
RetroHub

Release 1.0

rsubtil

Feb 24, 2024

CONTENTS

1	User guide	3
2	Theme Development	211
3	App Development	263
4	Class API	269

Welcome to RetroHub's documentation. This website contains plenty of information to help you get started with RetroHub, whether you're a regular user or a theme developer.

You can navigate the whole documentation through the side bar on the left, as well as searching for any page you want.

USER GUIDE

This section shows how to setup and configure RetroHub.

1.1 Setup

Welcome to RetroHub! This guide serves to show every step to create a setup around your retro gaming library.

These instructions can be followed for every kind of setup. It's only when you choose a theme that your setup will become highly customized to your liking.

1.1.1 Contents

Download

RetroHub can be downloaded from [GitHub releases](#).

RetroHub is available for **Windows**, **macOS** and **Linux**.

Installing

Warning: When downloading automatic builds from GitHub's CI, you may need to mark the app as executable. This is specially relevant for macOS, as not doing so will throw a generic `This app can't be opened` error. To do so, open a terminal and run:

- macOS: `chmod +x RetroHub.App/Contents/MacOS/RetroHub`
- Linux: `chmod +x RetroHub`

Windows

Windows releases work like portable applications. Extract to a folder of your choice and run the main `RetroHub.exe` executable.

macOS

macOS releases work as an app bundle. Run the downloaded `Retrohub.App`.

Warning: RetroHub is signed, but not notarized, so you need to enable apps from untrusted sources in your [Security & Privacy settings](#). For more information on what to do on specific scenarios, check [this page](#).

Linux

Linux releases work like portable applications. Extract to a folder of your choice and run the main RetroHub executable.

First Steps

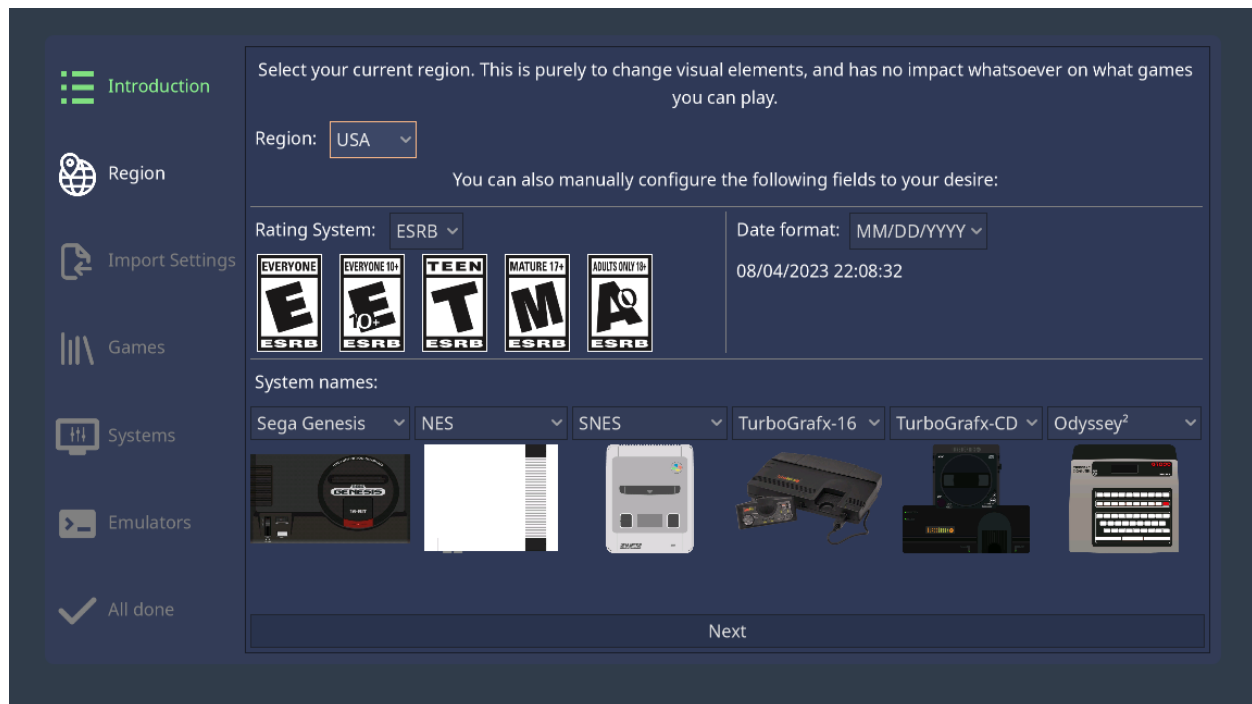
The first time you launch RetroHub, you'll be presented with a first-time wizard setup:



This setup will bootstrap your retro gaming library and set some default settings. Each section is described in more detail below.

Note: To enable the screenreader, you can press `Ctrl` on this screen.

Region

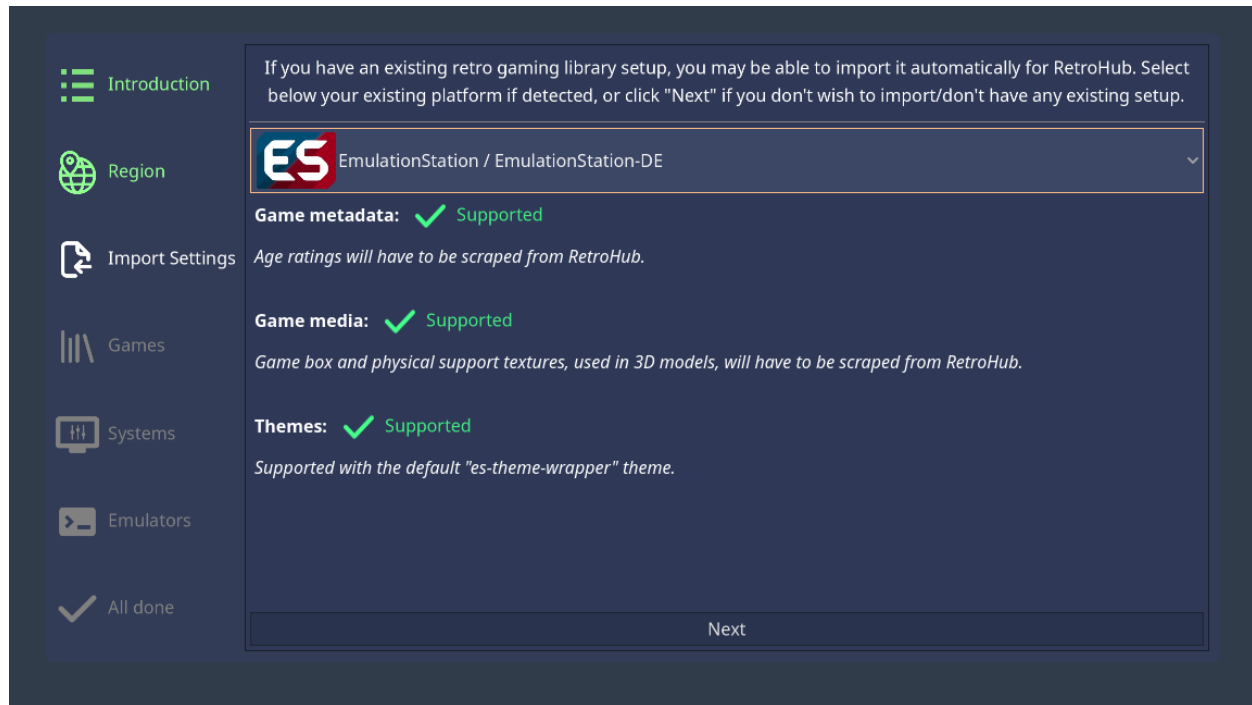


Region is used to set up some set some behavior and visual settings used throughout the app. You can pick a region to use a preset of settings, or customize each type of setting:

- **Rating System:** Game age rating system to use.
- **Date Format:** Format to present date & time.
- **System Names:** Names to use for consoles with region-specific names.

Note: Region settings are merely cosmetic and have no impact whatsoever on the games you can play.

Import Settings

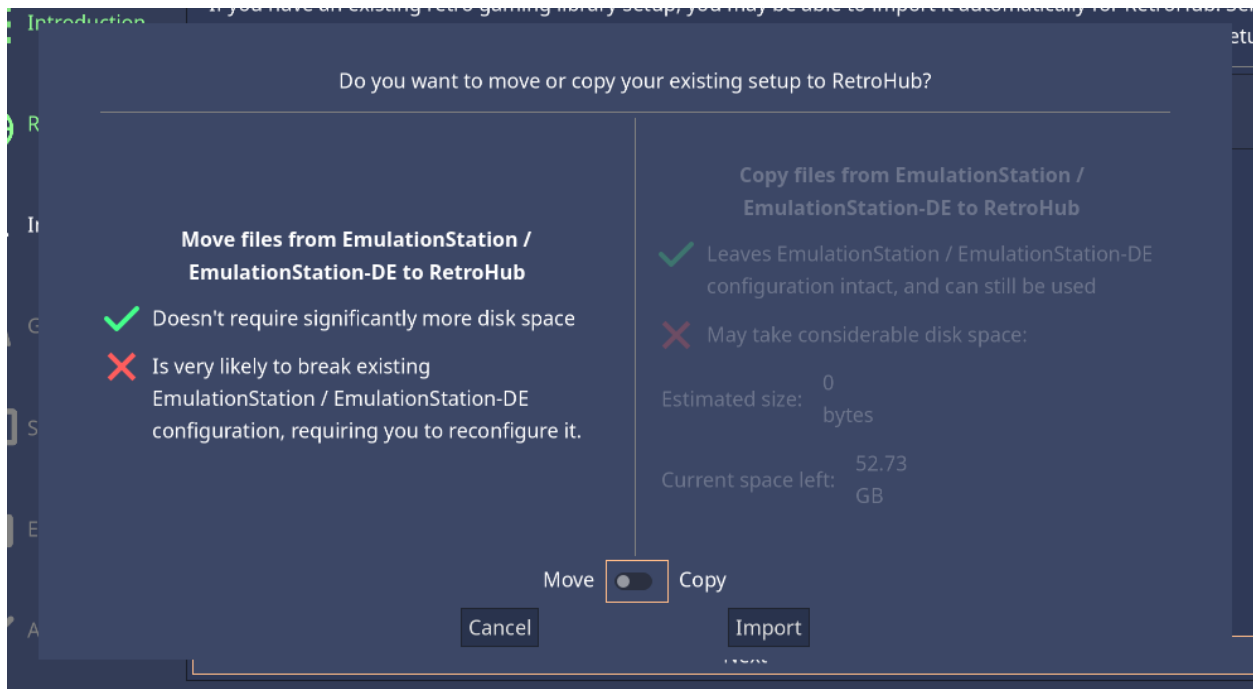


If you have an existing library setup on another frontend, RetroHub may be able to automatically import it. It will scan your files and show any valid apps found for you to import, as well as what kind of data it can be imported and any drawbacks:

- **Game metadata:** - Game text information, such as names, descriptions, publishers, release dates, play count, etc...
- **Game media:** - Game media, such as screenshots, videos, box arts, etc...
- **Themes:** Frontend specific themes which may work under a theme wrapper.

If you don't wish to import anything, you can also skip this step by picking Don't Import Settings.

