The Micro-Game Game Jam!

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 does have some epic prizes tho).

THE THEME

WarioWare micro-games

WarioWare games are compilations of short minigames, often played back-to-back. In this game jam participants will each create one micro-game, which will be incorporated into a larger MacGDC WarioWare game!

The Rules

- This jam is intended for veterans and beginner game-devs alike.
- You must work individually.
- Submissions must be under 10 secs in length. Aim for 4-8s.
- You will have until Feb 5th to finish your micro-game. This is the showcase date.
- For this game jam, **SUBMISSIONS MUST USE THE GODOT ENGINE**. This is so your micro-game can be properly incorporated into the final project. Follow the instructions in this document or come to the tutorial meeting (Jan 22nd). There will be an additional help session as well.
- There will be prizes, so do your best!

TENTATIVE DATES:

January 22nd, 2024 – Game jam opening and tutorial.

January 29th, 2024 – Game jam help session.

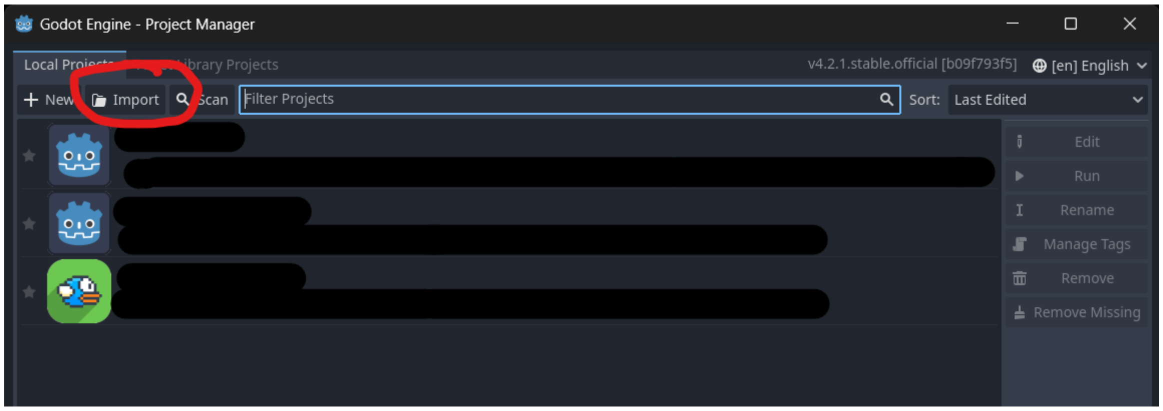
February 5th, 2024 – Game jam showcase!

Game Jam Entry: Initial Steps

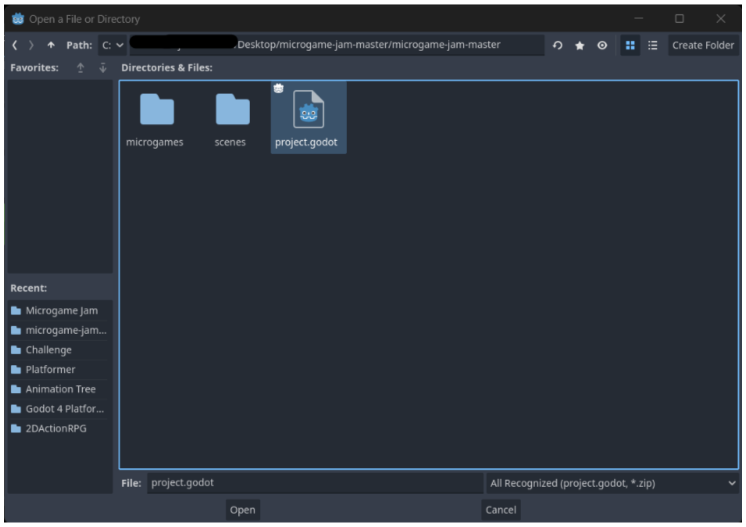
To ensure that your individual submissions can be properly incorporated into the final compiled game, please follow the instructions in this section.

