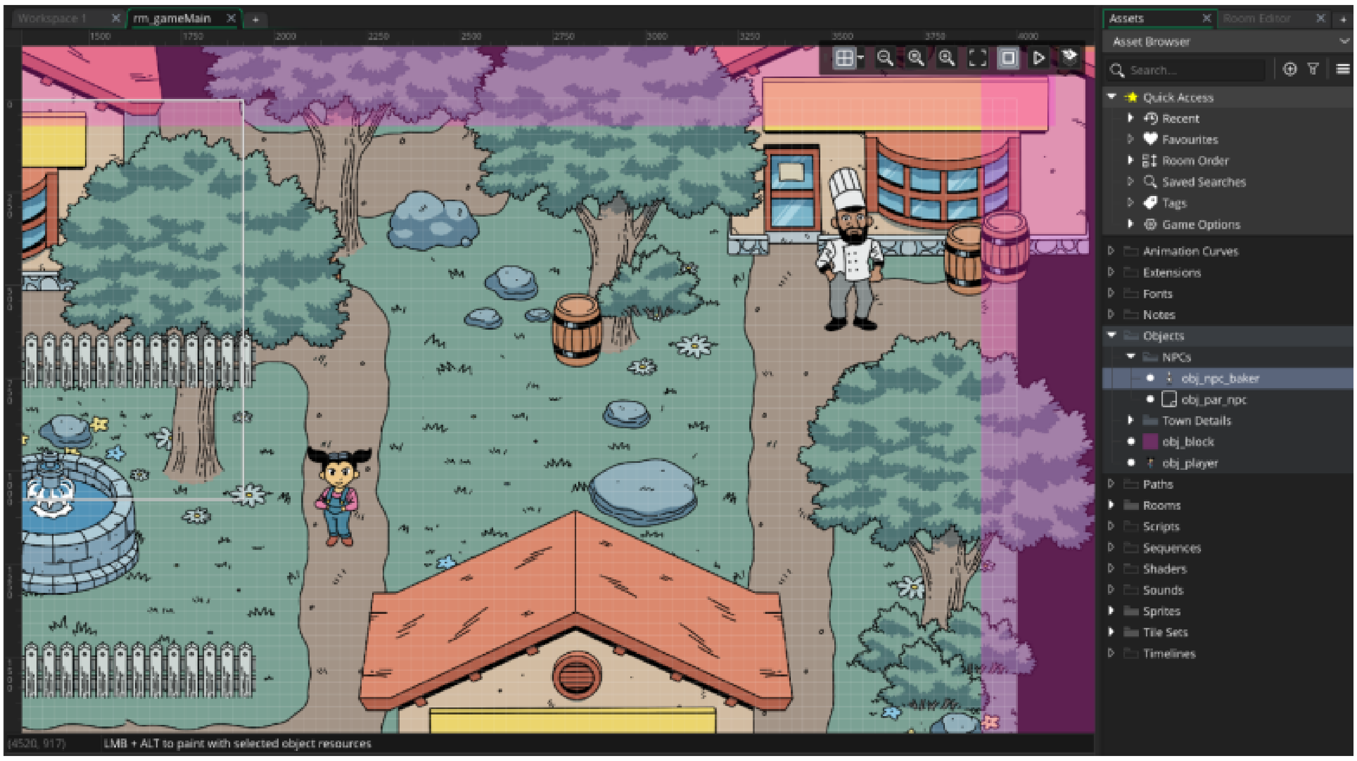
Let's start with the Baker. In the Asset Browser, create a new Group and name it NPCs. Drag this new Group into the Sprites group.

Next, using File Explorer (Windows) or Finder (Mac), navigate to the Assets folder provided with this course and open the Sprites > Characters and Items folder.

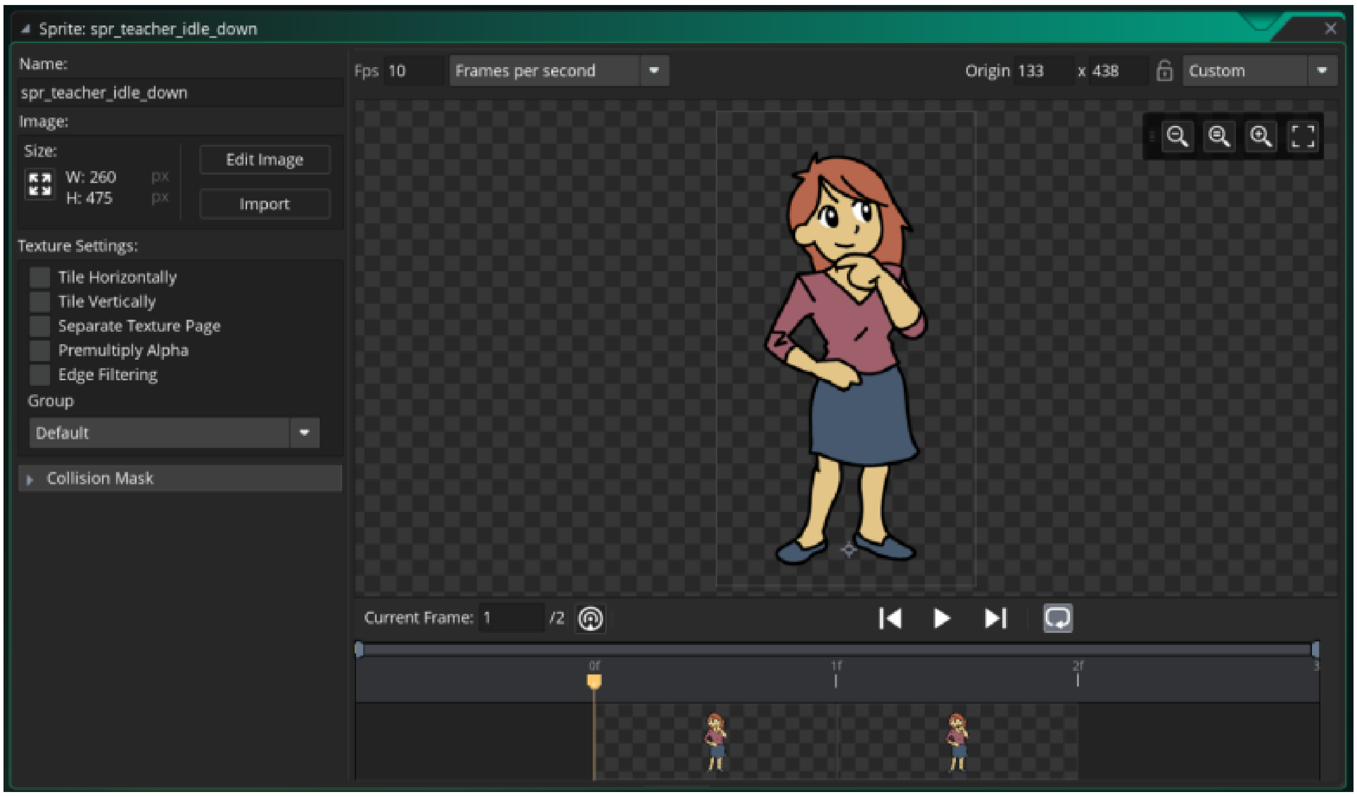
Drag the following Sprite into the new NPCs Group you made in the Asset Browser:

- `spr_baker_idle_down_strip04`





7.2 The Teacher NPC


Sprite: spr_teacher_idle_down

Name: spr_teacher_idle_down

Image:

Size: W: 260 px H: 475 px

Texture Settings:

- Tile Horizontally
- Tile Vertically
- Separate Texture Page
- Premultiply Alpha
- Edge Filtering

Group: Default

Fps 10 Frames per second

Origin 133 x 438 Custom

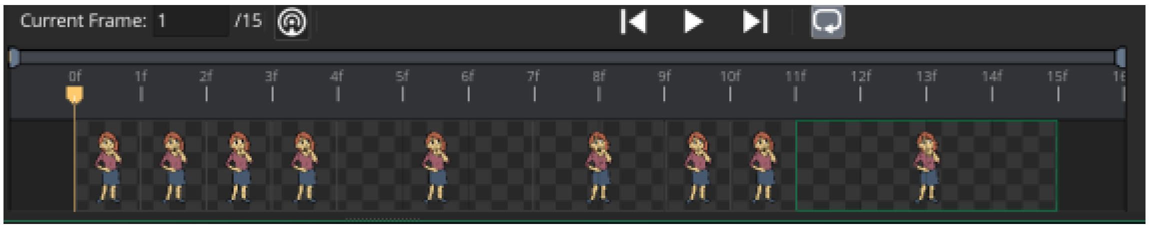
Current Frame: 1 /2

Right-click in the Asset Browser and create a new Object. Name it obj_npc_teacher and attach the spr_teacher_idle_down Sprite to it. In the Object Editor, click Parent and choose obj_par_npc to make the Teacher a child Object.