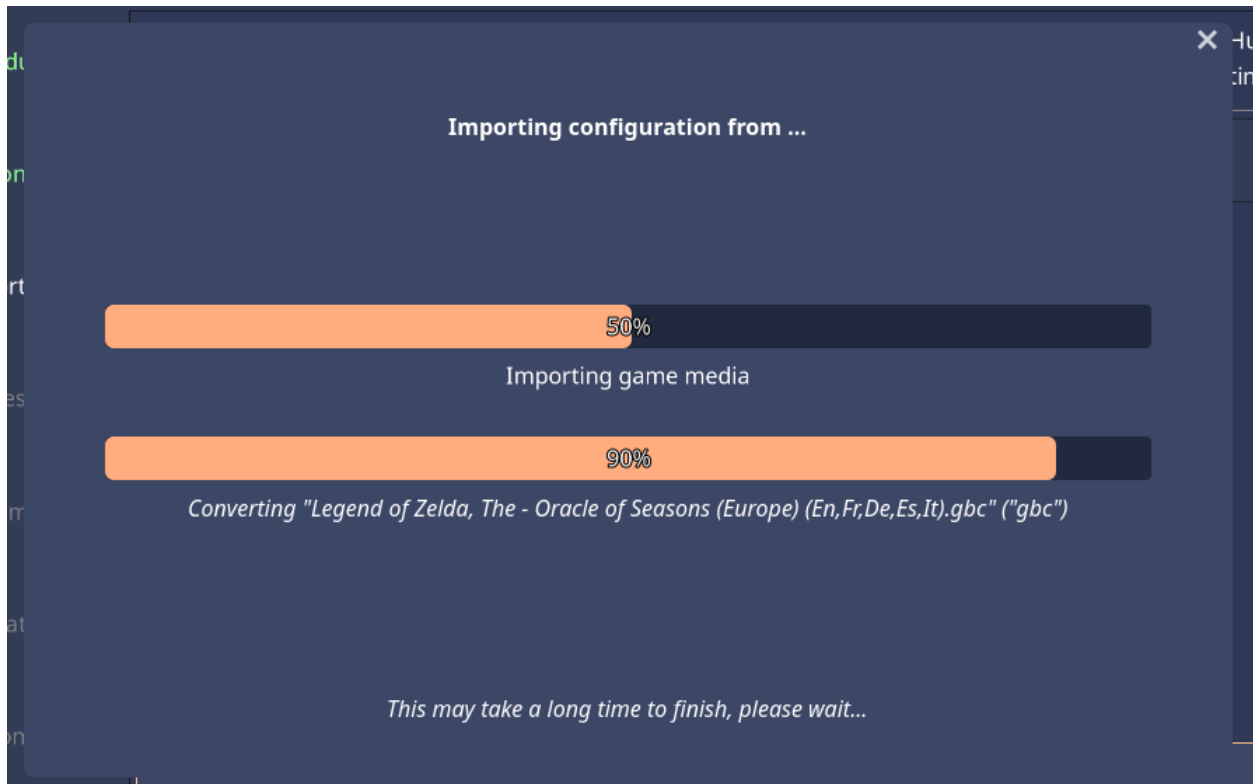
If you pick an app to import, you'll be asked whether you want to move or copy the media files (*game metadata files are always copied*):



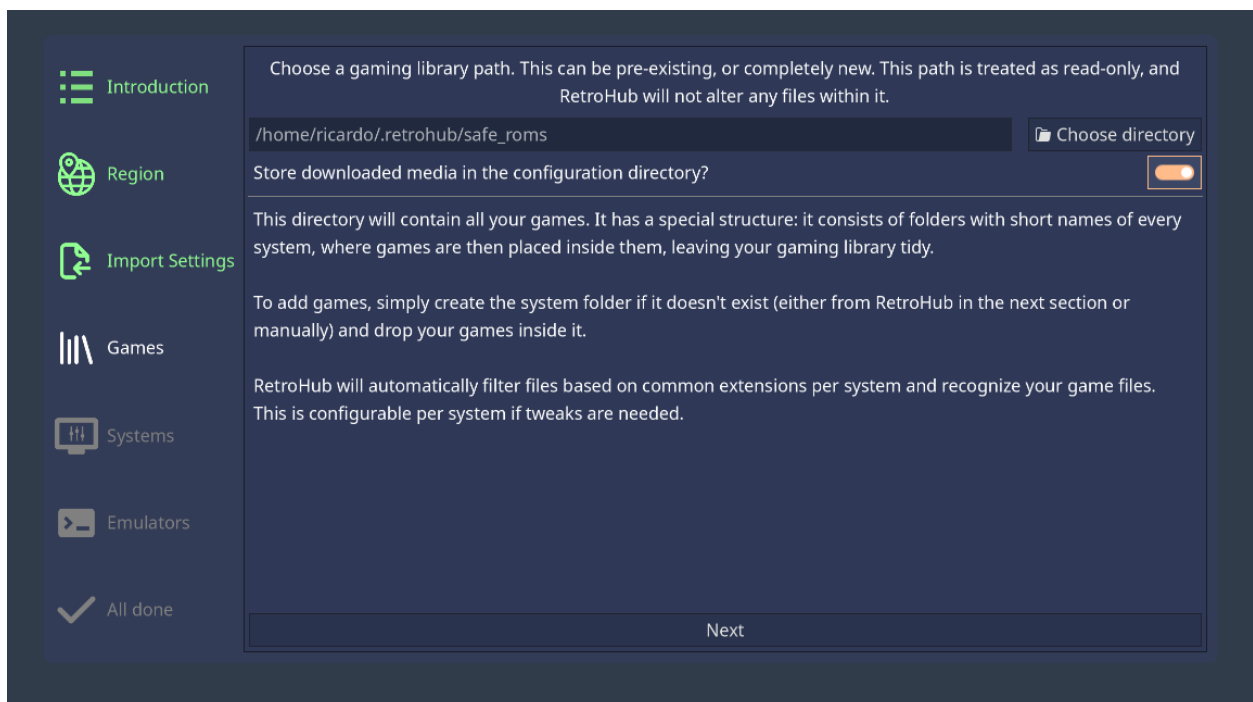
- **Moving** doesn't duplicate files, saving you disk space. However it will likely break your existing setup on the other frontend, requiring you to regenerate or recreate game information if you still intend to use it.
- **Copying** doesn't change any original files, leaving your original setup untouched. However it will consume more disk space, which can be a problem with very large libraries.

Note: Although RetroHub shows an estimate of much space it would take to copy files, always ensure you have more space available before proceeding.

After one of these options, RetroHub will begin importing the files. This step may take some time.



Games



Note: If you imported an existing setup earlier, the game path may already be set to your existing library. If you choose

a different path from what was detected, remember to move your games to the new location.

Here you should pick a folder for storing your games. This can either be empty or an existing one with games already.

This folder will have a defined structure to organize your games, consisting of folders with short system names, with games inside each corresponding folder:

```

games
├── c64
│   ├── game1.bin
│   └── game2.d64
├── gc
│   ├── game1.iso
│   └── game2.iso
├── snes
│   ├── game1.sfc
│   └── game2.sfc
└── ...

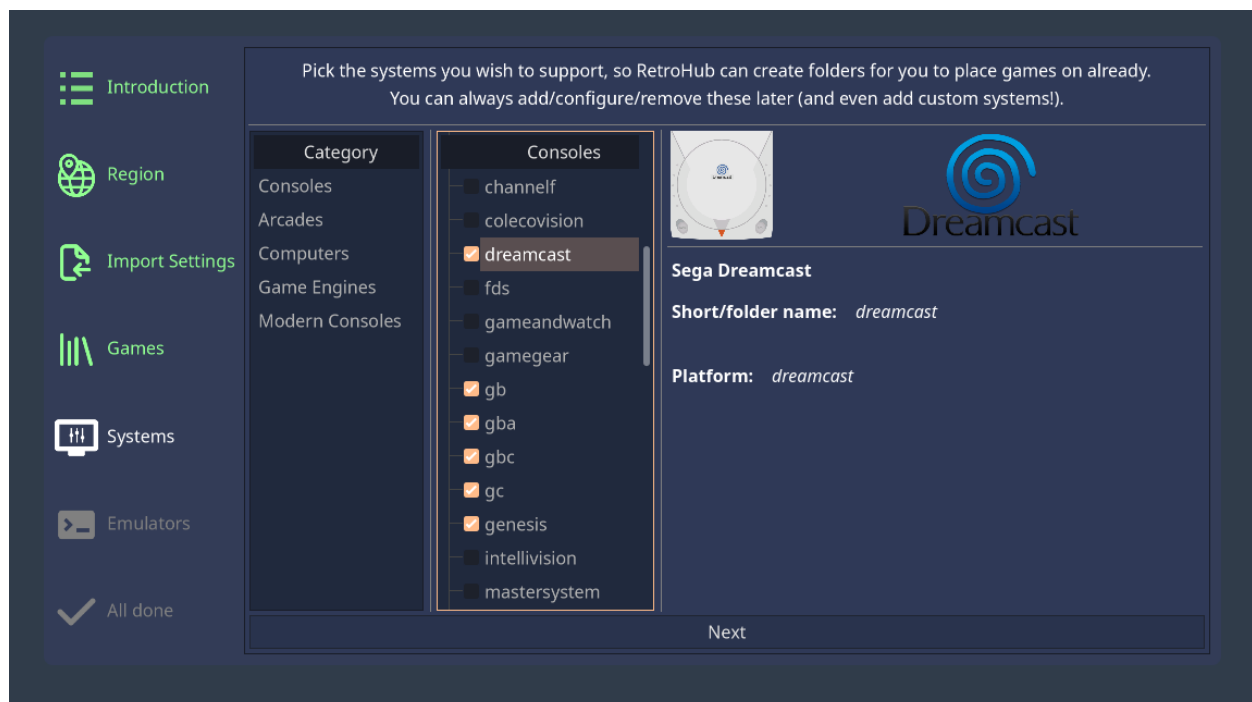
```

The system short names available can be checked at [Systems](#). The next section also allows you to bootstrap some or all system folders.

Each system has a set of file extensions associated, so RetroHub doesn't pick irrelevant files, such as text files. These extensions can also be configured per system later on.

Note: You can also choose to store downloaded game media on a separate directory. If you wish to do so, disable the `Store media in the configuration directory?` option and pick a folder for storing media.

Systems

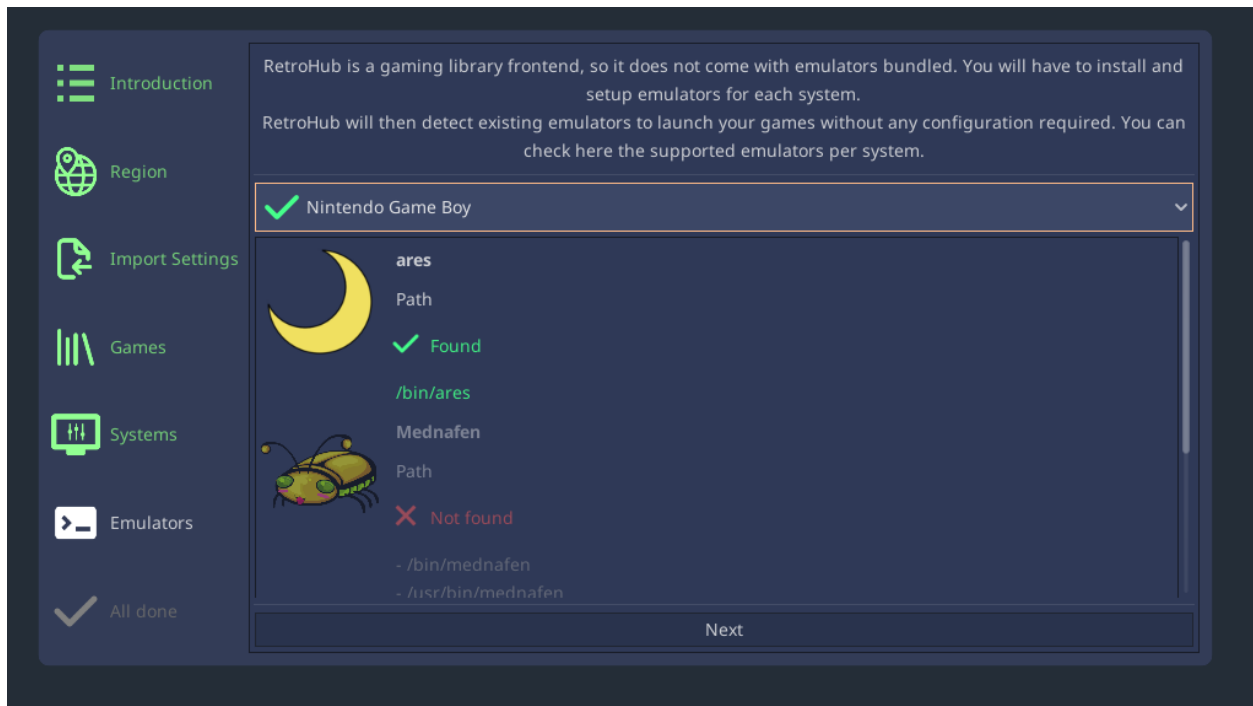


Here you'll find a list of all the systems RetroHub supports by default. They are separated by categories such as game consoles or computers, so you can add only a given kind of systems to your library.

