MICROGAME JAM 2024

PRESENTED BY THE MCMASTER GAME DEVELOPMENT CLUB

Hello and welcome, fellow gamers and game developers, to the first game jam for this year! Us execs have been planning this game jam behind the scenes for quite a while now, and we hope you have fun!

The Basics

If you haven't participated in a game jam before, here are the basics. Game jams are basically very informal contests. Participants are given a theme and a time limit and let loose to showcase their skills and bring their crazy ideas to life. There may or may not be prizes at the end of a jam – this jam has prizes! The theme this time is Microgames.

Microgames

Microgames are simple, short minigames that take 2-10 seconds to play – cutting off after the time limit to switch to a new one. They usually have a basic control scheme and straightforward mechanics. They're inspired by the *WarioWare* series, known for its eccentric collection of microgames. Last year, people came together to create tons of microgames, and you can play all the entries from that event here!

In this game jam, participants will create a microgame using **Godot**, which will then be incorporated into a larger compilation.

For the rules & more information, see #microgame-jam-2024 in the MacGDC discord.

Dates

November 13th, 2024 – Game jam start & tutorial workshop

November 18th and 25th, 2024 – Gamedev Circles (also a help session)

November 27th, 2024 – Game jam end date

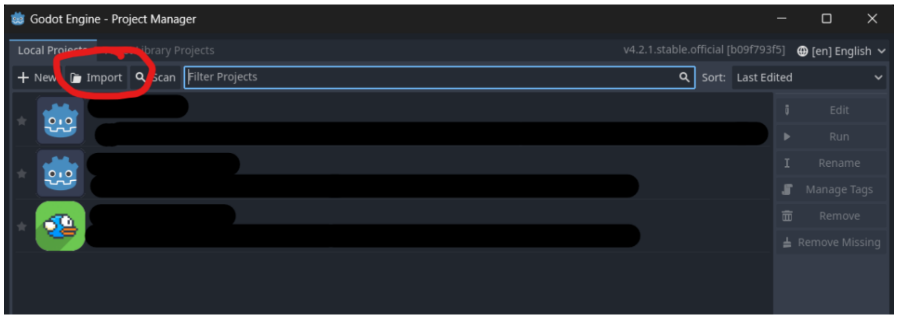
December 2nd, 2024 – Winners announced at Gamedev Circles

Tutorial

Follow the steps below for a quick guide on how to make a microgame or check out the workshop recording in the Discord!

In this tutorial, we're assuming you have some prior Godot experience. If you're an absolute beginner, we recommend watching the Brackeys Beginner Godot Tutorial!

1) Download and extract the template project from the .zip file posted in the Discord.
2) Open Godot and import the project.

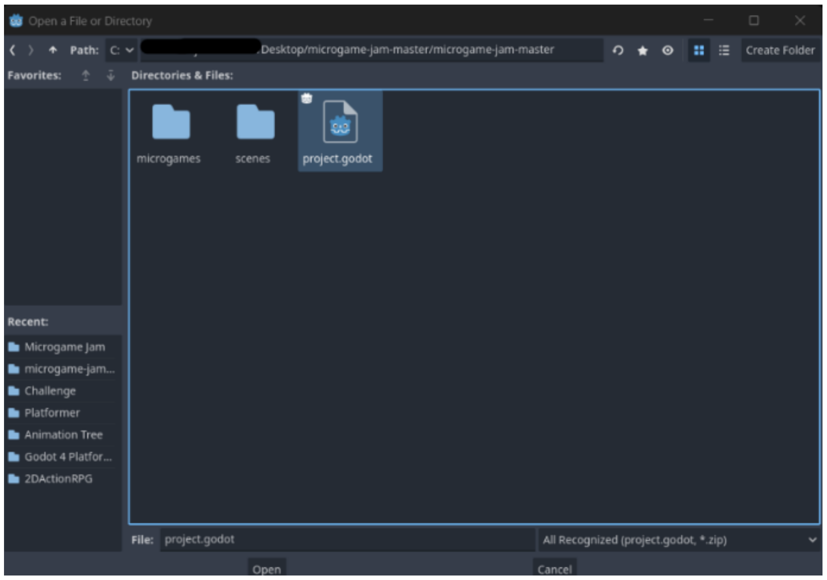


v4.2.1.stable.official [b09f793f5]