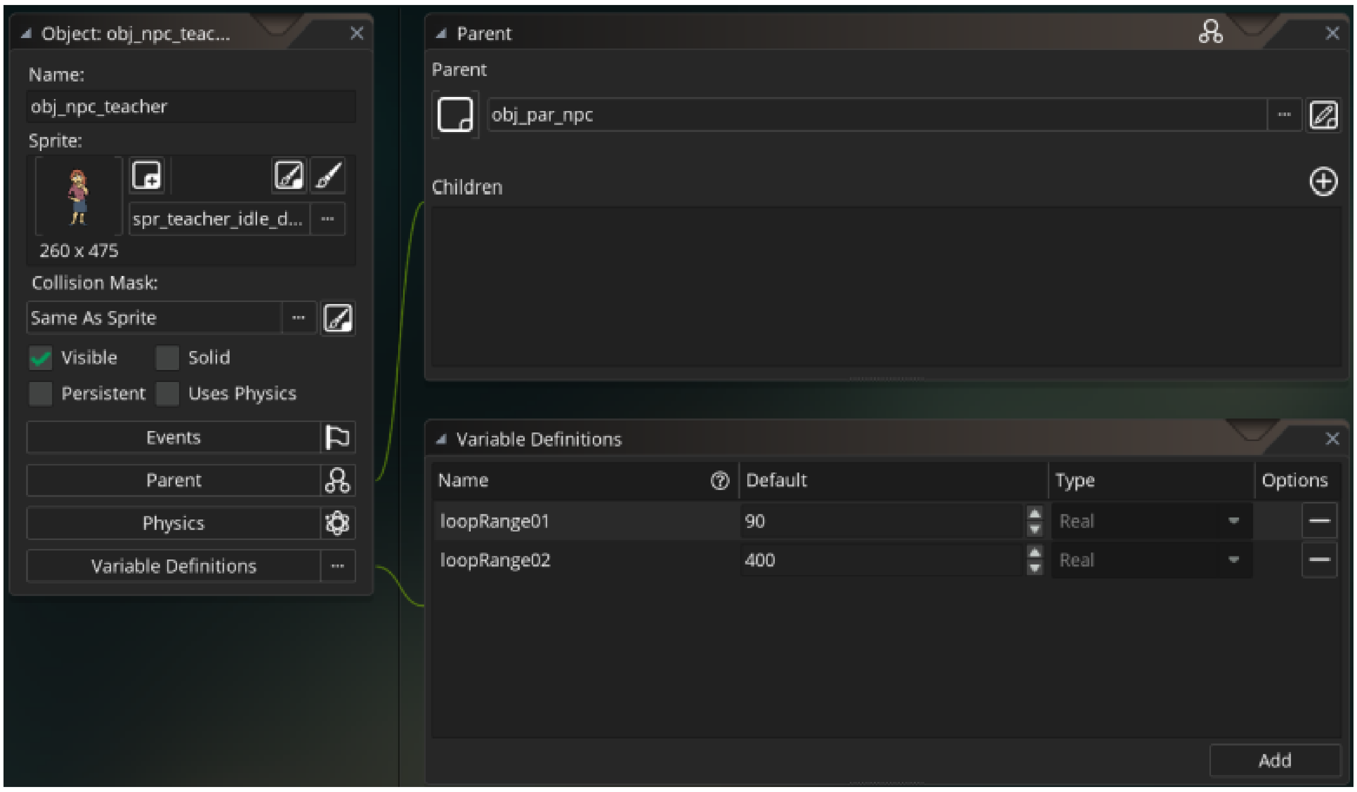
For our Teacher Object, let's do two things:

1. Edit its idle Sprite the same way we did with the Baker.
2. Change the values of its Variable Definitions.

First, open spr_teacher_idle_down again and edit the animation as we did with the Baker. Right now, the Sprite is just two frames of animation. Play around here however you like to add a different cadence to the Teacher's animation, and make sure to stretch the final frame out so there's a pause.



Next, open obj_npc_teacher and click Variable Definitions. Change the values for loopRange01 and loopRange02 to something other than the defaults.



Object: obj_npc_teac...

Name: obj_npc_teacher

Sprite: spr_teacher_idle_d...