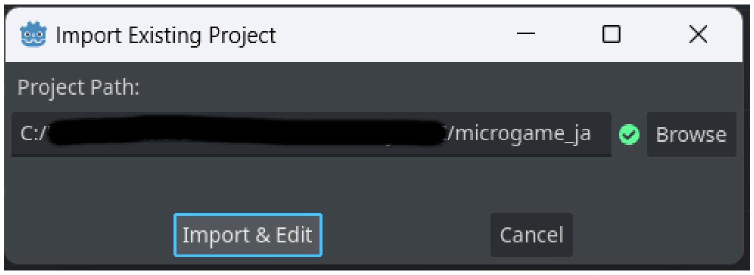
ACQUIRE THE TEMPLATE PROJECT

1. Download and extract the template project from the zip file posted in the Discord.
2. Open Godot (version 4.2) and import the project.



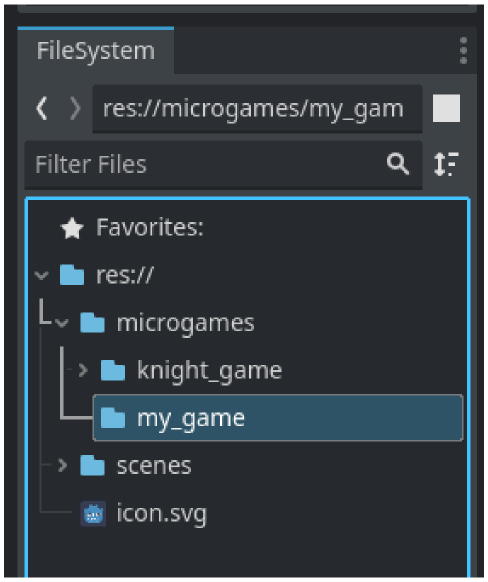
[redacted]