Pick any systems you want to support, and RetroHub will create folders for you to put the games under. If the game library already has some valid system folders, that system will be selected already, and cannot be deselected. Since RetroHub treats your game library as read-only later on, it cannot alter or delete any files. If you wish to remove a system, you'll need to either remove the folder in your library, or remove it from view in the settings.

You can also add custom systems and modify the default ones later on.

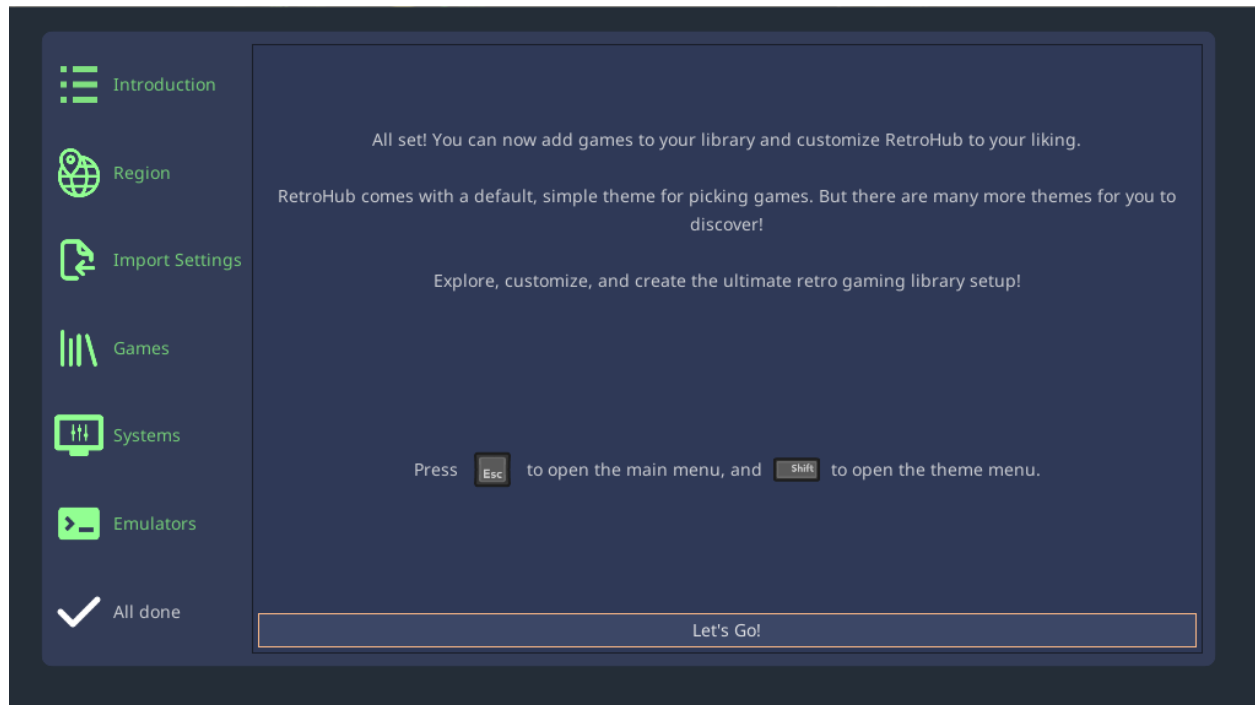
Emulators



This section gives you an overview of what emulators were detected and are thus expected to work right away. RetroHub is an emulator frontend, so it does not come with any pre-installed ones. You'll need to download, install and configure all your desired emulators previously before RetroHub can use them.

If RetroHub fails in finding an emulator, it shows the paths it tried to check on your system for its existence. If your emulator is working as intended, you may need to set its path manually later on.

All done



After this setup, your retro gaming library is configured! RetroHub ships with a default theme for you to start playing right away, but if you want to customize the look and feel of the app, move on to the next section to learn how to download, install and use custom themes.

Themes

Warning: Themes are essentially Godot projects. This means they run custom code, and can therefore be malicious.

Be careful where you download themes from. We recommend downloading only themes from the app and/or trusted sources.

A “Sandbox” mode for Godot is [in discussion](#).

Downloading

There currently aren't any existing community themes to download. This section will be documented once it gets more development.

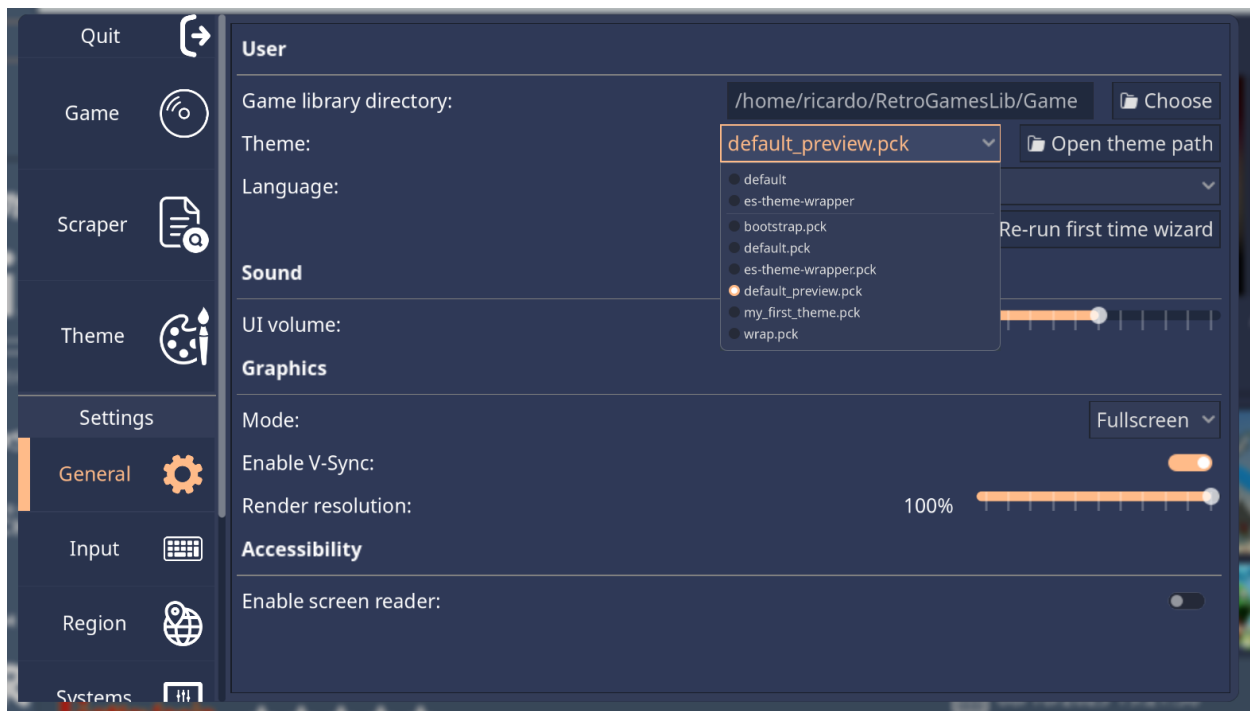
Installing

To install themes, move the theme file (.pck extension) inside the **themes** folder in RetroHub's config folder:

- **Windows:** C:\\Users\\<user_name>\\RetroHub\\themes
- **macOS:** /Users/<user_name>/.retrohub/themes
- **Linux:** /home/<user_name>/.retrohub/themes

You can also open the theme folder from the app's settings.

After that, pick your theme in the **General** settings. Themes above the separator are bundled by default, while the ones below it were manually downloaded.



All done

With this, you're ready to enjoy your retro gaming library! There's more pages with information about the app, so feel free to explore them.

If you have any questions, you can contact us on [Discord](#) or [Reddit](#).

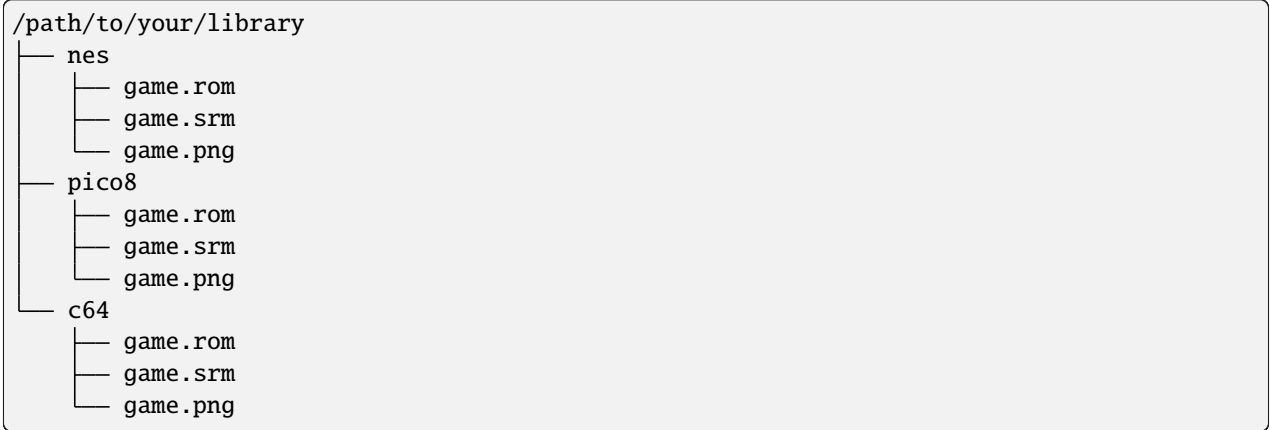
1.2 Game metadata

In this page, the metadata format is described, as well as a general description of how RetroHub handles your gaming library.

1.2.1 Gaming library setup

RetroHub treats your gaming library as a read-only directory, to avoid potential troubles. Other than creating these folders for new systems, RetroHub cannot edit these game files, nor delete system folders.

The directory structure is as follows: a set of folders with short system names, with the games for the system placed inside it.