260 x 475

Collision Mask: Same As Sprite

✓ Visible ☐ Solid

☐ Persistent ☐ Uses Physics

Parent

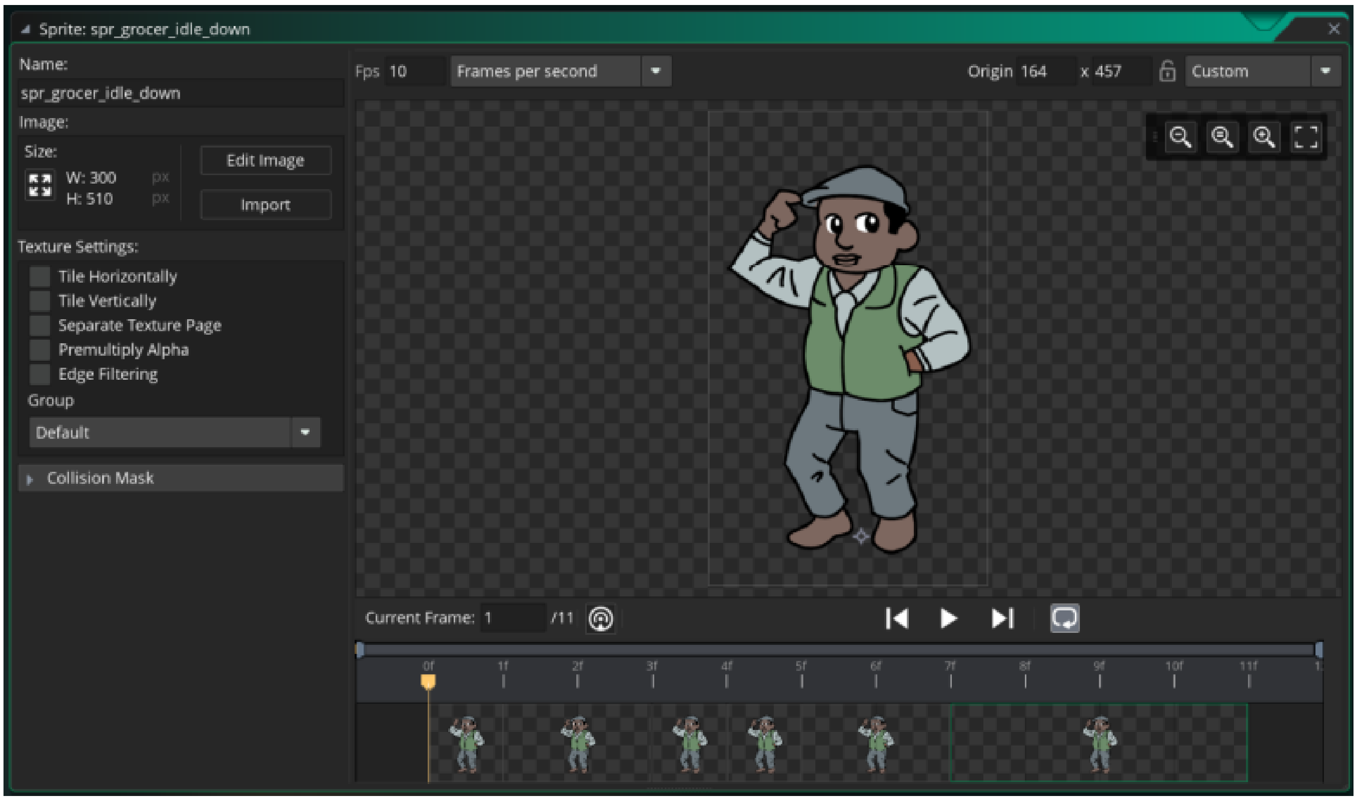
Parent: obj_par_npc

Children

Variable Definitions

Name	Default	Type	Options
loopRange01	90	Real	
loopRange02	400	Real	

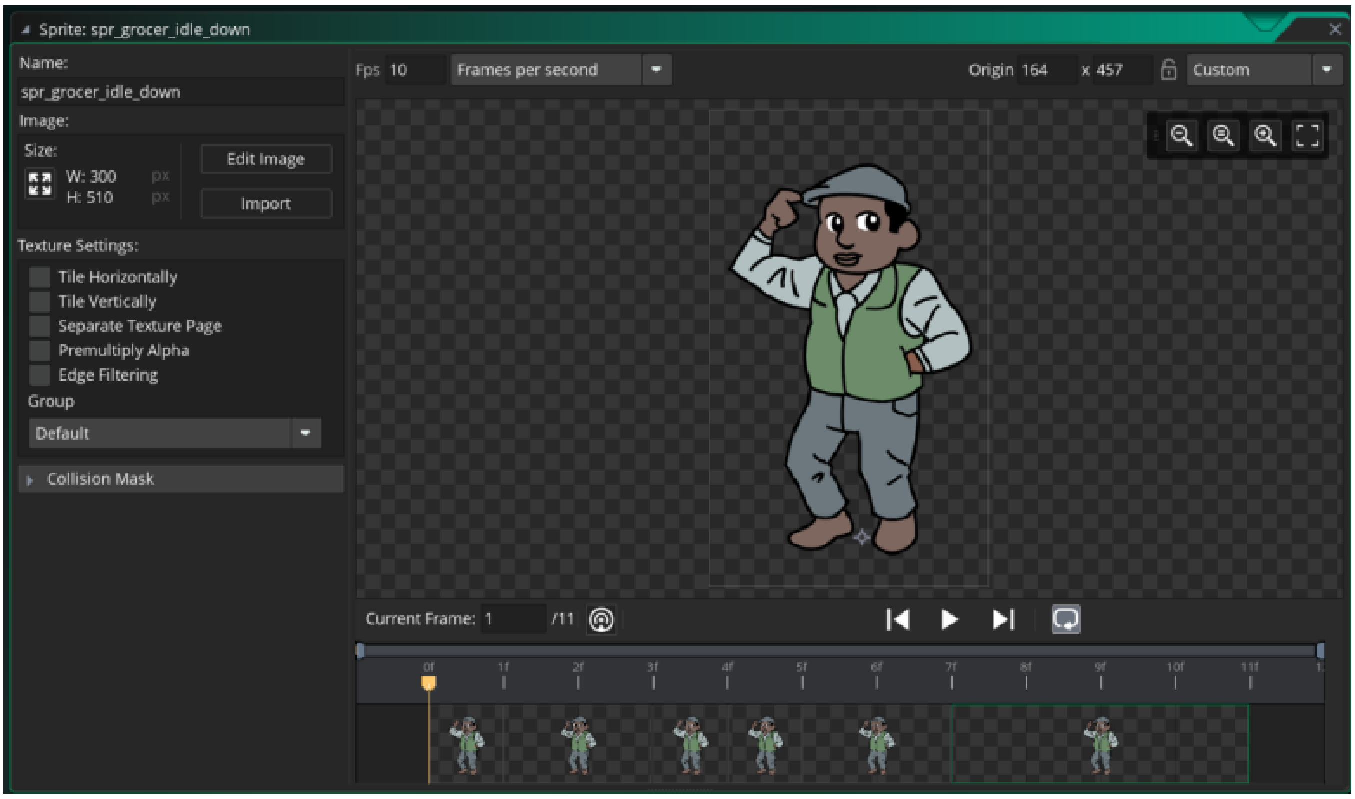
7.3 The Grocer NPC



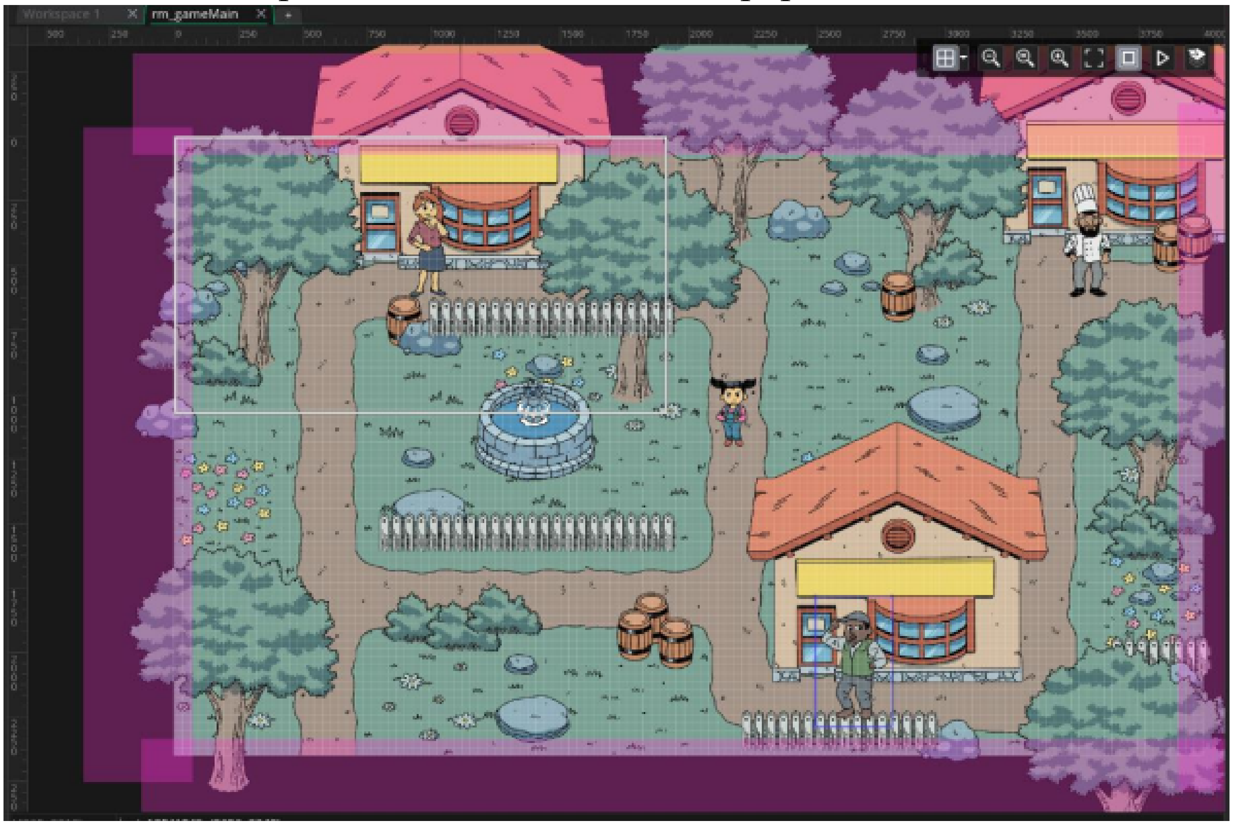
Create another Object and name it obj_npc_grocer. Attach the spr_grocer_idle_down Sprite to it. Just like above, make the Grocer Object a child of obj_par_npc.

In the Object Editor, click Variable Definitions and change the defaults here to be different values (they don't have to be the same as the values you chose for the Teacher Object).

Next, open spr_grocer_idle_down again and repeat the steps we took above for the Teacher Sprite to enhance the Grocer's idle animation; again, make sure the last frame of animation has a pause.



Once you have both the Teacher and Grocer ready, Go to rm_gameMain again. Make sure you have the Instances layer selected and drag a single Instance each of obj_npc_teacher and obj_npc_grocer into the Room. Position the two NPCs in front of the two unoccupied stores; now our town is populated!

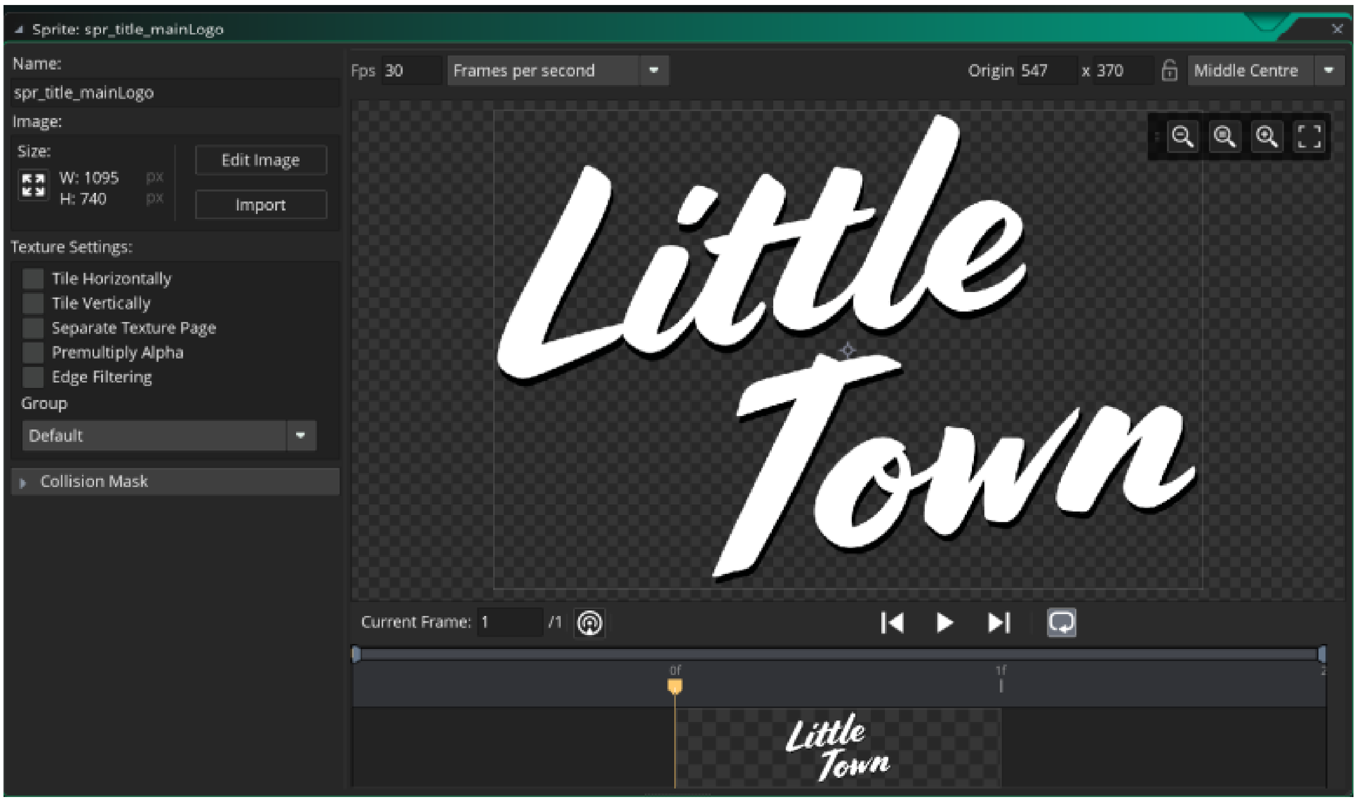


8. Creating The Title Screen

Using File Explorer (Windows) or Finder (Mac), navigate to the Assets folder provided with this course. Go to Sprites > User Interface and drag the files there into the Asset Browser:

- `spr_bg_mainTitle`
- `spr_title_mainLogo`
- `spr_title_pressEnter`

Open both spr_title_mainLogo and spr_title_pressEnter Sprites. Set the Origin for both to Middle Centre.