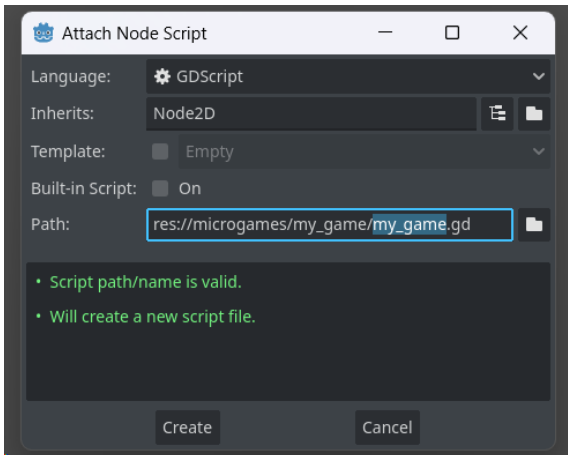
SET UP YOU GAME SCENE

3. In the **FileSystem Dock**, 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.

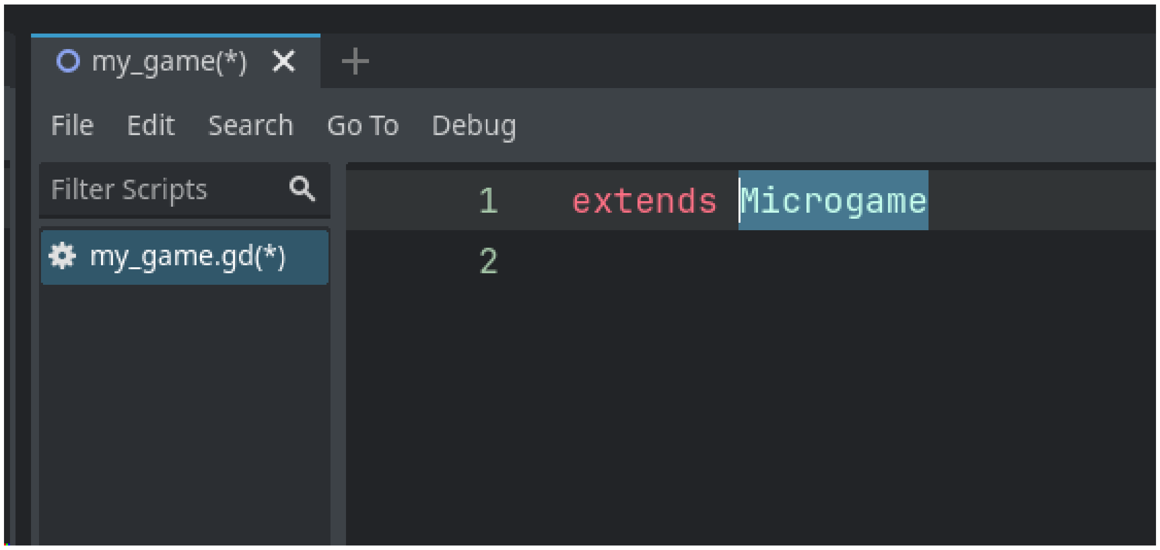
4. In the "my_game" folder, create a new scene with a **Node2D** as its root node. This will be the main **SceneTree** of your game.
5. Attach a script to the root node. Double check that this script is being saved in your microgame folder.


Attach Node Script

Language: GDScript
Inherits: Node2D
Template: Empty
Built-in Script: On
Path: res://microgames/my_game/my_game.gd

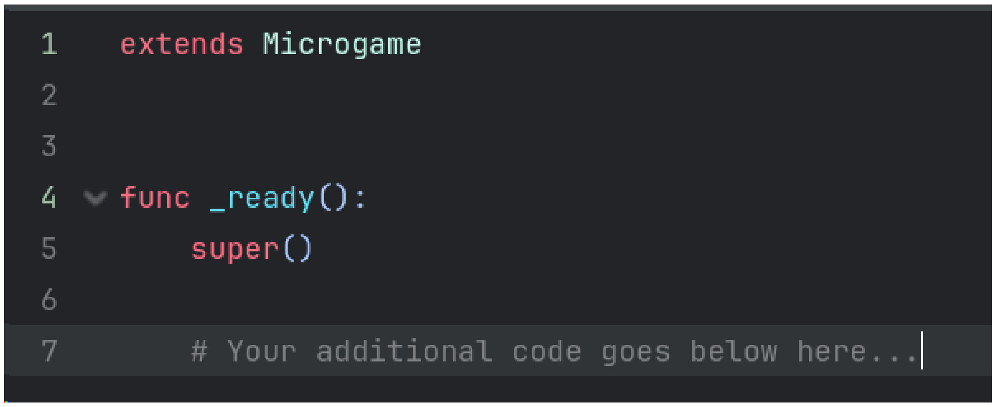
- Script path/name is valid.
- Will create a new script file.

6. Open the script. At the top, it should extend **Node2D**. Instead, we want the game to extend the **Microgame** class. This class is a custom-made class that gives your microgame all the necessary settings for a microgame, and lets it interface with the main game.