[en] English

Sort: Last Edited

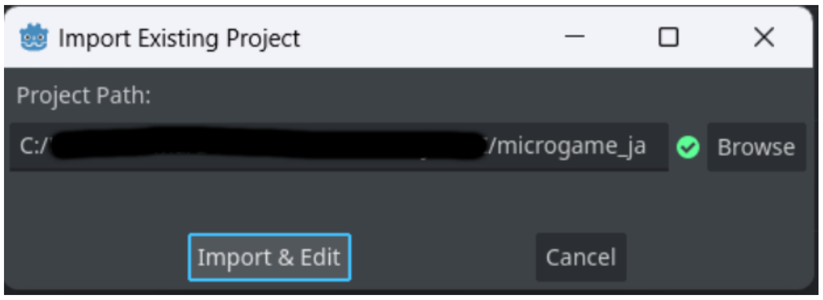
[redacted]



Path: C: [redacted] Desktop/microgame-jam-master/microgame-jam-master

File: project.godot

All Recognized (project.godot, *.zip)

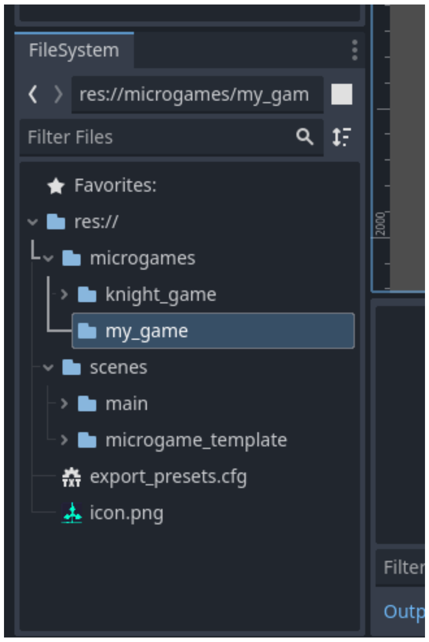


Project Path:

C:/[redacted]/microgame_ja

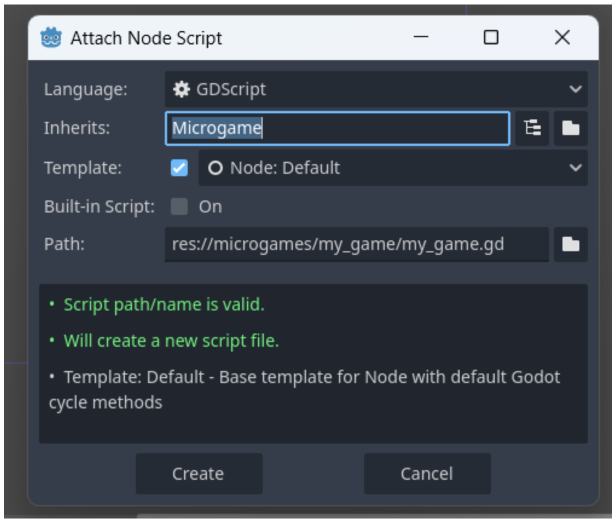
After you import the project, hit Play to run it – there should be one playable microgame.

3) In the **FileSystem Dock** in the bottom left, make a new folder for your microgame. The folder should be located inside the microgames folder.



This folder should house all your game files – the assets you use, the ".tscn" scene files, the ".gd" script files, etc. We called ours "my_game" but you should name it something unique.

