Educational Game for Learning Beavers Developer Manual

A comprehensive developer guide to working on and contributing to the game

---

How to obtain the source code.

Our latest source code can be found at
https://github.com/Mikenunz1/EG_For_Learning_Beavs
To download from the command line interface use the following:
git clone https://github.com/Mikenunz1/EG_For_Learning_Beavs

The layout of your directory structure.

The layout of the directory structure is as follows:

The root directory contains the .github/workflows, Game Files, Online Multiplayer Demo, Research Material, Sound Feature Demos, and reports directories. It also contains the living document, the godot project file, and the export presets file.

The Game_Files directory contains folders for Assets, Scenes, and Scripts.
The Assets folder has subfolders for Sound, Sprites, and UI. Sound has Music and Sounds. Sounds has Enviroment Ambience, Environment Interaction SFX, environmental interaction SFX, environmental interaction SFX, and UI SFX. Sprites has sprite sheets and sprite statics. The Scenes directory has subfolders for Environmental, Functional, NPCs, Player, and UI. The Scripts folder contains all the GDscript files for the project.

Online multiplayer demo contains 3 scripts which are used to demonstrate the multiplayer aspects of the game. Research Material contains two files related to educational material for the game. Sound Feature Demos contains 3 Godot project files containing example sound feature implementations. Reports contain all the status reports from the initial development of the game.

How to build the software.

To build the software, you will need to download two necessary components.

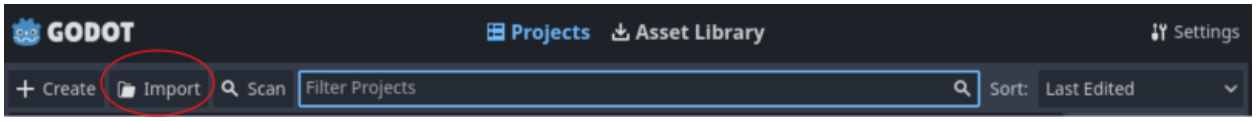
Step 1: To start, install Godot 4.3 from https://godotengine.org/download. Download the Godot Engine for your operating system. Please ensure that you choose the Godot Engine (not the .NET version) as seen below.

The downloaded .zip file will contain two executable files. Extract these files to a preferred location on your device.

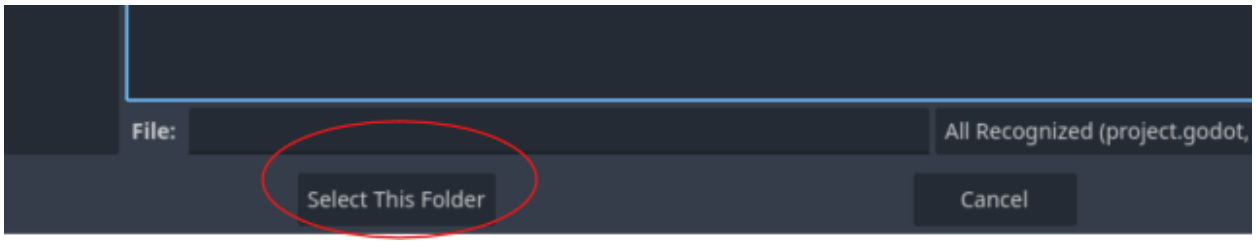
Step 2: Next, acquire the root directory containing all the necessary game files from the GitHub repository for the game: https://github.com/Mikenunz1/EG_For_Learning_Beavs/tree/main. You can either download the files or use Git's command line interface to import them into the desired location on your device.

Step 3: Now, you can import these files as a new project into Godot. Start by launching the Godot_v4.3-stable_win64.exe that you downloaded in Step 1.

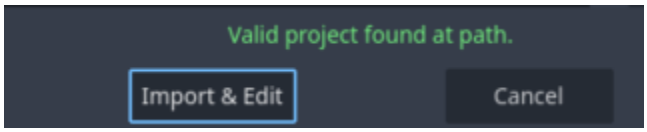
Start by clicking the import button in the top left.