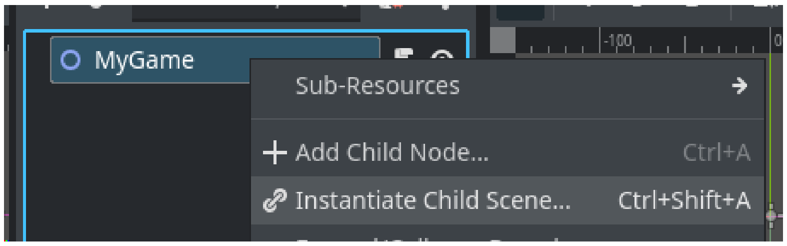
```
extends Microgame

```

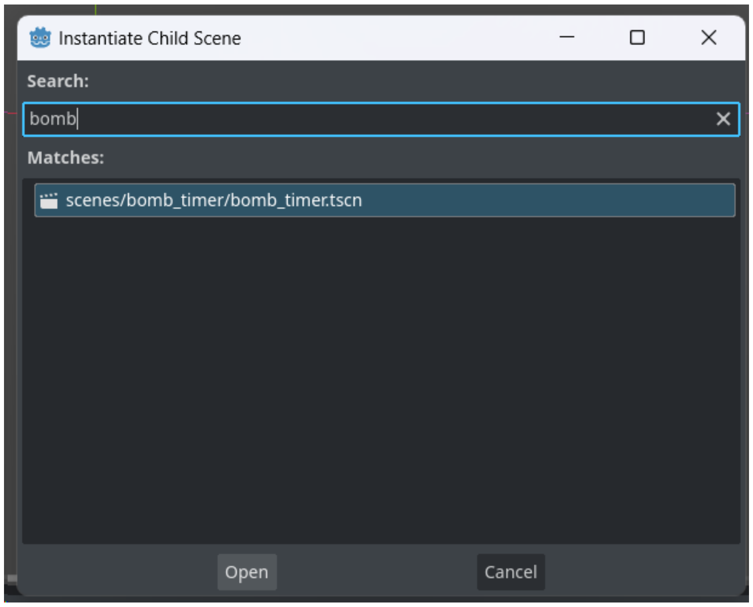
7. If your main microgame script uses the **_ready**() function, make sure that the first act of this function is calling **super**(), then the rest of the code you wish to implement. This ensures that the **_ready**() function of the Microgame class is called.



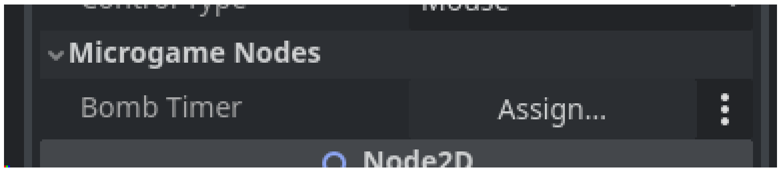
```
extends Microgame


func _ready():
	super()

	# Your additional code goes below here...
```

8. Last step, but very important. We need to add a bomb timer to your game.
9. On your microgame's root node, instantiate a **bomb_timer** scene.
   Right-click on the root node and click "**Instantiate Child Scene**." In the pop-up, you can search "bomb" and the timer should show up.



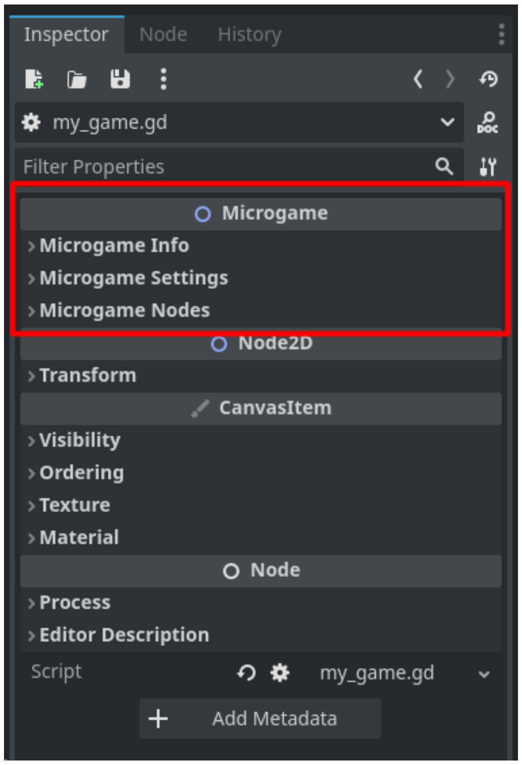


10. Position the bomb at an appropriate area on the screen.
11. Next, in the root node's **Inspector Dock**, locate the **Bomb Timer** property. Click "Assign" and then you can select the **bomb_timer** that we just added.