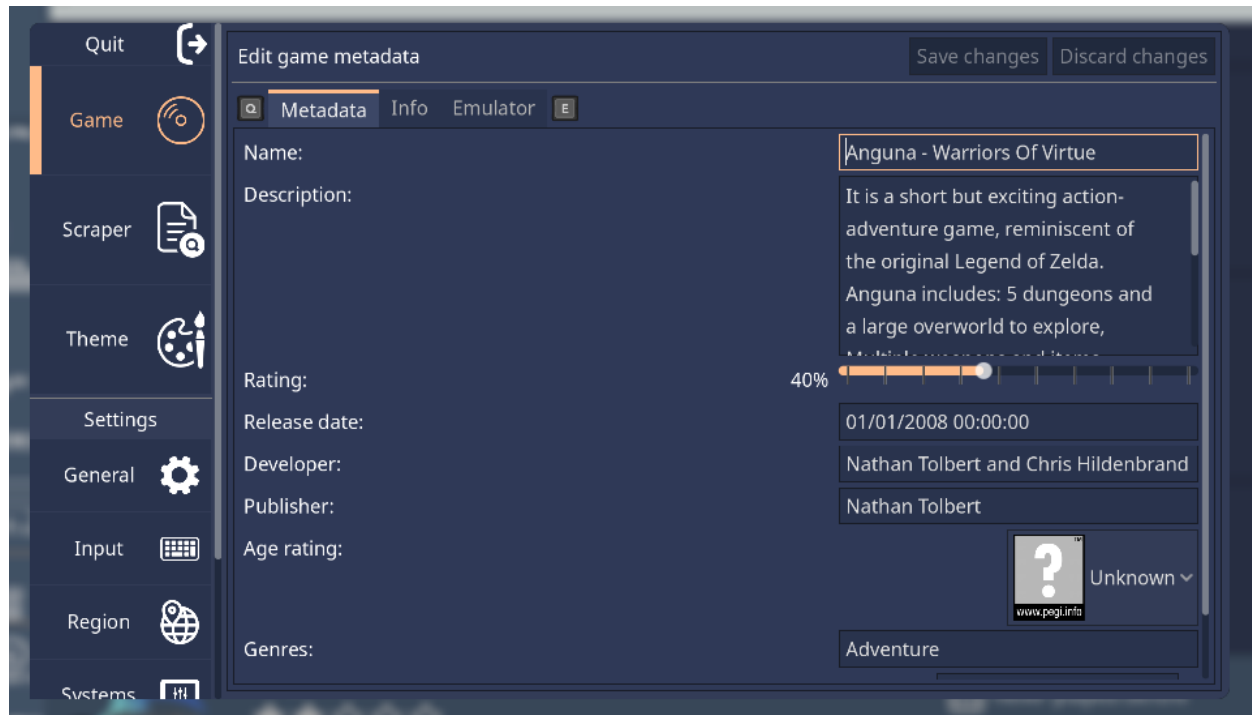
For that reason, RetroHub stores configuration files and metadata information in another directory:

- **Windows:** C:\Users\<<username>\RetroHub
- **macOS:** /Users/<username>/.retrohub
- **Linux:** /home/<username>/.retrohub

1.2.2 Adding metadata

When RetroHub finds a valid game file, it starts without any metadata. Only the name will be set to the file name. You can then populate a lot of information for the game, such as the name, description, release date, developer, publisher, genre, etc.

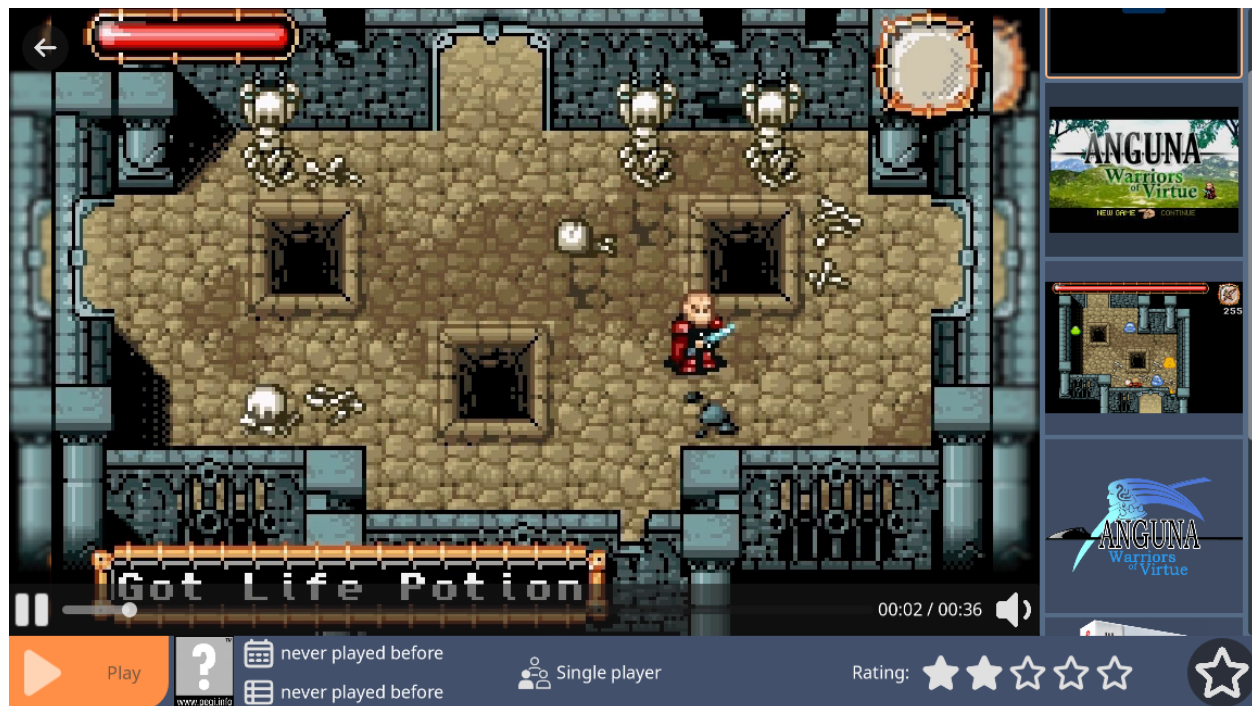
To do this, select a given game, then open the **Settings** ( / ) panel. The **Game** tab should be selected by default; if not, select the **Game** tab from the left.



Here you can edit your game's metadata. Don't forget to save changes when you're done.

1.2.3 Adding media

Game entries may also have media to show. This can be a screenshot, a logo, a short gameplay video, a box texture, etc...



Due to the nature of media files, this has to be done outside of RetroHub. Suppose you have the game My Game

[!] .rom, in the snes folder. To add media to it, you'll need to:

- Go to RetroHub's configuration directory (check above for the location)
- Go to the gamemedia folder
- Enter the snes folder (if it doesn't exist yet, create it)
- Create the appropriate media folder (see below)
- Add the media file, renamed to the game's name (for example, My Game [!] .png)

The available media files are the following:

Media	Folder name	File extension	Description
Logo	logo	.png / .jpg	The game's logo (title art)
Screenshot	screenshot	.png / .jpg	A screenshot of the game
Title Screen	title-screen	.png / .jpg	A screenshot of the game's title screen
Video	video	.mp4	A short gameplay video
Box Render	box-render	.png / .jpg	A picture of the game's box
Box Texture	box-texture	.png / .jpg	The game's box as a texture, to be used for a 3D model
Support Render	support-render	.png / .jpg	A picture of the game's physical support (cartridge, CD, etc...)
Support Texture	support-texture	.png / .jpg	The game's physical support as a texture, to be used for a 3D model
Manual	manual	.pdf	The game's manual

Warning: Currently, RetroHub cannot display manual files.

1.2.4 Filling game information automatically

Setting up metadata and media for your entire library is very time consuming. To accelerate this process, RetroHub can use scraper services that automatically fill most, if not all, information for you. Check the [Scraping](#) page for more information on how to use them.



1.3 Scraping

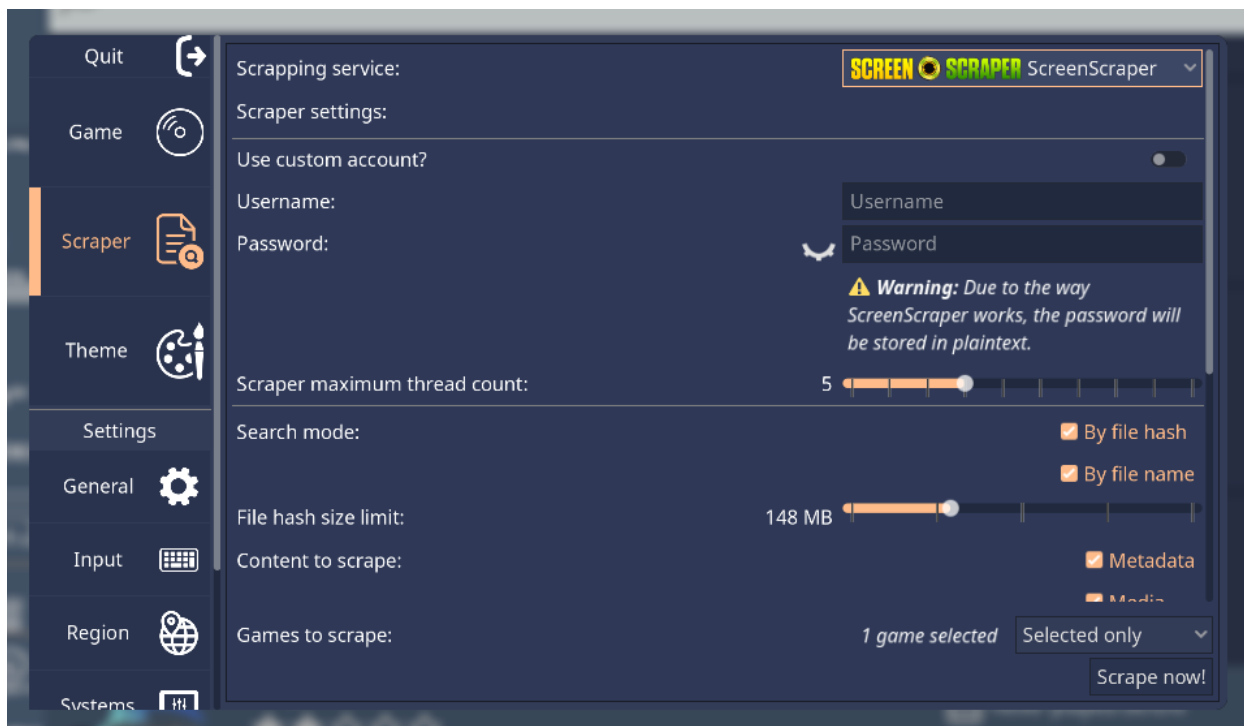
A scraper is a tool that can automatically fetch a game's metadata and download associated media. RetroHub supports the following scrapers:

- [ScreenScraper](#)

1.3.1 Usage

Note: It's recommended to use keyboard and mouse during the scraping process, as you might need plenty of text input and navigating through lists.

You can scrape one, multiple, or even your entire library of games. Open the **Settings** ( / ) menu and select the **Scraper** tab.



Select the desired scraping service. Different scrapers will have different levels of *Search Support*, *Metadata Support* and *Media Support*.