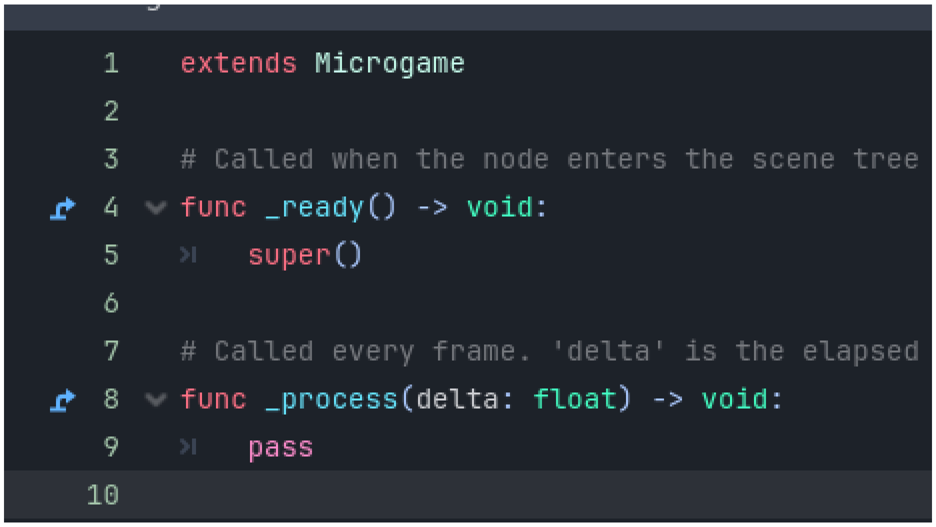
4) In the "my_game" folder, create a new scene with a **Node2D** as its root node. This will be the main **SceneTree** of your game.
5) Attach a script to the root node. Make sure the script inherits the Microgame class.


Attach Node Script

Language: GDScript
Inherits: Microgame
Template: ✓ Node: Default
Built-in Script: On
Path: res://microgames/my_game/my_game.gd

- Script path/name is valid.
- Will create a new script file.
- Template: Default - Base template for Node with default Godot cycle methods

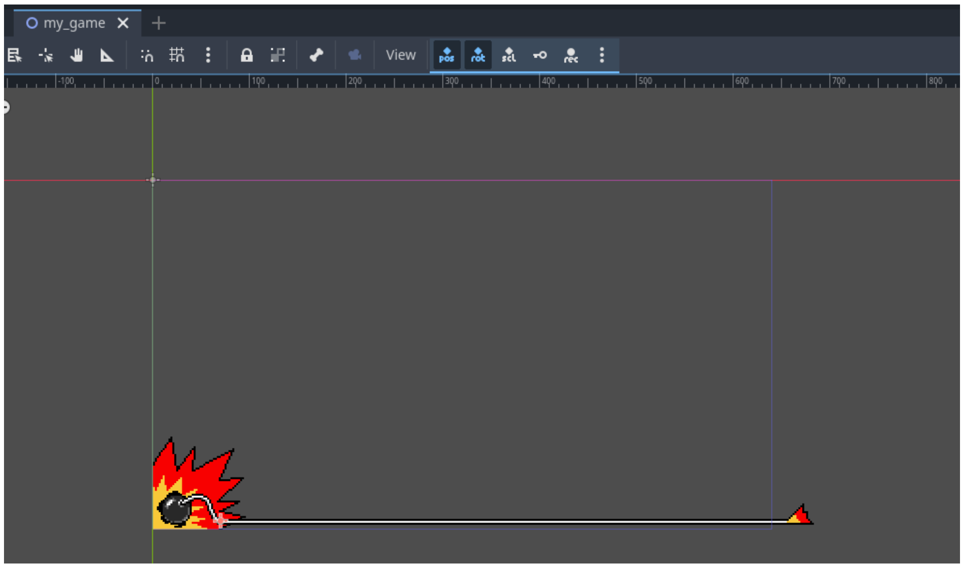
6) Make sure that the first line of the **_ready()** function is calling **super()**, then you can put the rest of the code you want. Calling **super()** ensures that the **_ready()** function of the **Microgame class** is called so that your microgame works.