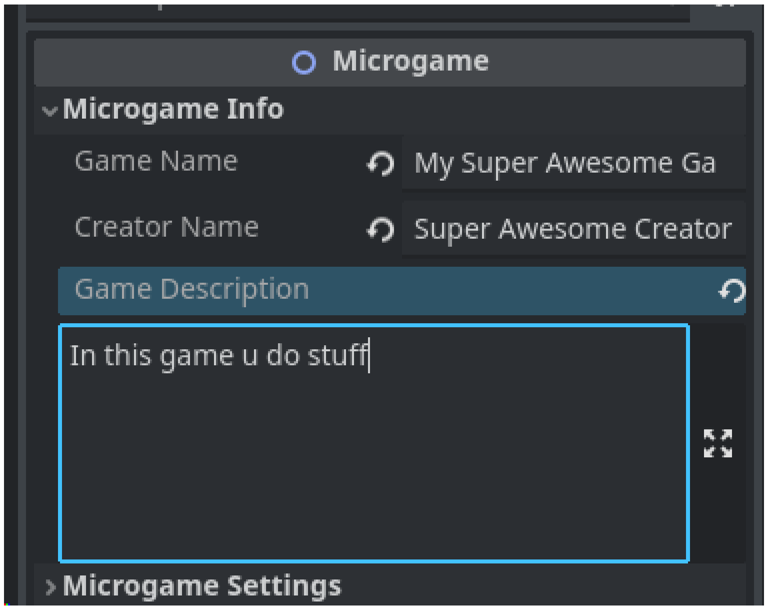
LETTING YOUR MICROGAME INTERPHASE WITH MAIN GAME

12. 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.



13. Fill out the settings/parameters.

Microgame info has all the general details about each microgame. We will need it if you win.



Game Name: My Super Awesome Ga

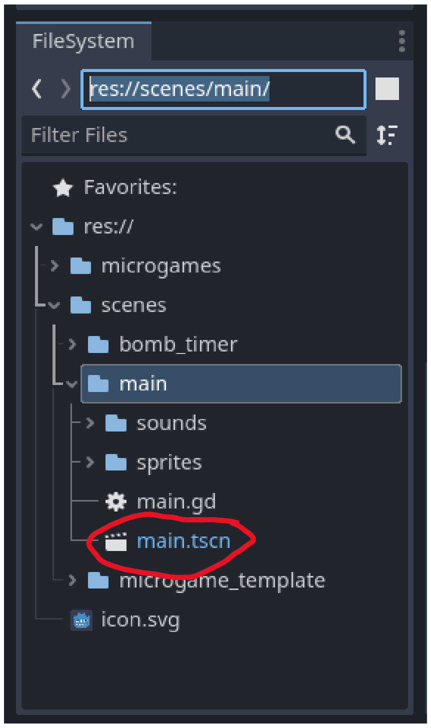
Creator Name: Super Awesome Creator

Game Description: In this game u do stuff

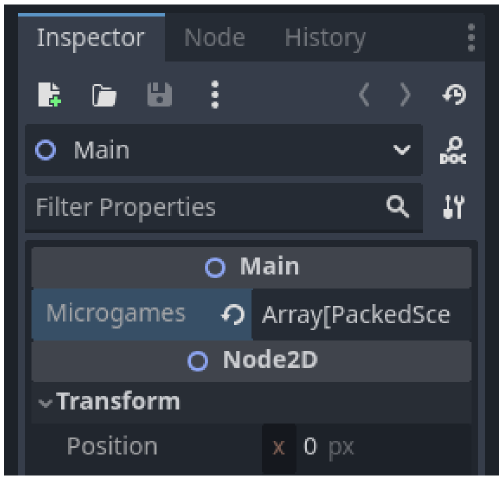
Microgame Settings has all the game design relevant parameters.