Use the navigation pane to find the directory that you stored the game files. Click on the downloaded folder and press, “Select This Folder.”



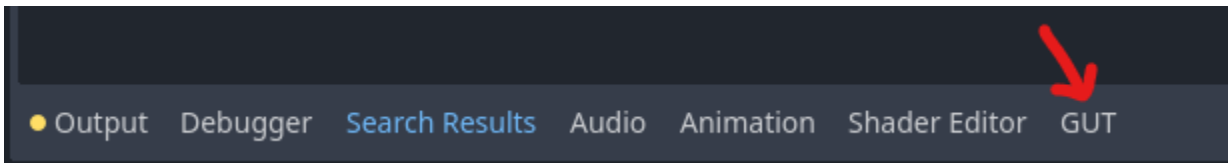
From here, select “Import & Edit” to confirm the folder selection.

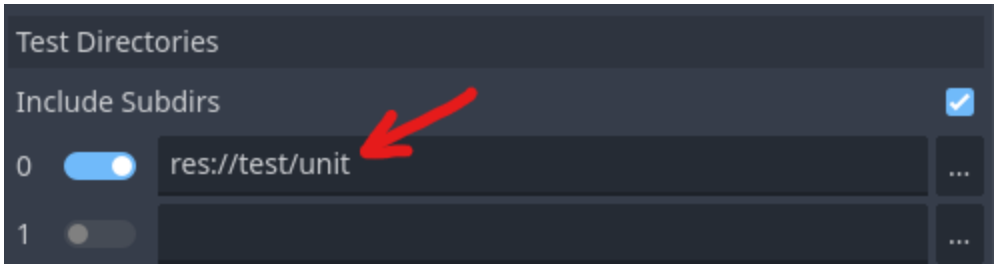


Godot will automatically open up the build of the project for editing. In this window, you will be able to view and edit the project as you deem fit.

How to test the software.

To test the software, you need to install the Godot Unit Test (GUT) plugin in Godot. This process is detailed here, https://gut.readthedocs.io/en/latest/Install.html. Once GUT is installed, navigate to the project and click on project settings. In project settings, click on plugins and enable GUT. Then you need to navigate to the GUT option on the bottom of the Godot interface. You need to configure the test directories to include res://test/unit and res://test/integration on the right side of the screen.





Then you will have the option to run all the tests. To run a specific test file, you need to open the file and then GUT will have an option to run that specific test. To run a specific test function, you can place your cursor on that function within the .gd test file, and GUT will give the option to run that specific function. For more information on how to use GUT, you can go to https://gut.readthedocs.io/en/latest/Quick-Start.html#run-tests.

How to add new tests.

For new tests, you create a .gd file in either the res://test/unit folder or res://test/integration folder. The filename must begin with “test_” or GUT will not run the test. These tests will be written by extending GutTest, and will be written using GUT test design. Each test function must be started test_, or else it will not be counted as a test in GUT. More information about writing GUT tests can be seen here https://gut.readthedocs.io/en/latest/Quick-Start.html#creating-tests.

How to build a release of the software.

The Continuous Integration automatically creates a build of the Godot project in the GitHub Repository. It will automatically create a release with the defined tag variable in the Github Actions workflow file. The version tag is defined in the .github/workflows/main.yml folder next to the tag: variable.

```
40          - name: create release
41            uses: ncipollo/release-action@v1.14.0
42            with:
43              allowUpdates: true
44              #indicates if the release should be marked as a prerelease
45              prerelease: true
46              token: ${{ secrets.GITHUB_TOKEN }}
47              generateReleaseNotes: true
48              #change the below release tag to the current version number
49              #if the version tag already exists, it will update that release
50              tag: v0.0.1
51              #the above can be replaced with ${{ github.ref_name }} to auto name
52              artifacts: ${{ steps.export.outputs.archive_directory }}/*
53
```

If the variable remains unchanged between pushes / pull requests, it will update the release with that tag. Changing the tag allows a new release to be automatically constructed. If the build and release creation is successful, the GitHub actions tab will show that the latest workflow run was successful.