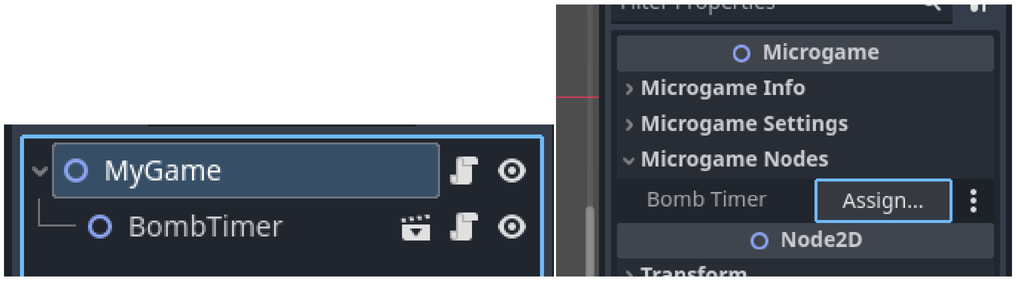
```
extends Microgame

# Called when the node enters the scene tree
func _ready() -> void:
	super()

# Called every frame. 'delta' is the elapsed
func _process(delta: float) -> void:
	pass

```

This is what your code should look like. The Microgame class will take care of all the code required to fit your microgame into the main game. But we still need a timer!

7) On your microgame's root node, instantiate a **bomb_timer** scene. Right-click on the root node and click "**Instantiate Child Scene**." In the popup, you can search "bomb", and the timer should show up.
8) Position the bomb at an appropriate area on the screen.

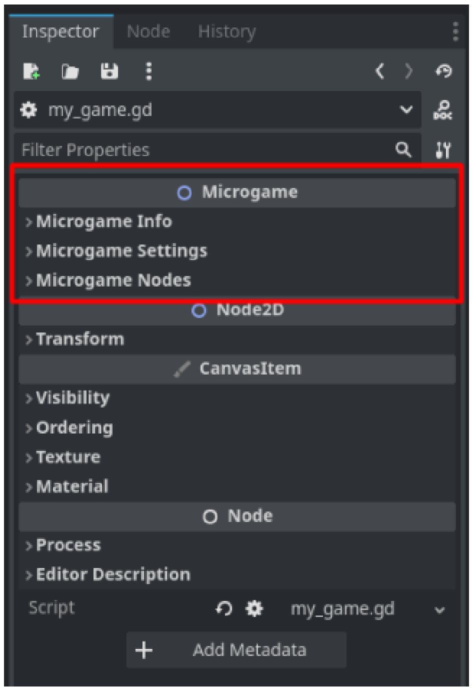


9) Next, in the root node's **Inspector Dock**, locate the **Bomb Timer** property. Click "Assign" and then you can select the **bomb_timer** that we just added.

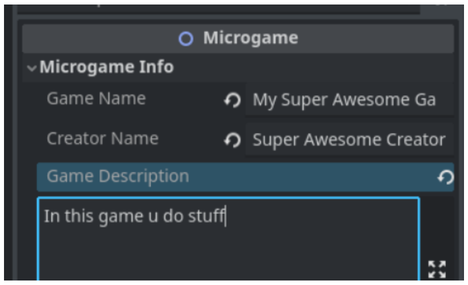


At this point, you can click Run Current Scene to watch the timer tick down!

10) Because we extended **Microgame**, we get access to a lot of custom-made parameters/settings. Click on the root node of your microgame, and in the Inspector dock you should see a list of settings.



11) Make sure to fill out your Microgame Info and Settings.



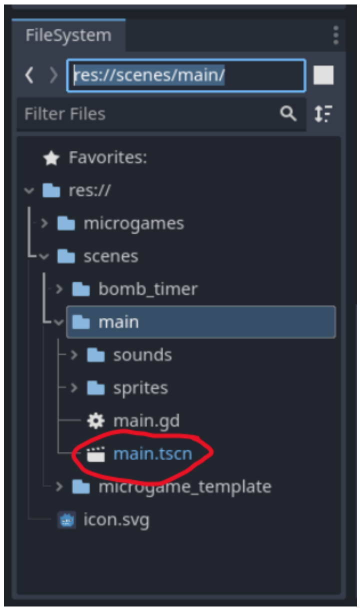
Game Name My Super Awesome Ga
Creator Name Super Awesome Creator
Game Description
In this game u do stuff

Microgame Info has all the general details about each microgame. We will need this info to verify who made the game and for the compilation and winners.