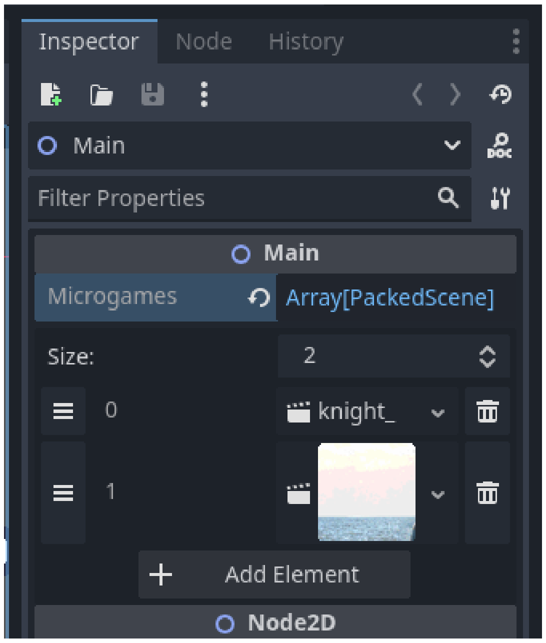
- **Game length**: How long will your game take to end? (seconds)
- **Loss on timeout**: Will the player win/lose when timer runs out?
- **Message**: A small message that will appear before your game starts, make it short and memorable. This will be instructions for the player.
- **Control Type**: Specify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

14. Now, the 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.

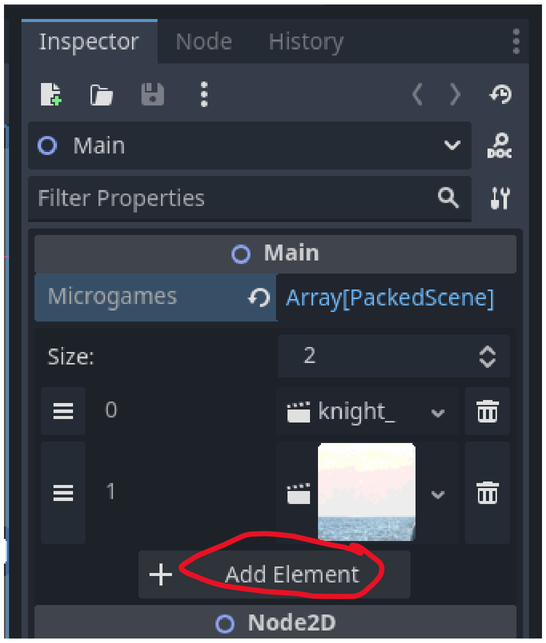


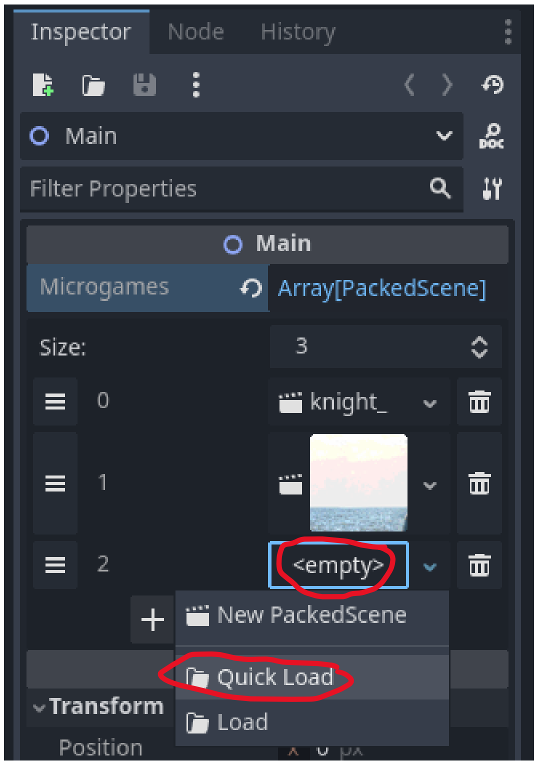
15. Go to **Inspector Dock** of the root node. You should see a property called "**Microgames**" which has already been set. Do not reset it, but click on the property value (click on "**Array[PackedScene]**").This should open a list of elements.





16. 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.





17. 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.

<u>One final important step</u>: 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!!!!!