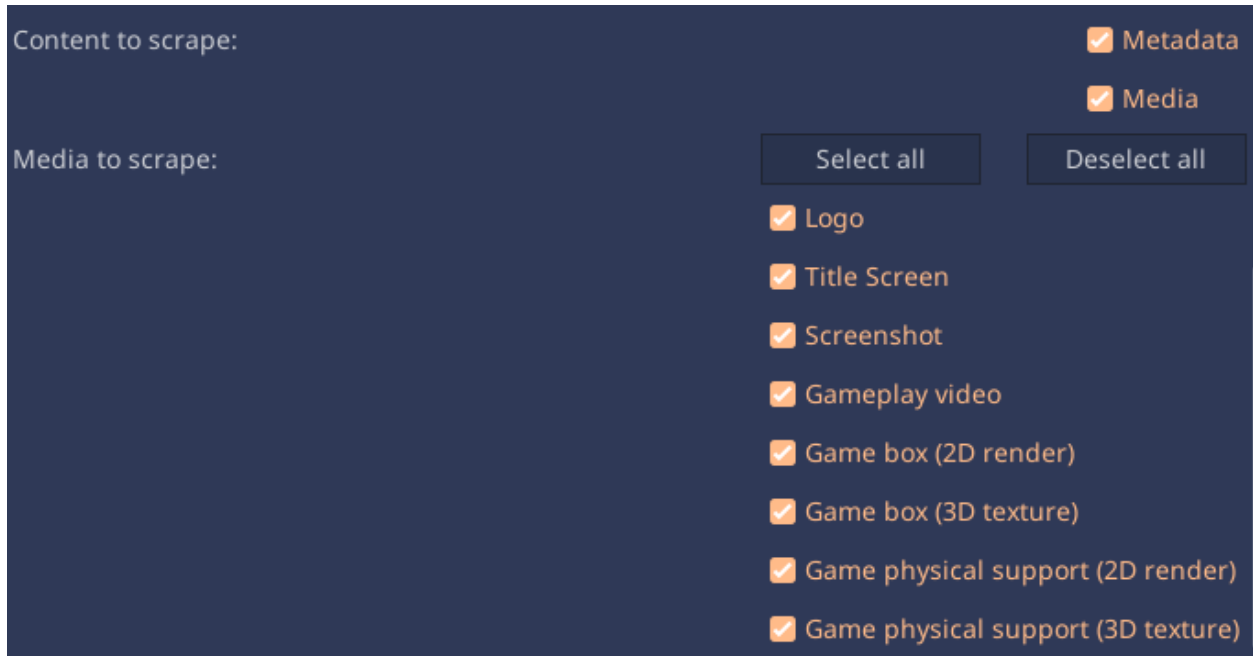
Note: We recommend using ScreenScraper as it is the most complete scraper right now.



Check how to search for your games:

- **By file hash:** Computes the MD5 hash of your game files and uses it to uniquely identify it on the scraper's database. This is the most reliable way to detect your games. If this doesn't work (either because the file is too big, or the scraper doesn't know that hash), it will fallback to search by name if enabled.
- **By file name:** Searches for your game using the game's file name, minus the extension. RetroHub will ask you to choose between multiple results if available, and also allow you to refine the search term.

- **File hash size limit:** When searching by hash is enabled, this option allows you to set the maximum file size (in MB/GB) allowed. Files bigger than this will not be hashed, and will fallback to search by name if enabled.



Content to scrape: ☒ Metadata ☒ Media

Media to scrape:

- ☒ Logo
- ☒ Title Screen
- ☒ Screenshot
- ☒ Gameplay video
- ☒ Game box (2D render)
- ☒ Game box (3D texture)
- ☒ Game physical support (2D render)
- ☒ Game physical support (3D texture)

Check what kind of information you want to scrape:

- **Metadata:** Game textual information (name, description, release date, etc.)
- **Media:** Game media (box art, screenshots, videos, etc.)

Then check each type of media you wish to scrape.



Games to scrape: *1 game selected*

Lastly, select what games you want to scrape, and start the process by clicking the **Scrape now!** button.

Depending on the amount of games, types of media, and your internet connection, this process may take quite some time to complete. A new popup will appear where you can accompany the progress for each game, as well as fix problems found for some games (such as game title not found, multiple results available, etc...)



Warning: For large game libraries, the media files can start to take up a lot of space (>1GB). Ensure you have enough free space on your hard drive before scraping, as this may fill your entire disk space.

1.3.2 ScreenScrapper

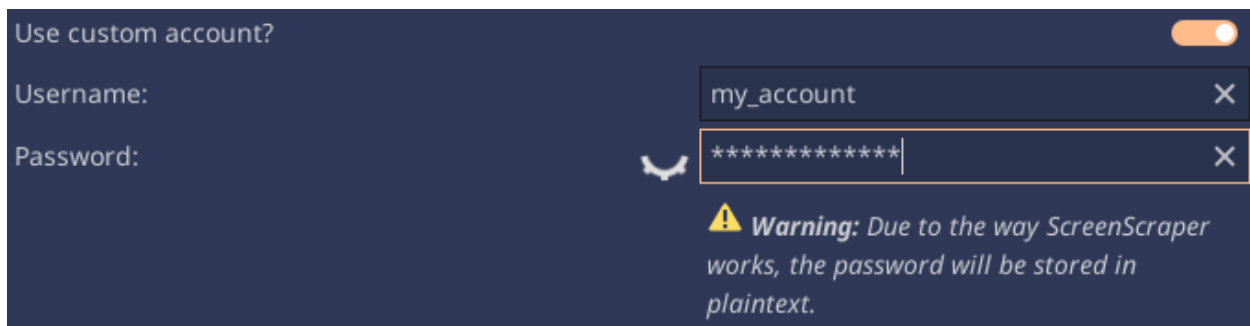
Region

ScreenScrapper will return results according to your currently set region, and not the game's region.

API quota

ScreenScrapper has both a daily and hourly quota for API requests. If this limit is reached, RetroHub will start showing errors when scraping.

To circumvent this, you should make a free account on their website at <https://www.screenscraper.fr/> and enter your credentials in the **Scraper** settings:



This will scrape with your personal quota instead. If that runs out as well, you'll have to wait until the next day/hour to scrape again.

Thread usage

By default, RetroHub will scrape with 2 threads. If your custom account has access to more threads, RetroHub will use all of them. To limit how many threads are used, you can set it in the **Scraper** settings:



1.3.3 Search Support

Search	ScreenScraper
File hash	yes
File name	yes

1.3.4 Metadata Support

Metadata	ScreenScraper
Name	yes
Description	yes
Rating	yes
Release Date	yes
Developer	yes
Publisher	yes
Genres	yes
Number of players	yes
Age Rating	yes

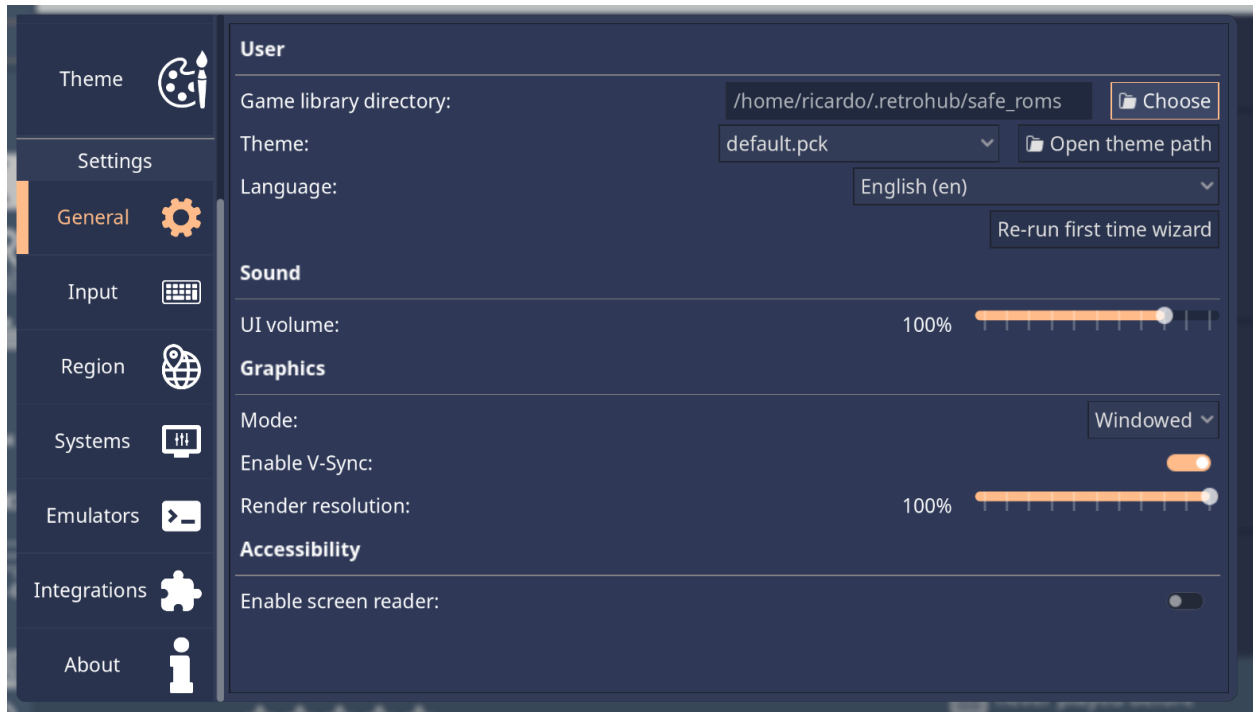
1.3.5 Media Support

Media	ScreenScraper
Logo	yes
Screenshot	yes
Title Screen	yes
Video	yes
Box Render	yes
Box Texture	yes
Support Render	yes
Support Texture	yes
Manual	yes

1.4 Configuration

While RetroHub is designed with minimal configuration in mind, there's plenty of options available to further customize it to your liking. This page thus covers all available configurations.

1.4.1 General



User

- **Game library directory:** The directory where your games are stored.

Warning: If you change this, RetroHub will not move the existing games to the new directory. You will have to do that manually.

- **Theme:** Currently loaded theme
- **Language:** App language
- **Re-run first time wizard:** Re-run the first time wizard

Sound

- **UI Volume:** Volume for UI sounds

Graphics

- **Mode:** Windowed or fullscreen
- **Enable V-Sync:** Enable V-Sync
- **Render resolution:** Resolution to render the theme at. Main UI is not affected.

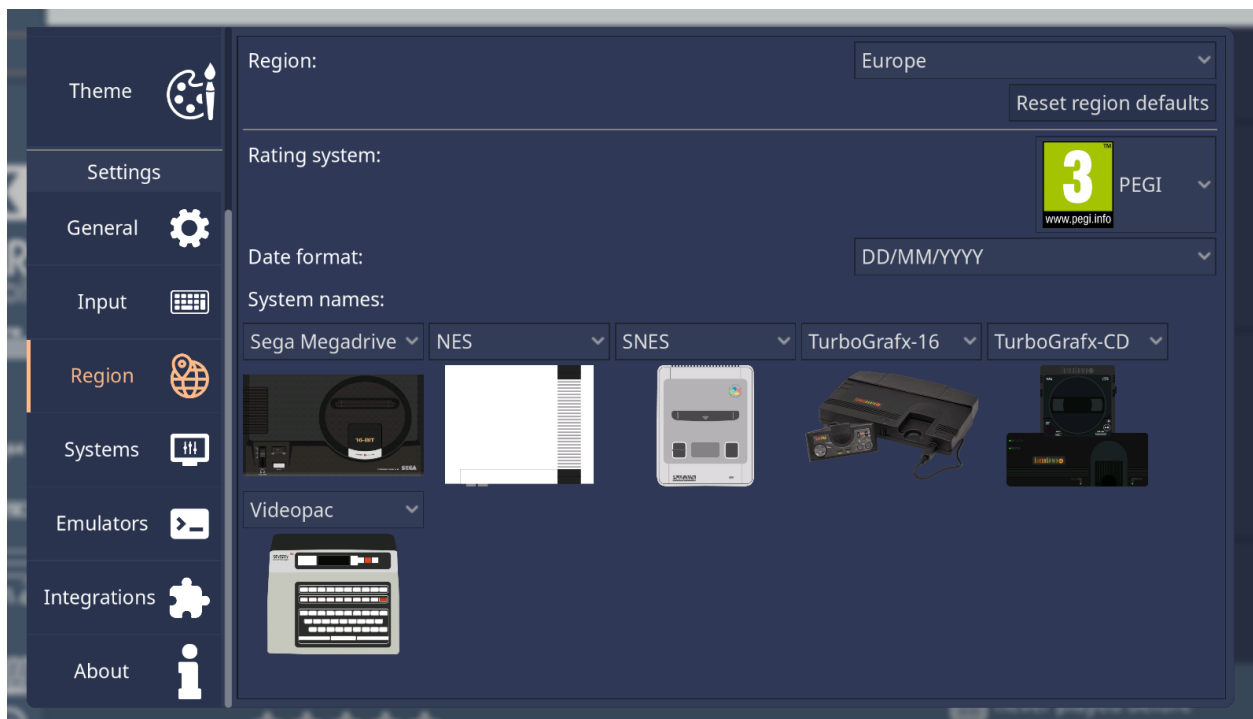
Accessibility

- **Enable screen reader:** Enables a screen reader to read the interface out loud.