Sprite: spr_title_mainLogo

Name: spr_title_mainLogo

Image:

Size: W: 1095 px H: 740 px

Texture Settings:
- Tile Horizontally
- Tile Vertically
- Separate Texture Page
- Premultiply Alpha
- Edge Filtering

Group: Default

Fps 30 Frames per second

Origin 547 x 370 Middle Centre

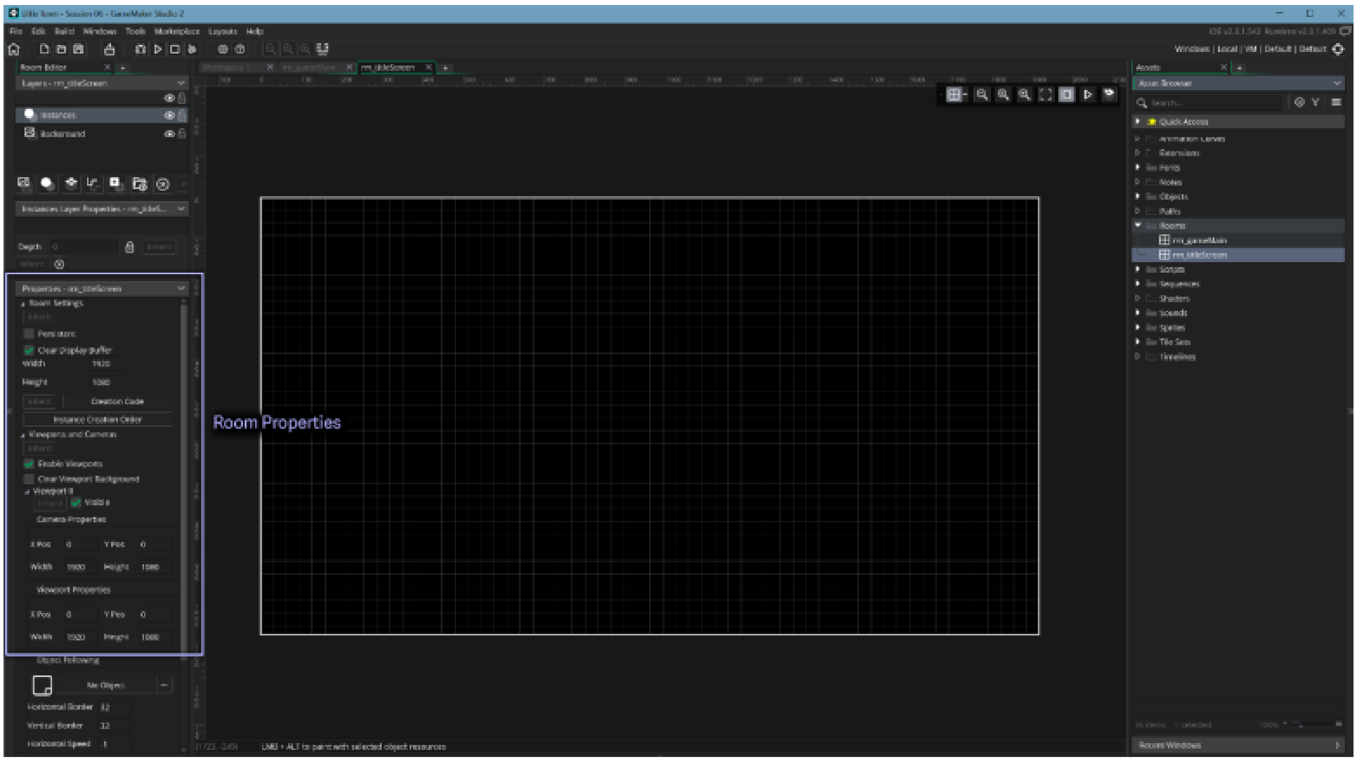
Current Frame: 1 /1

Next, we need to create a new Room to use as a title screen.

In the Asset Browser, right-click on the Rooms group and choose Create > Room. Name this new Room rm_mainTitle.

Open rm_mainTitle if it's not open already; using the Room Editor panel, change the following properties:

- Set the Width to 1920 and the height to 1080
- Enable Viewports
- Make `Viewport 0` visible
- Set the Camera width to 1920 and its height to 1080
- Set the Viewport width to 1920 and its height to 1080



Properties - rm_titleScreen
Room Settings
Persistent
Clear Display Buffer ✓
Width 1920
Height 1080
Viewports and Cameras
Enable Viewports ✓
Clear Viewport Background
Viewport 0
Visible ✓
Camera Properties
X Pos 0 Y Pos 0
Width 1920 Height 1080
Viewport Properties
X Pos 0 Y Pos 0
Width 1920 Height 1080

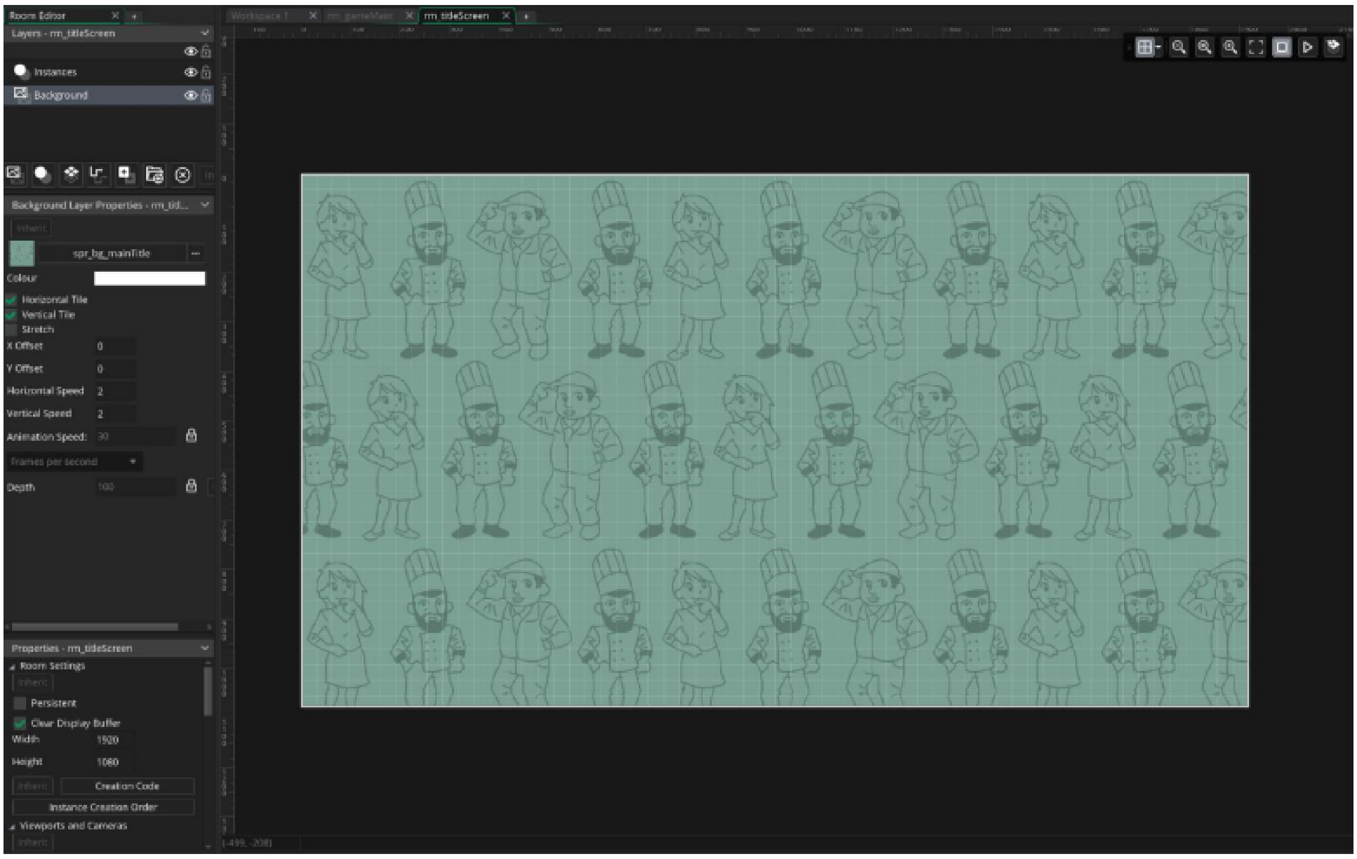
Next, in the Background Layer Properties section, is a drop-down menu that currently says "No Sprite."

Click this and choose the spr_bg_mainTitle Sprite we just imported. You'll notice that it's a square image, just sitting in the top-left corner of the room.

What we want to do is tile the image.