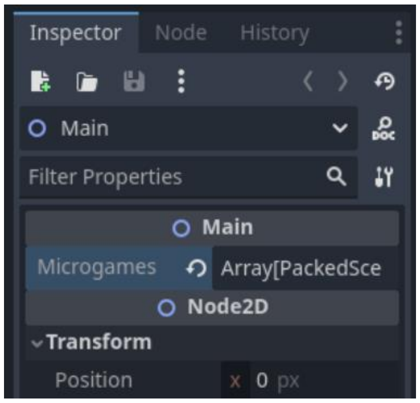
Microgame Settings has all the game design relevant parameters.

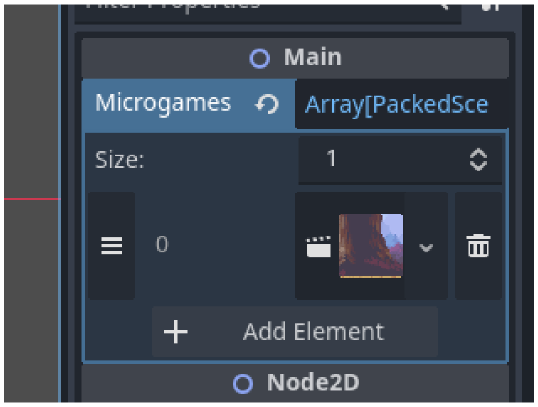
- **Game length:** How long will your game take to end? (seconds)
- **Loss on timeout:** By default, will the player win/lose when timer runs out?
- **Message:** A small message that will appear before your game starts, make it short and memorable. This will be instructions for the player.
- **Control Type:** Specify which controls your game uses. Three options: mouse, WASD + spacebar, or both. Will show up in the main game below the message.

12) Now, the main game needs to be able to "read" your microgame. In the **FileSystem Dock**, go to the path "**res://scenes/main/**". Open the "**main.tscn**" scene.

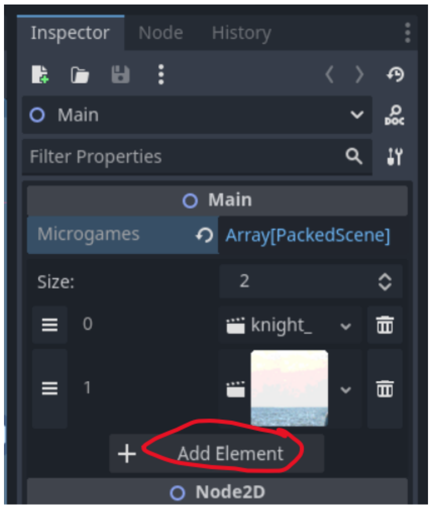


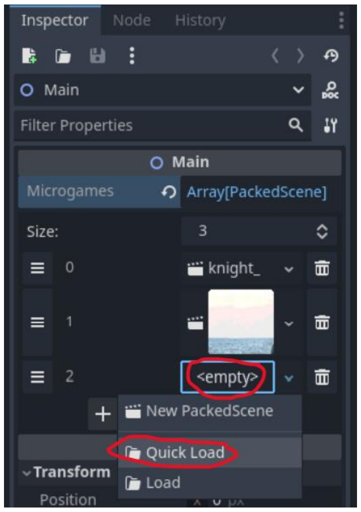
13) Go to **Inspector Dock** of the root node. You should see a property called "**Microgames**" which has already been set. Do not reset it but click on the property value (click on "**Array[PackedScene]**"). This should open a list of elements.





14) Click on "**Add Element**". This creates a new empty element. Now set the value of this element to be the main scene of your microgame by clicking on "**<empty>**" then selecting "**Quick Load**" in the dropdown menu.





15) Select your microgame node – now, when you run the main game, your game should also be run along with some templates that the execs have created! Your microgame might only show a blank screen for now since we haven't added anything to it yet.

Next Steps

Now the setup is all done. You are free to get started on your microgames!

One final important step: emit the signals "**lose_game**" and "**win_game**" when the player loses/wins respectively, so that the main game knows the outcome of each microgame. You do not need to initialize these signals (they are included in the **Microgame** class that your node extends).

Ok, now you know everything you need to be able to create a Microgame for this jam! Best of luck to you game devs!!!!!