1.4.2 Input

Check the dedicated *Input Methods* page.

1.4.3 Region



Note: Region settings are used to customize how RetroHub displays different types of information, and has no impact whatsoever on what games can be played.

- **Region:** User region

- **Reset region defaults:** Resets the other settings to the default according to the selected region.
- **Rating system:** Rating system to use for games: ESRB, PEGI or CERO.
- **Date Format:** Date format to present dates in.
- **System Names:** Preferred system names for systems with different names per region.

Note: If you have games on two system folders that are the same but have different names per region (e.g. `genesis` and `megadrive`), RetroHub will combine both games and show them under your preferred system name. Likewise, if you change the region system name, you don't need to move files in your library.

1.4.4 Systems

Check the dedicated [Systems](#) page.

1.4.5 Emulators

Check the dedicated [Emulators](#) page.

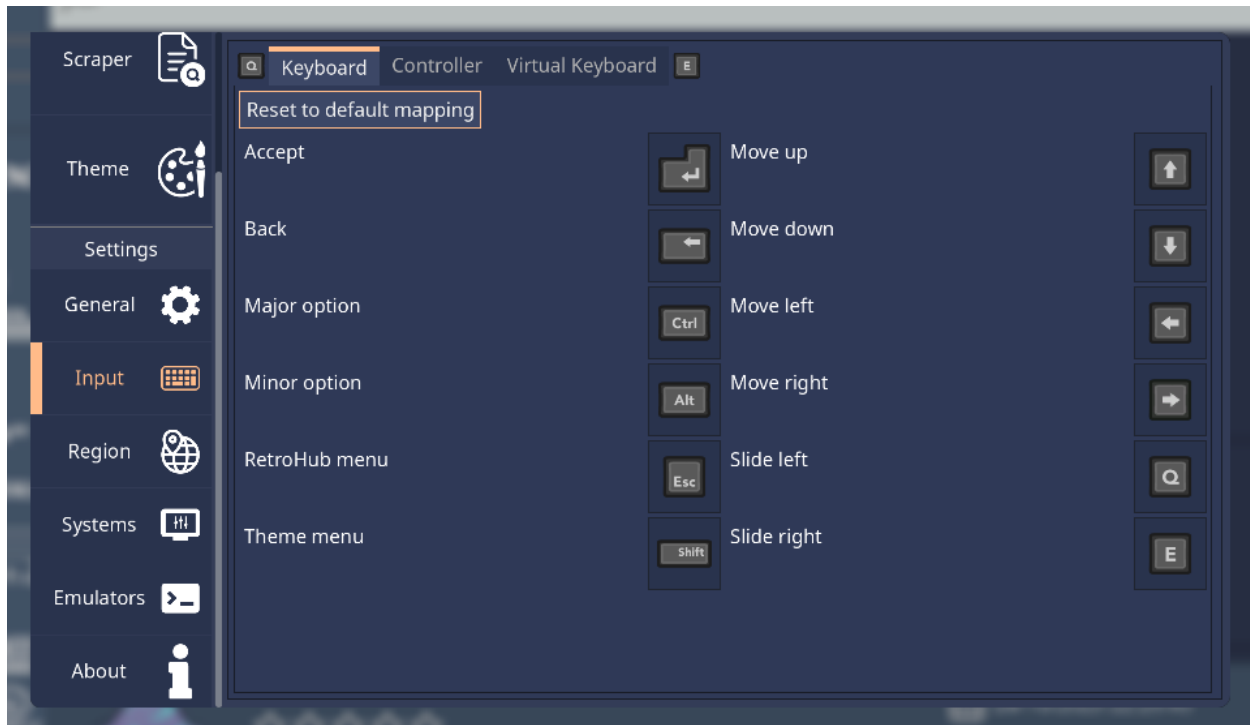
1.4.6 Integrations

Check the dedicated [Integrations](#) page.

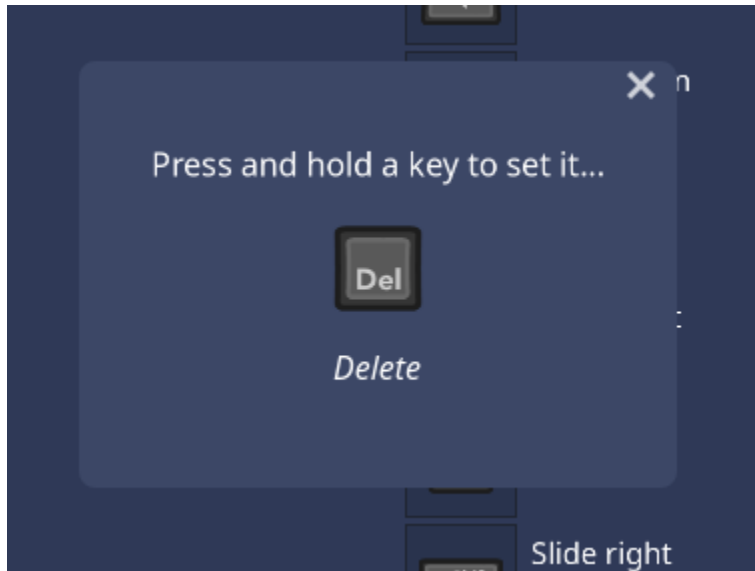
1.5 Input Methods

RetroHub has full support for keyboard, mouse and controller input. You can use one or all input methods seamlessly throughout the app.

1.5.1 Keyboard



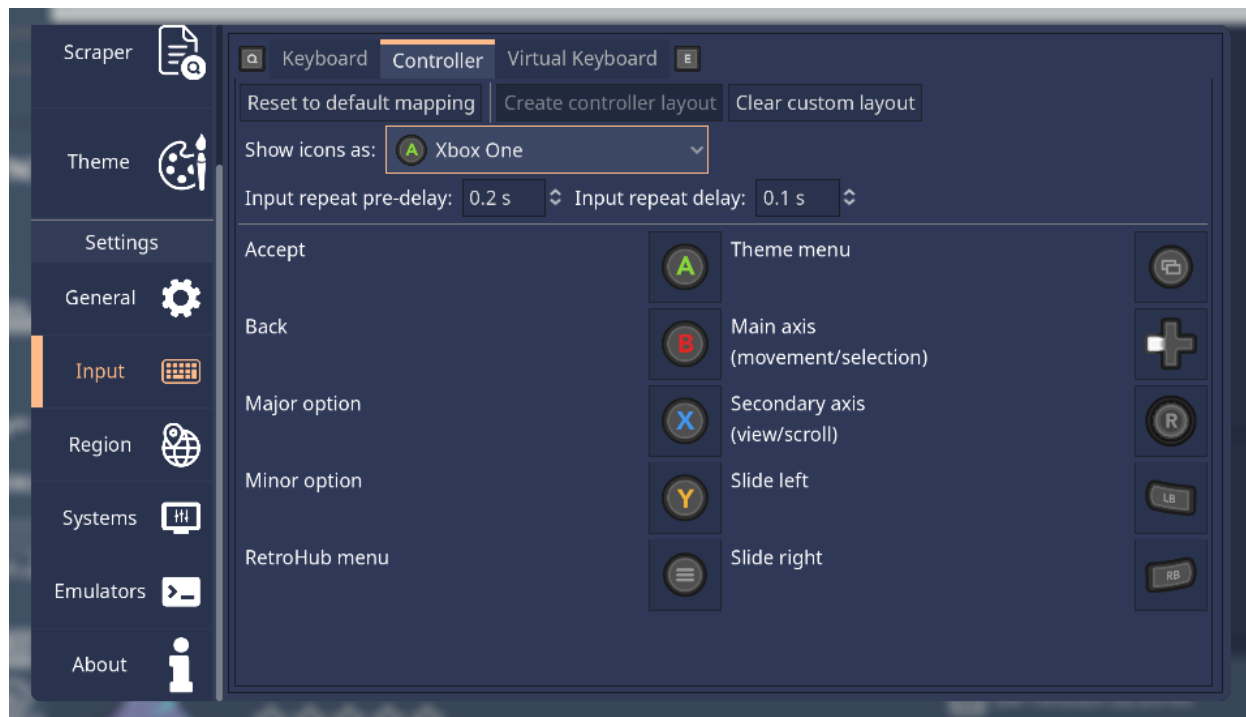
You can remap actions to different keys. Click on the desired action, then press the new key to use on that action.



Note: It's normal if the displayed icon and text don't match. That's because RetroHub uses the "physical" key location on the keyboard, independent of your keyboard layout.

Note: You can't use the same key for more than one action. If you try to remap a key to an existing action, they will be swapped.

1.5.2 Controller



RetroHub should detect your controller automatically once it's plugged in. If detected successfully, it will also show appropriate icons for your controller model.

You can remap buttons and axis to other alternatives. Click on the desired action, then choose the new button/axis to replace to. If that button/axis is used on another action, they are swapped.



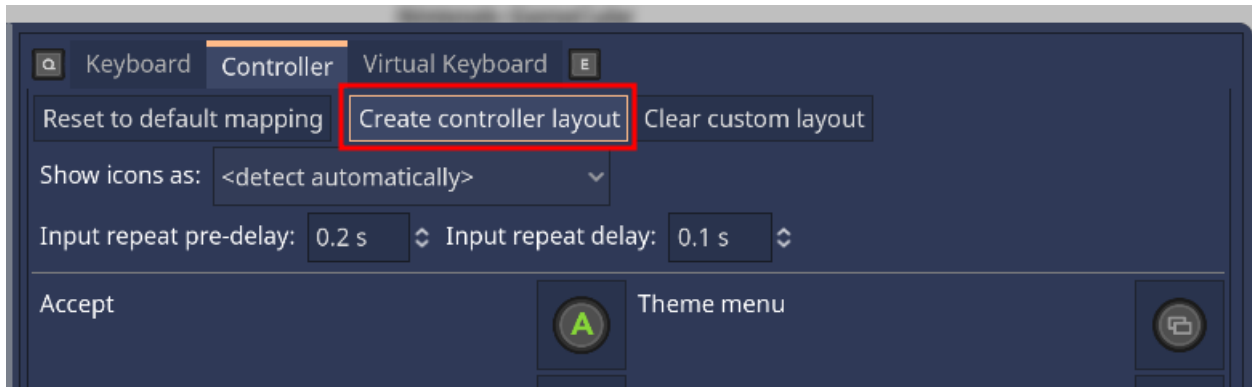
Beyond remapping, there are some more settings for customizing the controller's behavior:

- **Create controller layout:** Create a custom layout for your controller if it's not behaving correctly. See [Controller Layouts](#) for more details.
- **Clear custom layout:** Clear the custom layout for your controller. This will revert to the default layout.
- **Show icons as:** Choose what icons to display for your controller. If automatic detection doesn't work, you can manually set your controller type here.
- **Input repeat pre-delay:** The delay before the button/axis starts repeating when held down.

- **Input repeat delay:** The delay between each repeat event when held down.