So, in the Background Layer Properties section, do the following:

- Check `Horizontal Tile`
- Check `Vertical Tile`
- Change the `Horizontal Speed` to 2
- Change the `Vertical Speed` to 2



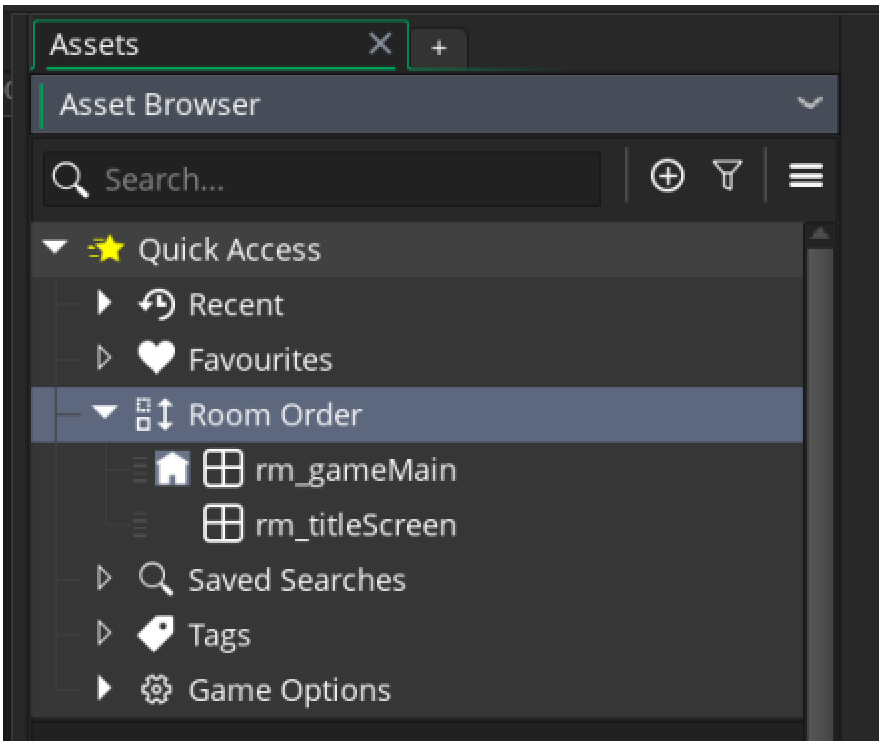
spr_bg_mainTitle
Colour
✓ Horizontal Tile
✓ Vertical Tile
Stretch
X Offset 0
Y Offset 0
Horizontal Speed 2
Vertical Speed 2
Animation Speed: 30
Frames per second
Depth 100

Persistent
✓ Clear Display Buffer
Width 1920
Height 1080

We should test our new Room to see what we've done so far in action. However, if we run the game again now, we won't see it.

That's because of our game's current Room Order. You might ask: if our game has multiple Rooms, how does GameMaker Studio 2 know which one to load first?

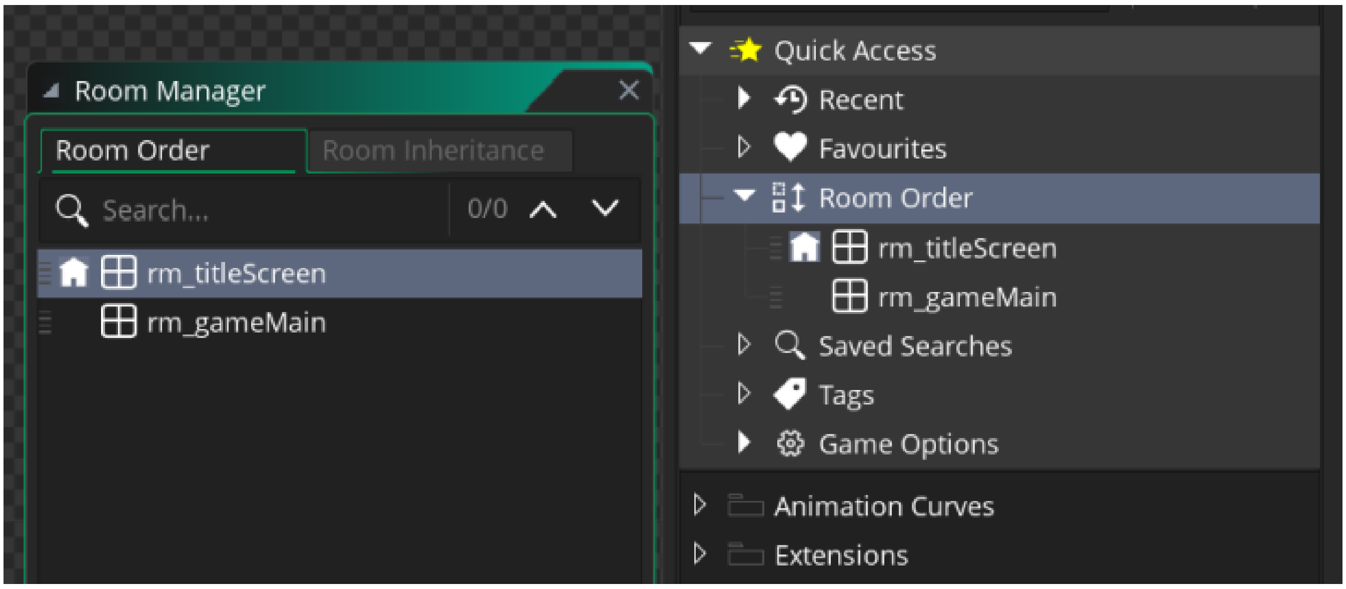
Look in the Asset Browser; you'll see a special section called Quick Access. Open it, and you'll see another sub-section called Room Order.



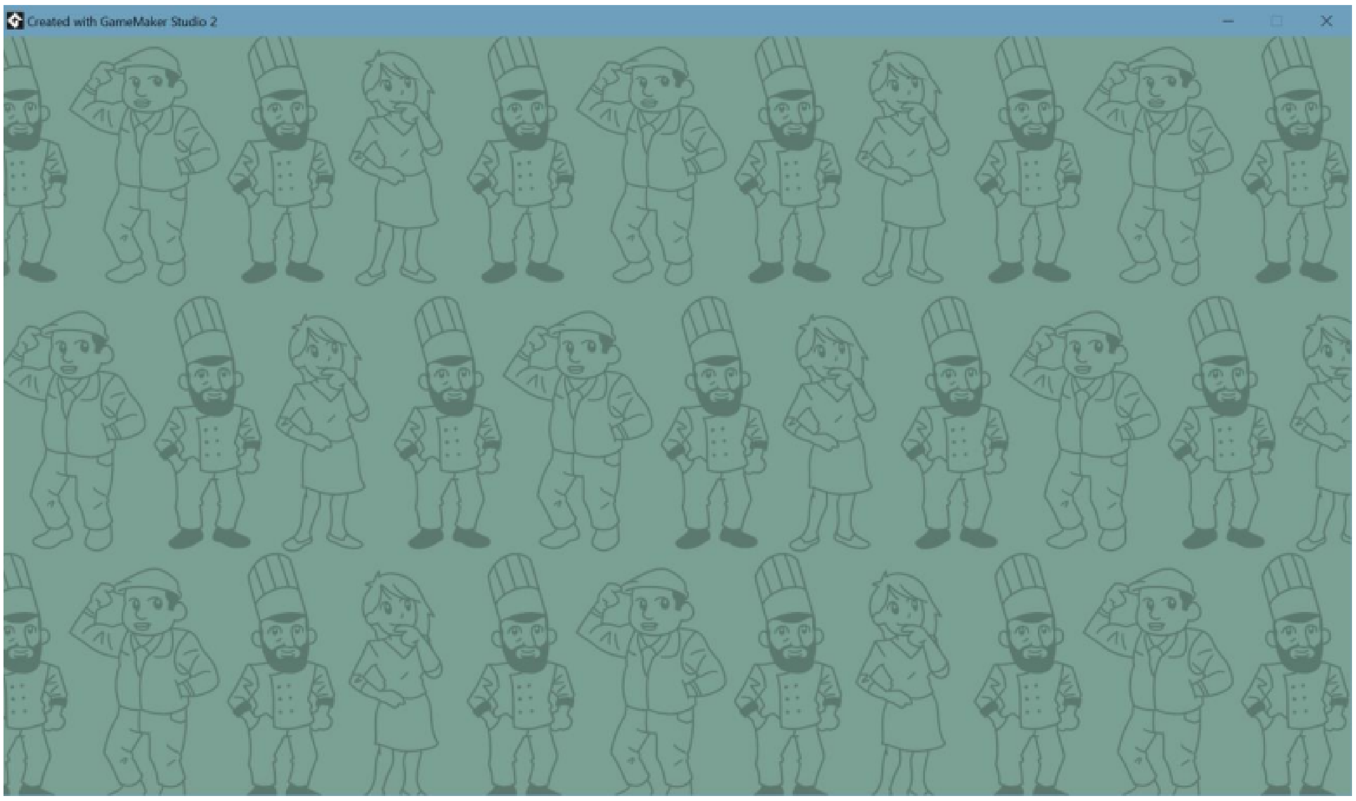
In this Room Order list, you'll see every Room in your game. (We currently only have two.)

You'll notice that rm_gameMain has a little home icon next to it; this indicates that rm_gameMain will be the first Room shown to the player when our game is loaded.

To change this, drag rm_titleScreen above rm_gameMain. If rm_titleScreen has the home icon beside it, you know it will be the "first" Room in the game.



With this done, run the game again to test it. You should see our new title screen Room, and the tiled background image scrolling diagonally.



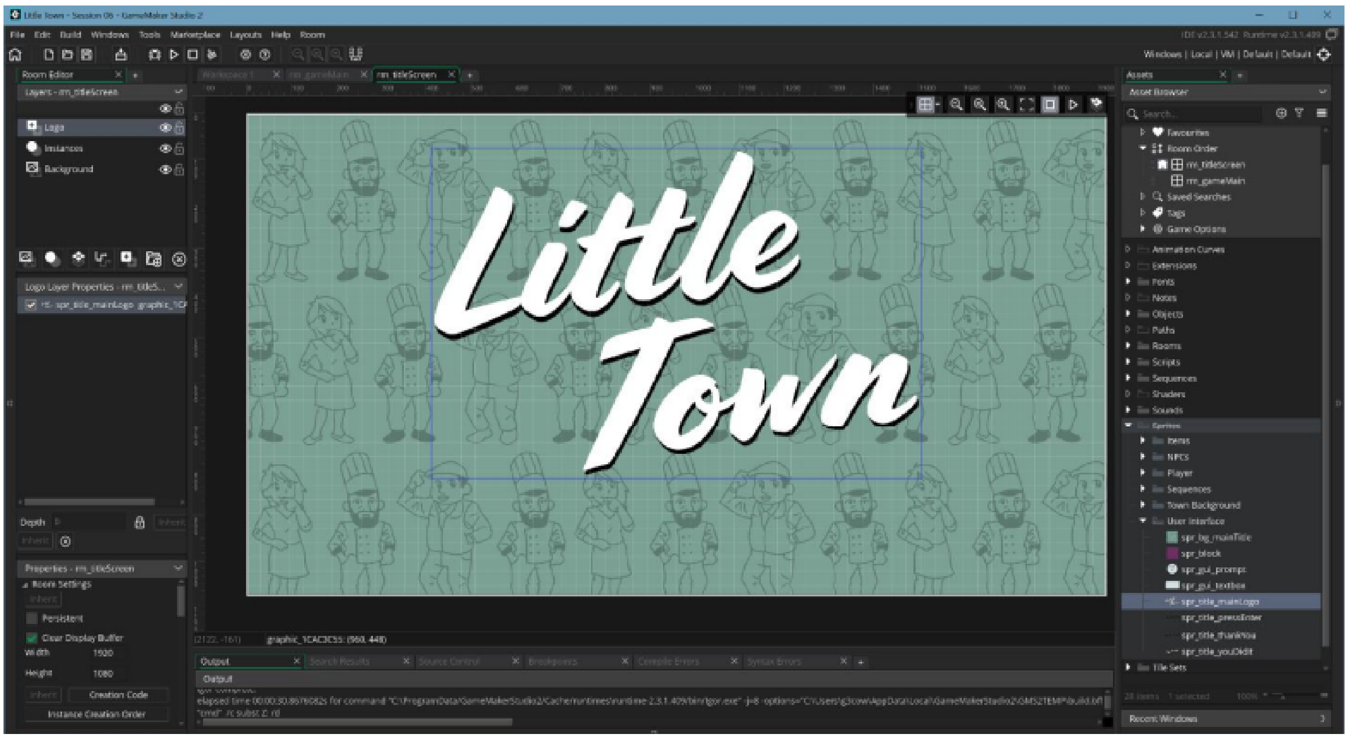
To change the direction and speed of the moving background, change the Horizontal Speed and Vertical Speed settings in the Room Editor for the title screen Room.

Now that we have our title screen Room set up, we need to populate it with two things: a logo and a prompt for the player to begin playing the game.

Open rm_titleScreen again. In the Layers section of the Room Editor, create a new Asset Layer.

Rename this layer Logo.

With this new layer selected, drag the spr_title_mainLogo Sprite from the Asset Browser into the Room. Position it however you like.



Next, we are going to create an animated prompt to encourage players to press a key in order to start the game. And thankfully, have the perfect tool for doing this: Sequences!

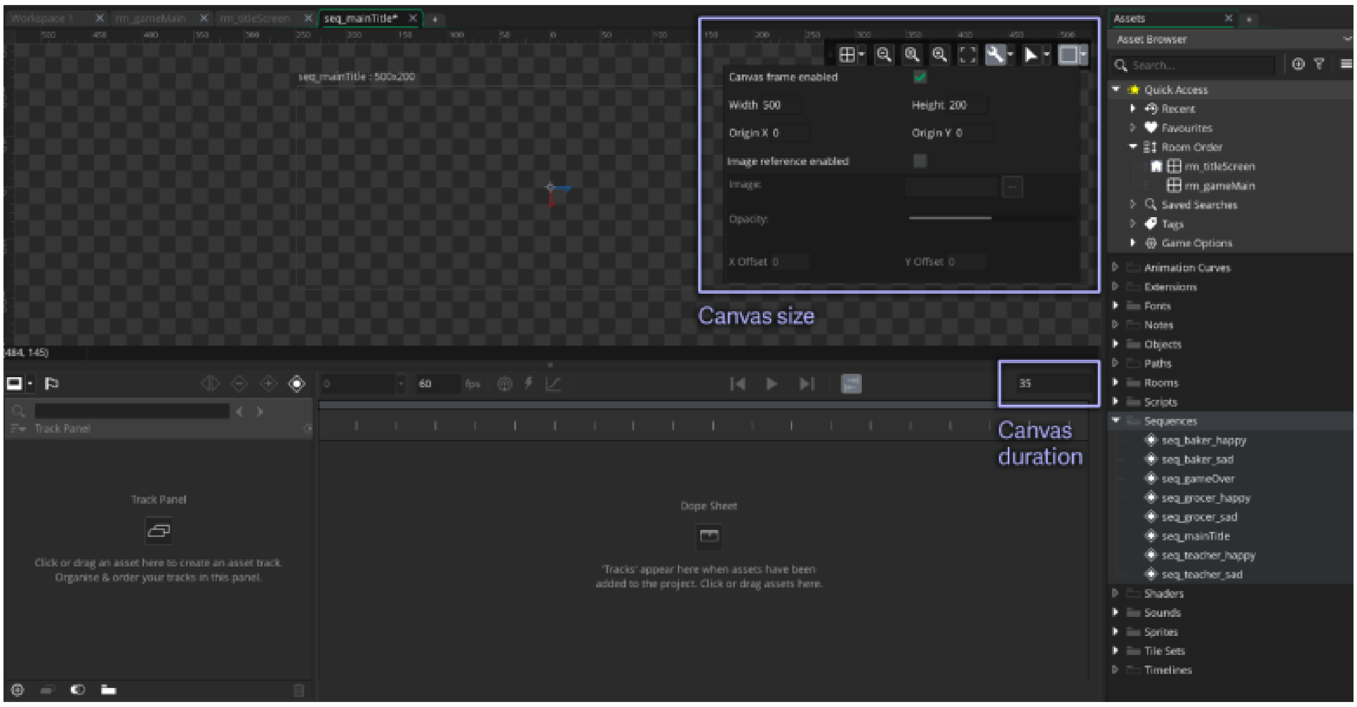
Right-click on the Sequences group in the Asset Browser and choose Create > Sequence.

Rename this new Sequence seq_mainTitle.

This Sequence doesn't need to have a large Canvas like our previous ones, so let's do some setup.

In the Sequence Editor, make the following changes:

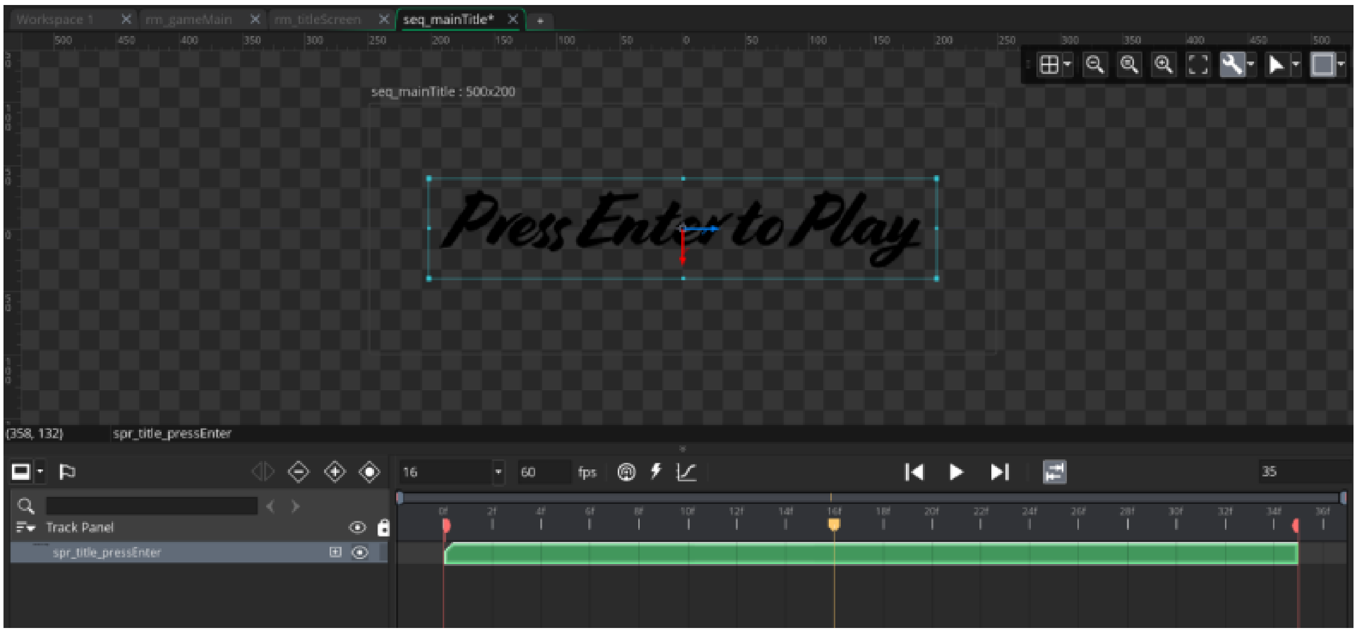
- Change the Canvas width to 500 and the height to 200
- Make the Sequence duration 35 frames



Canvas frame enabled
Width 500
Height 200
Origin X 0
Origin Y 0
Image reference enabled
35

Next, drag the spr_title_pressEnter Sprite from the Asset Browser onto the Canvas (position it in the centre).

In the Dope Sheet, make sure the asset key for the spr_title_pressEnter track spans the entire Sequence.