Controller Layouts

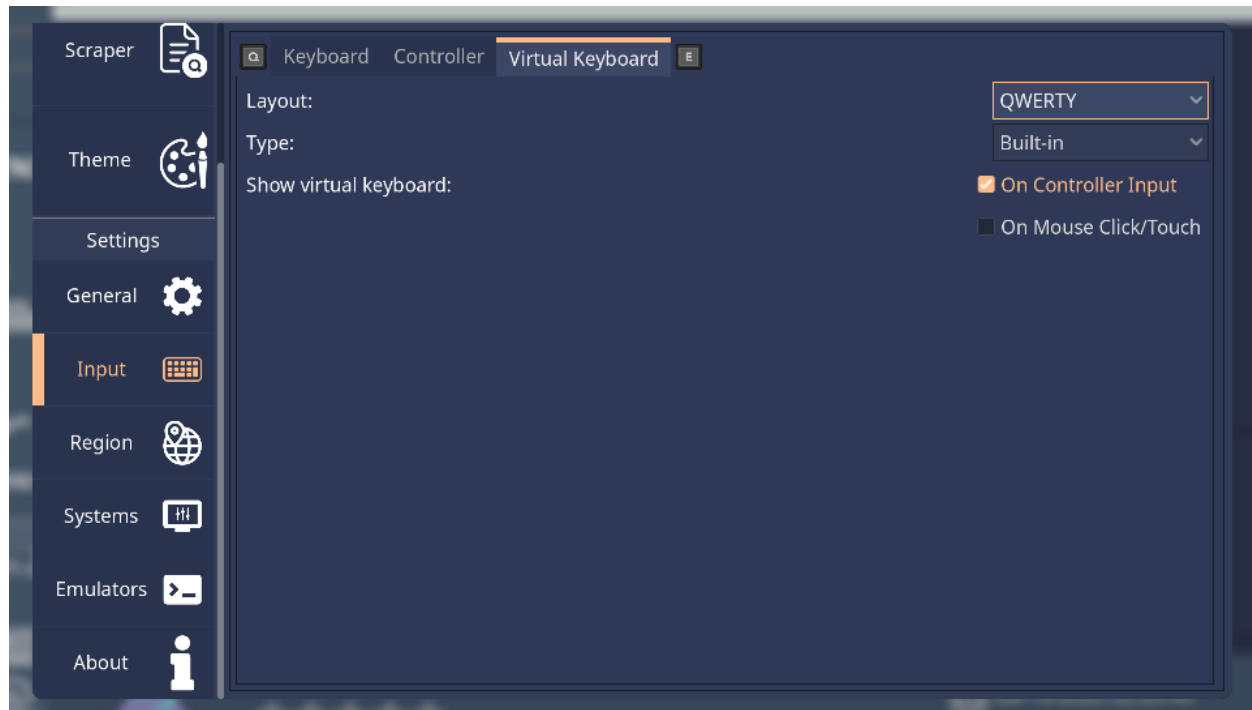
If your controller is not behaving properly (such as buttons not doing anything or doing the wrong action), RetroHub is using a wrong layout for your controller. In that case, you must create a controller layout for RetroHub to understand your controller inputs.




This will bring up a tool to map your controller buttons. Click or move on the highlighted buttons/axis as instructed to map it. If you do not have the specified button or if it doesn't work on your controller, you can skip it by clicking on the "Skip" button.



1.5.3 Virtual Keyboard



RetroHub comes with a virtual keyboard to input text when a physical keyboard isn't available.

- **Layout:** Virtual keyboard layout
- **Show virtual keyboard:**
 - **On Controller Input:** Show the virtual keyboard when a controller is used, and a text field is selected with .
 - **On Mouse Click/Touch:** Show the virtual keyboard when there's a mouse click or touch on a text field.

1.6 Systems



A system is a group of games that run under the same or similar hardware.

1.6.1 Adding custom systems

Warning: Themes will receive your custom system, but will likely not “know” your system, so it might be themed generically or even not show up at all.

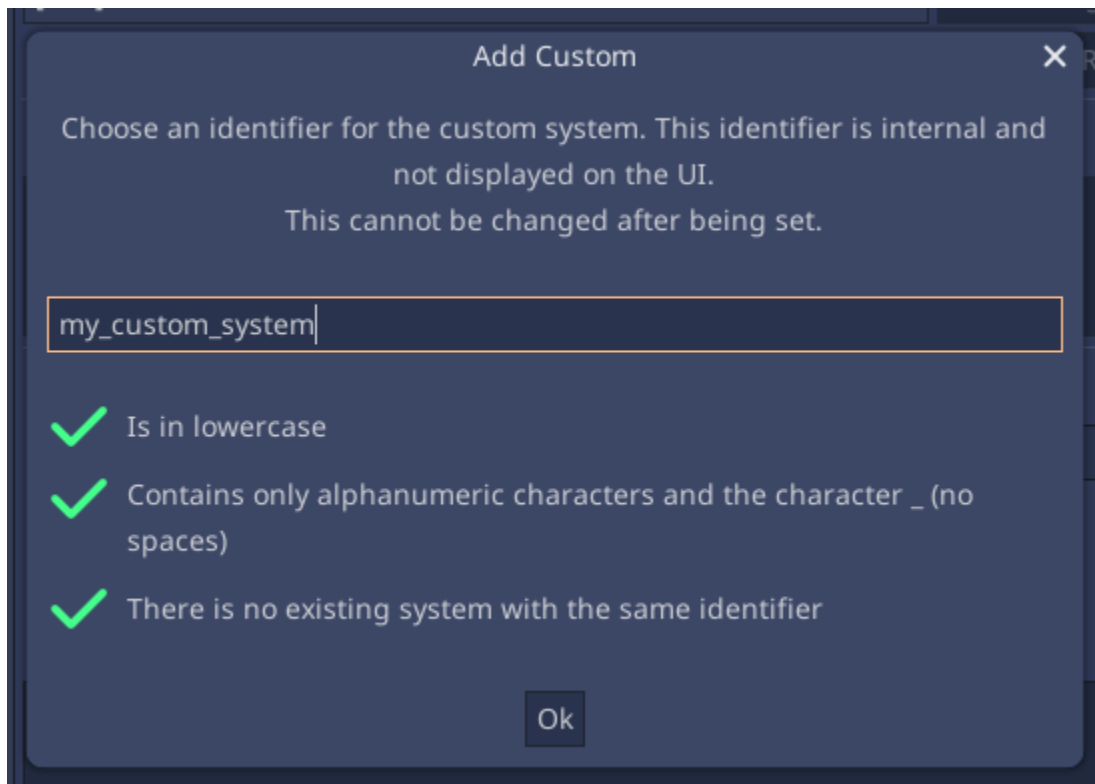
There are two ways to achieve this:

Application

Open the **Settings** ( / ) panel, and navigate to the **Systems** section. Click on the **Add custom system**.



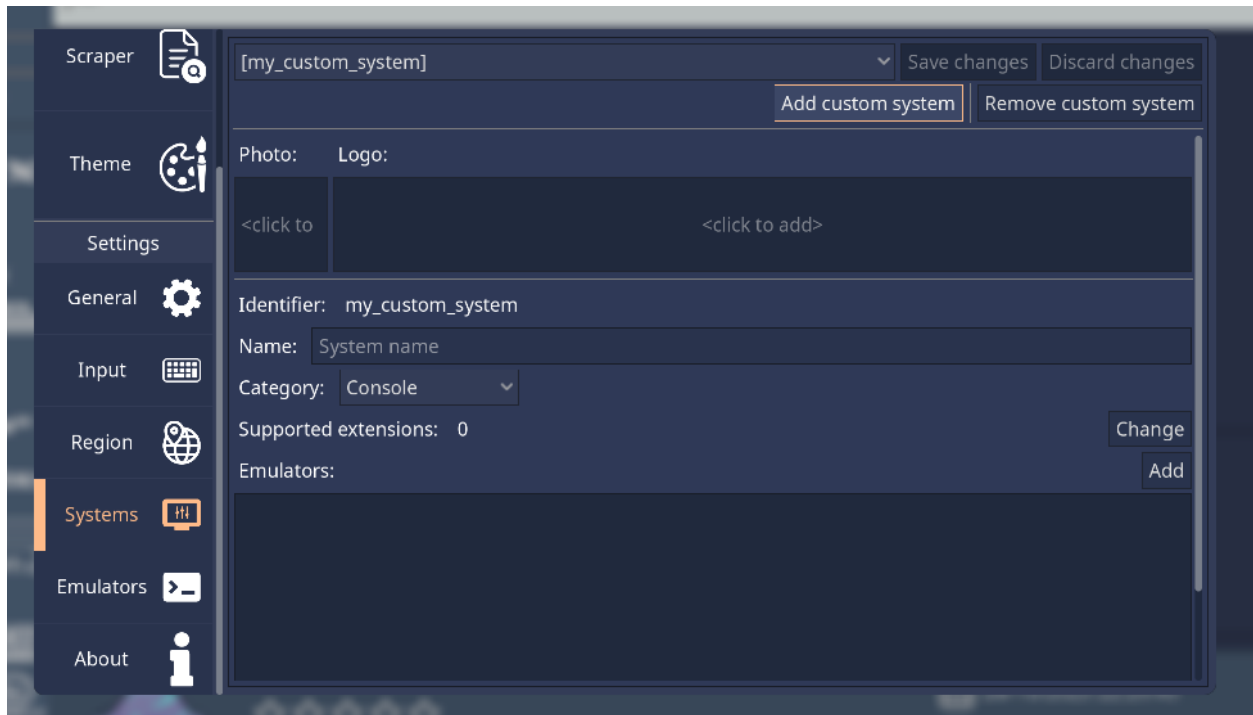
This will bring a popup to choose a short system name. This is what's used internally to uniquely identify the system, and to create the system folder in your game library.



It must follow the following rules to be valid:

- Be lowercase
- Use only alphanumeric characters (a-z and 0-9) and the underscore character
- Not be an existing system short name

After that you must now set all the information relevant to that system in each available field, such as it's name, accepted game file extensions, and emulators to be used.



Manually

Edit the `rh_systems.json` file in the configuration directory. Add a dictionary entry to the existing array, with the system information:



```
[
  {
    "name": "my_custom_system",
    "platform": "my_custom_system",
    "fullname": "My Custom System",
    "extension": [
      ".abc",
      ".rom"
    ],
    "emulator": [
      "my_custom_emulator",
      {
        "retroarch": [
          "my_custom_core"
        ]
      }
    ],
    "category": "console"
  }
]
```

For more information on the necessary keys and their values, see the *rh_systems.json specification* section.

1.6.2 Changing system settings

There are two ways to achieve this:

Application

Open the **Settings** ( / ) panel, and navigate to the **Systems** section.



Then modify these fields to your liking. Remember to save changes when you're done.

You can also revert to the default settings by clicking the **Restore default system** button.

Manually

Edit the `rh_systems.json` file in the configuration directory. Add a dictionary entry to the existing array, with the `name` field set to the name of the system you want to modify, and then changing the other key values:

```
[
  {
    "name": "n64",
    "fullname": "Nintendo 64 (Custom Name)",
    "extension": [
      ".abc",
      ".rom"
    ]
    "category": "computer"
  }
]
```

For more information on the existing keys and their values, see the [rh_systems.json specification](#) section.

1.6.3 Removing default systems

To remove a default system, you can remove or rename that system's folder in your gaming library. However, if you do not want to do that, you can also "hide" it from the app instead.

There are two ways to achieve this:

Application

Note: This is still not implemented in the app. For now you must follow instructions on the **Manually** section.

Manually

Edit the `rh_systems.json` file in the configuration directory. Add a dictionary entry to the existing array, with the `name` field set to the name of the system you want to remove, and then add the special `#delete` key to it:

```
[
  {
    "name": "n64",
    "#delete": true
  }
]
```

For more information, see the [rh_systems.json specification](#) section.