Next, we want to create a simple fade effect

Move the yellow playhead a little ways into the Sequence (say, around frame 5 or so).

Select the asset key and click the "Add parameter track" button in the Track Panel. Choose to add a Colour Multiply parameter track.

Next, record a keyframe at this position.

Select the new keyframe and click the colour swatch in the Colour Multiply parameter track (if you don't see this detail, click the arrow to the left of spr_title_pressEnter in the Track Panel to reveal it).

In the Colour Picker that pops up, set the Alpha channel to 0. Click "OK" when you're done. We've made the "Press Enter to Play" asset transparent.

Next, move the playhead to near the end of the timeline, but not to the last frame. We suggest around frame 30.

Repeat the process at this new position:

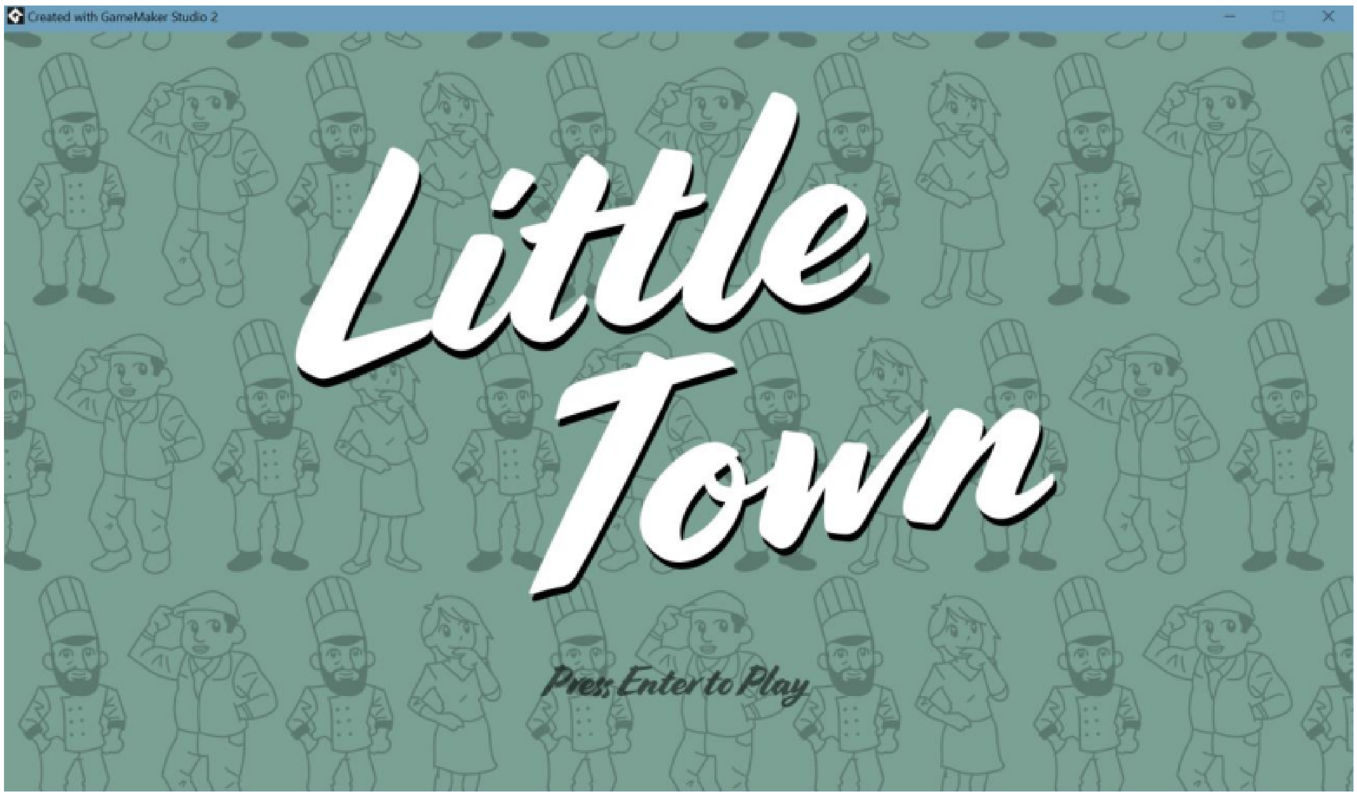
- Record a keyframe here
- Select the keyframe and click the colour swatch in the `Colour Multiply` parameter track
- In the Colour Picker, set the `Alpha` back to 255

If you scrub through the Sequence, or press the Play button to play it, you should see the text asset now fade from transparent to opaque.

Now that we have a simple linear fade effect in place, we can do something new with it.

What we actually want is for this asset to fade in and out repeatedly, for as long as the player remains on the title screen of our game.

Once you're happy with the layout of the title screen, run your game to test it. You should see the "Press Enter to Play" prompt fading in and out, while the tiled background scrolls indefinitely.



Our new title screen is looking lovely, but we still need to add functionality to it.

Rather than make new Objects, we're going to use one we already have:
obj_control

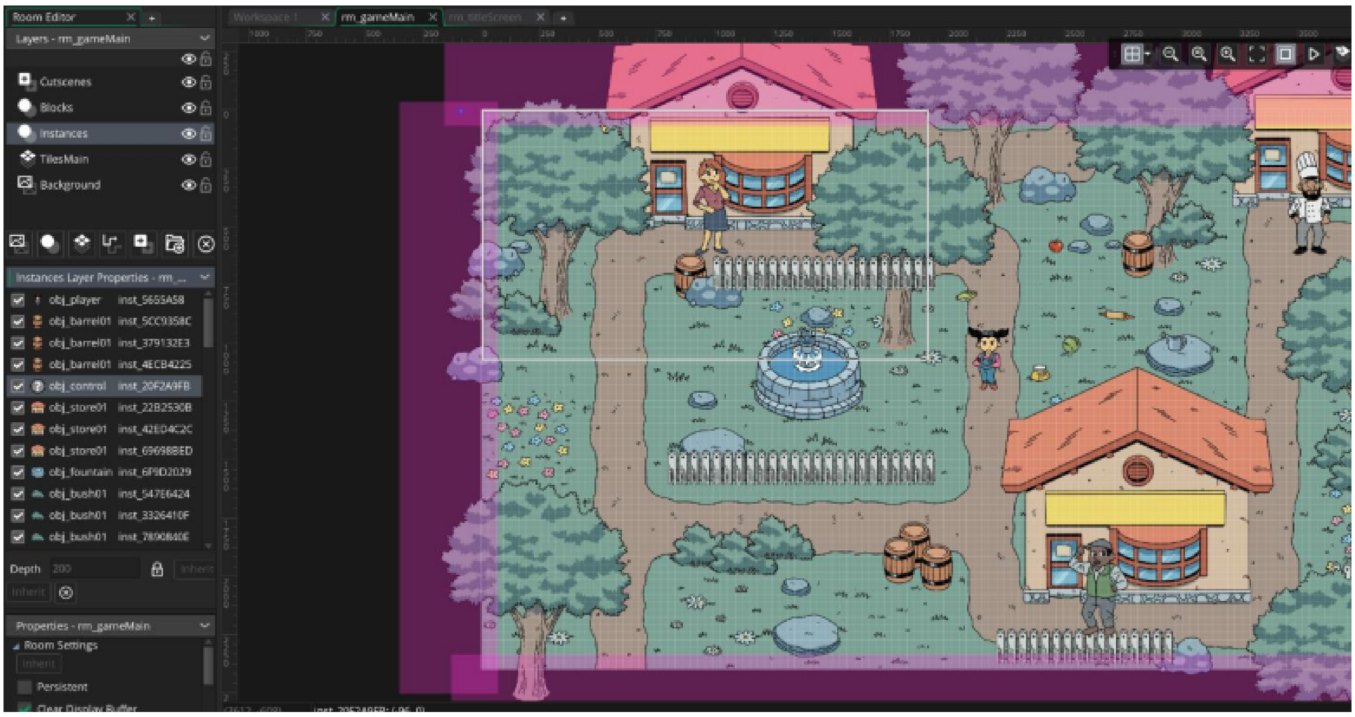
Open rm_gameMain and select the Instances layer.

In the Instances Layer Properties section, you'll find a long list of every single instance that's been placed on that layer.

If you scroll through the list, you'll find the instance of obj_control that we placed before.

Select this instance in the list, and press Delete to remove it from the Room.

You can also just select the instance of obj_control within the Room itself and delete it.



Open rm_titleScreen again and select the Instances layer.

Drag obj_control from the Asset Browser into the room and place it wherever you like.

Next, open obj_control from the Asset Browser.

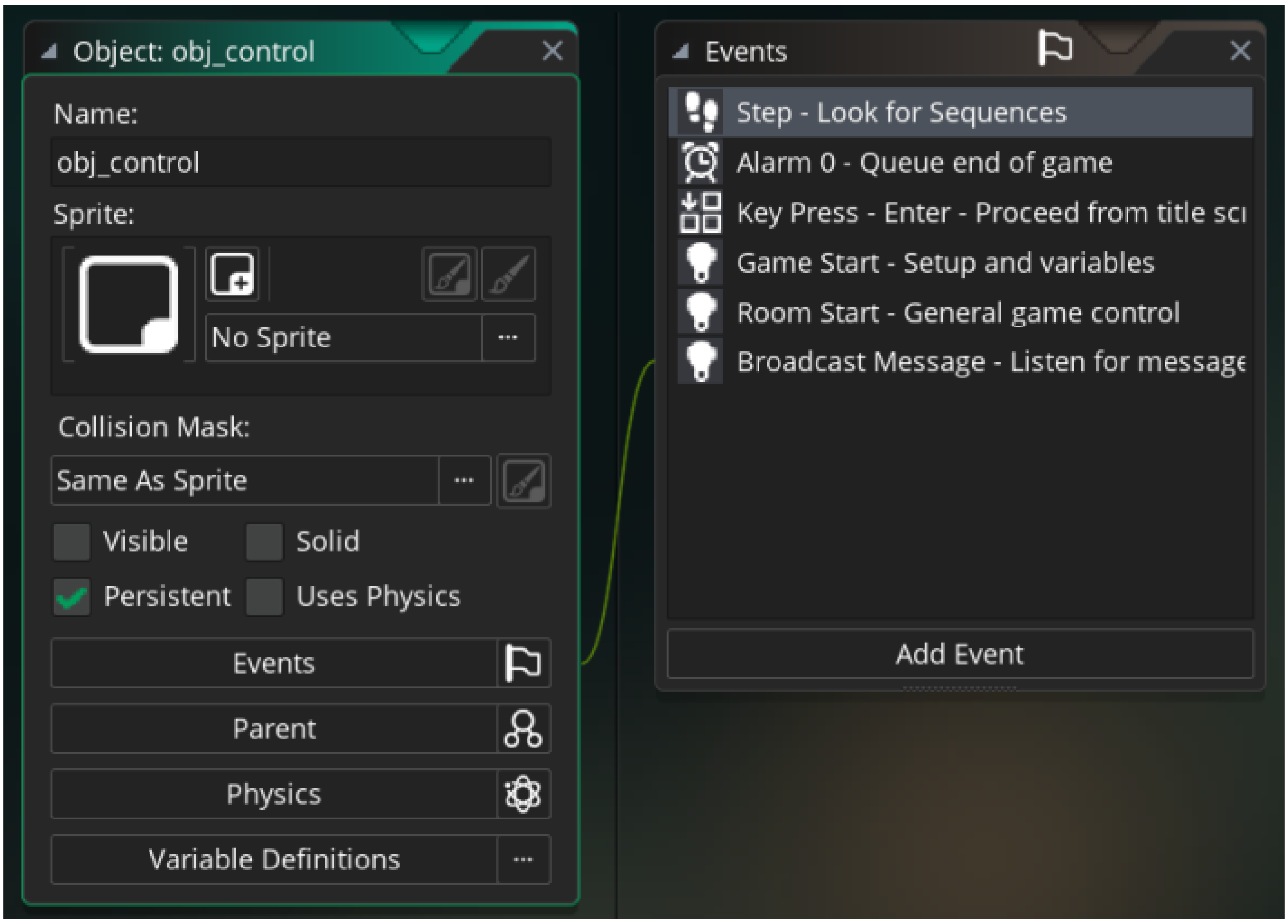
In the Object Editor, there are options such as Visible, Solid, etc. One of these options is Persistent.

Normally, an Object only exists within the Room it was created (or placed). If you change Rooms, any Object in the previous Room ceases to exist, as far as your game is concerned.

However, a persistent Object remains in the game once it is created (or once the game loads a Room that contains it). A persistent Object travels between Rooms automatically.

Since we’ve moved obj_control from rm_gameMain to rm_mainTitle, we need to make it persistent so that it will carry over to the main gameplay and continue to do all the things we’ve written for it to do.

So, check Persistent in the Object Editor.



Object: obj_control

Name: obj_control

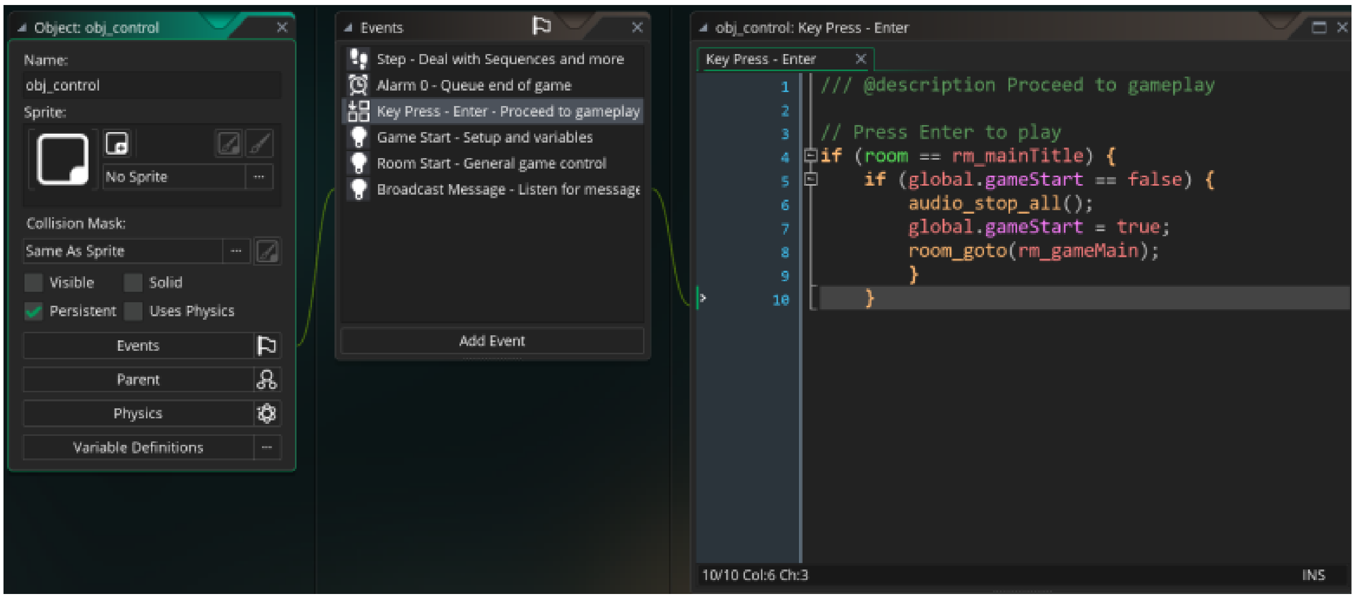
Sprite: No Sprite

Collision Mask: Same As Sprite

- [ ] Visible
- [ ] Solid
- [x] Persistent
- [ ] Uses Physics

Events

- Step - Look for Sequences
- Alarm 0 - Queue end of game
- Key Press - Enter - Proceed from title sc
- Game Start - Setup and variables
- Room Start - General game control
- Broadcast Message - Listen for message



Object: obj_control

Name: obj_control

Sprite: No Sprite

Collision Mask: Same As Sprite

- [ ] Visible
- [ ] Solid
- [x] Persistent
- [ ] Uses Physics

Events

- Step - Deal with Sequences and more
- Alarm 0 - Queue end of game
- Key Press - Enter - Proceed to gameplay
- Game Start - Setup and variables
- Room Start - General game control
- Broadcast Message - Listen for message

obj_control: Key Press - Enter

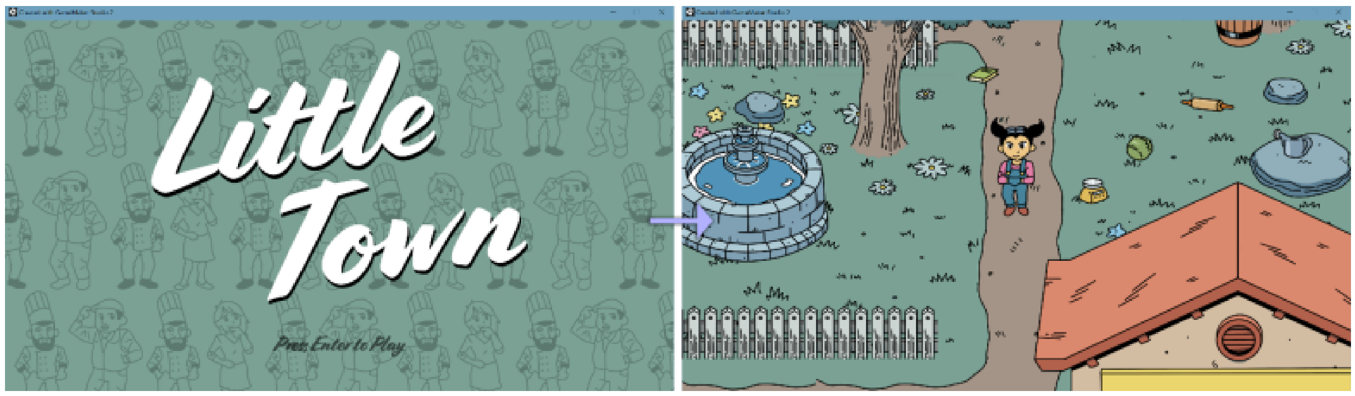
Key Press - Enter

```
/// @description Proceed to gameplay

// Press Enter to play
if (room == rm_mainTitle) {
    if (global.gameStart == false) {
        audio_stop_all();
        global.gameStart = true;
        room_goto(rm_gameMain);
    }
}
```

10/10 Col:6 Ch:3 INS

Run your game again to test it again. On the title screen, you should hear your music looping, and the “Press Enter to Play” asset will be gently blinking in and out.

Press the Enter key — you should transition immediately to the gameplay room, where you can begin the game as before.



Conclusion

You can change what each of the three NPCs says for their myText Variable Definition to give clearer or more difficult hints as to which item they want.
You can also hide the six items in more devious locations, or design your town to make them harder to spot.
Currently our Little Town has six items in it.
You'll notice that the items can seem relevant to more than one character.