1.6.4 rh_systems.json specification

This page details the JSON specification of the systems file, `rh_systems.json`. This file is applied on top of the default systems specification file, which can be checked [here](#).

Warning: Always close RetroHub before editing this file, otherwise your changes may be overridden!

Warning: Ensure the file is valid before using it. You can use a JSON validator such as [JSONLint](#).

This JSON consists of an array of dictionary entries. Each entry details a custom system or modifications to an existing system:

```
[
  {
    "name": "psx",
    "extension": [
      ".bin",
      ".cue"
    ]
  }
]
```

(continues on next page)

(continued from previous page)

```

    },
    {
        "name": "amiga",
        "#delete": true
    },
    {
        "name": "my_custom_system",
        "platform": "my_custom_system",
        "fullname": "My Custom System",
        "extends": "ps4",
        "extension": [
            ".abc",
            ".rom"
        ],
        "emulator": [
            "my_custom_emulator",
            {
                "retroarch": [
                    "my_custom_core"
                ]
            }
        ],
        "category": "console"
    }
]

```

Each entry has specific keys for system information, detailed below:

- name - String (**required**)

Short name for the system

```
"name": "n64"
```

- platform - String (**required**)

To what platform this system belongs, if applicable. This is for systems that are functionally the same (such as using the same emulators), but are different enough to be displayed separately for users (*for example, Nintendo 64 DD on Nintendo 64, or Sega Genesis CD on Sega Genesis*).

If this isn't the case, set it to the same value as **name**.

```
"name": "n64dd",
"platform": "n64"
```

- fullname - String (**required**)

System name displayed to the user

```
"fullname": "Nintendo 64"
```

- extension - Array (**required**)

Accepted file extensions. These are case insensitive, so `.bin` and `.BIN` are the same, for example.


```
"extension": [
    ".n64",
    ".v64",
    ".z64",
    ".zip"
]
```

- **emulator** - Array (**required**)

Valid emulators to use for this system. It is composed of either:

- A String for simple emulators

```
"emulator": [
    "mupen64plus"
]
```

- **A Dictionary for complex emulators:**

– RetroArch (key `retroarch`): array of strings of accepted libretro cores

```
"emulator": [
    {
        "retroarch": [
            "mupen64plus",
            "parallel_n64"
        ]
    }
]
```

- **category** - String (**required**)

System category. Must be console, computer, arcade or modern_console

```
"category": "console"
```

- **extends** - String (*optional*)

Copies the configuration of an existing system. With this, you can omit all the other keys (except `name`) as they will be copied from the system you are extending. You can still specify keys, which will override the copied values.

```
"name": "n64dd",
"fullname": "Nintendo 64 DD",
"extends": "n64"
```

- **#delete** - Boolean (*optional*)

If this key exists, RetroHub will delete this system, making it unavailable to the user. Must be combined with a `name` key to know which system to delete.

```
"name": "n64dd",
"#delete": true
```

1.6.5 System list



AMIGA

Commodore Amiga

Basic Information

- **Short name:** amiga
- **Type:** Computer
- **Supported extensions: 19**
 - .adf
 - .adz
 - .dms
 - .fdi
 - .ipf
 - .hdf
 - .hdz
 - .lha
 - .slave
 - .info
 - .cue
 - .ccd

- .chd
- .nrg
- .mds
- .iso
- .uae
- .m3u
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



AMIGA

Commodore Amiga 600

Basic Information

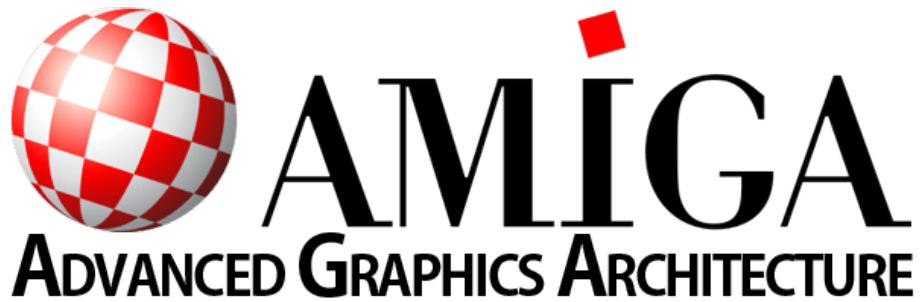
- **Short name:** amiga600
- **Type:** Computer
- **Supported extensions:** 19
 - .adf
 - .adz
 - .dms
 - .fdi
 - .ipf
 - .hdf
 - .hdz
 - .lha
 - .slave
 - .info
 - .cue
 - .ccd
 - .chd
 - .nrg
 - .mds
 - .iso
 - .uae
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Commodore Amiga 1200

Basic Information

- **Short name:** amiga1200
- **Type:** Computer
- **Supported extensions:** 19
 - .adf
 - .adz
 - .dms
 - .fdi
 - .ipf
 - .hdf
 - .hdz
 - .lha
 - .slave
 - .info
 - .cue
 - .ccd

- .chd
- .nrg
- .mds
- .iso
- .uae
- .m3u
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



AMIGA CD32

Commodore Amiga CD32

Basic Information

- **Short name:** amigacd32
- **Type:** Console
- **Supported extensions:** 19
 - .adf

- .adz
- .dms
- .fdi
- .ipf
- .hdf
- .hdz
- .lha
- .slave
- .info
- .cue
- .ccd
- .chd
- .nrg
- .mds
- .iso
- .uae
- .m3u
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*





Amstrad CPC

Basic Information

- **Short name:** amstradcpc
- **Type:** Computer
- **Supported extensions: 10**
 - .cdt
 - .cpr
 - .cpc
 - .dsk
 - .kcr
 - .m3u
 - .sna
 - .tap
 - .voc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Caprice32*
- *Arnold*
- *MAME*



Apple II

Basic Information

- **Short name:** apple2
- **Type:** Computer
- **Supported extensions:** 3
 - .dsk
 - .nib
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mednafen*
- *MAME*



Atari 800

Basic Information

- **Short name:** atari800
- **Type:** Computer
- **Supported extensions:** 14
 - .xfd
 - .atr
 - .atr.gz
 - .xfd.gz
 - .atx

- .cdm
- .cas
- .car
- .bas
- .bin
- .dcm
- .a52
- .xex
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Atari800*
- *Altirra*
- *MAME*



Atari ST

Basic Information

- **Short name:** atarist
- **Type:** Computer
- **Supported extensions: 11**
 - .st
 - .msa
 - .stx
 - .dim
 - .ipf
 - .img
 - .rom
 - .raw
 - .ctr
 - .m3u
 - .zip

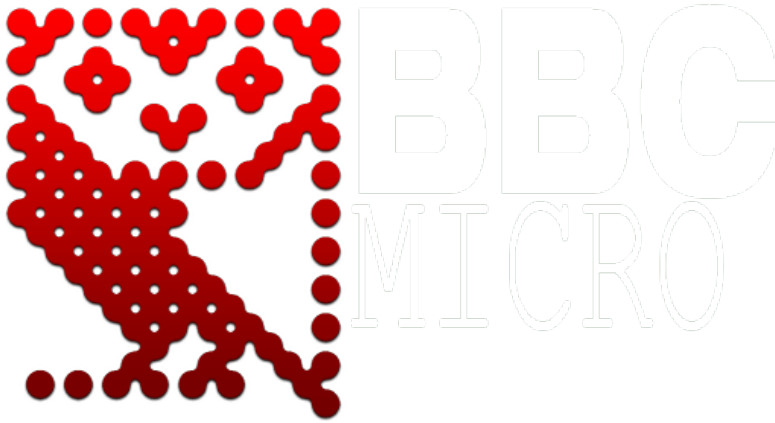
Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Hatari*





BBC Micro

Basic Information

- **Short name:** bbcmicro
- **Type:** Computer
- **Supported extensions: 9**
 - .ssd
 - .dsd
 - .adf
 - .adl
 - .img
 - .fdi
 - .uef
 - .csw
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *BeebEm*
- *B-em*



Commodore 64

Basic Information

- **Short name:** c64
- **Type:** Computer
- **Supported extensions:** 37
 - .bin
 - .cmd
 - .crt
 - .d2m
 - .d4m
 - .d64
 - .d6z
 - .d71
 - .d7z
 - .d80
 - .d81
 - .d82
 - .d8z

- .g41
- .g4z
- .g64
- .g6z
- .gz
- .lnx
- .m3u
- .nbz
- .nib
- .p00
- .prg
- .t64
- .tap
- .vfl
- .vsf
- .x64
- .x6z
- .20
- .40
- .60
- .a0
- .b0
- .rom
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Denise*
- *VICE*



Tandy Color Computer

Basic Information

- **Short name:** coco
- **Type:** Computer
- **Supported extensions:** 16
 - .cas
 - .c10
 - .cue
 - .k7
 - .dsk
 - .vdk
 - .dmk
 - .rom
 - .wav
 - .bas
 - .asc
 - .jvc
 - .os9
 - .ccc

- .sna
- .dgn

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *XRoar*



DOS (PC)

Basic Information

- **Short name:** dos
- **Type:** Computer
- **Supported extensions:** 16
 - .exe
 - .com

- .bat
- .zip
- .dosz
- .iso
- .cue
- .ins
- .img
- .ima
- .vhd
- .jrc
- .tc
- .m3u
- .m3u8
- .conf

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *DOSBox*





Dragon 32

Basic Information

- **Short name:** dragon32
- **Type:** Computer
- **Supported extensions:** 16
 - .cas
 - .c10
 - .cue
 - .k7
 - .dsk
 - .vdk
 - .dmk
 - .rom
 - .wav
 - .bas
 - .asc
 - .jvc
 - .os9
 - .ccc
 - .sna
 - .dgn

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *XRoar*



Macintosh

Basic Information

- **Short name:** macintosh
- **Type:** Computer
- **Supported extensions:** 5
 - .dsk
 - .img
 - .hvf
 - .cmd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Thomson MO/TO Series

Basic Information

- **Short name:** moto
- **Type:** Computer
- **Supported extensions:** 6
 - .fd
 - .sap
 - .k7
 - .rom
 - .m7
 - .m5

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



MSX

Basic Information

- **Short name:** msx
- **Type:** Computer
- **Supported extensions:** 11
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col
 - .dsk
 - .cas
 - .sg
 - .sc
 - .m3u

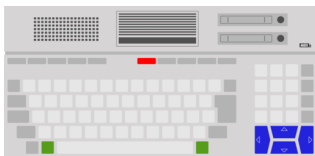
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*
- *openMSX*
- *Emulicious*



MSX2

Basic Information

- **Short name:** msx2
- **Type:** Computer
- **Supported extensions:** 11
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col

- .dsk
- .cas
- .sg
- .sc
- .m3u
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*
- *openMSX*
- *Emulicious*



MSX Turbo R

Basic Information

- **Short name:** msxturbor
- **Type:** Computer
- **Supported extensions: 11**
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col
 - .dsk
 - .cas
 - .sg
 - .sc
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*
- *openMSX*
- *Emulicious*





Tangerine Oric

Basic Information

- **Short name:** oric
- **Type:** Computer
- **Supported extensions:** 4
 - .dsk
 - .tap
 - .ort
 - .wav

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *Oricutron*





Palm OS

Basic Information

- **Short name:** palm
- **Type:** Computer
- **Supported extensions:** 4
 - .prc
 - .pqa
 - .img
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*





IBM PC

Basic Information

- **Short name:** pc
- **Type:** Computer
- **Supported extensions:** 16
 - .exe
 - .com
 - .bat
 - .zip
 - .dosz
 - .iso
 - .cue
 - .ins
 - .img
 - .ima
 - .vhd
 - .jrc
 - .tc
 - .m3u
 - .m3u8
 - .conf

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *DOSBox*



NEC PC-8000 / PC-8800 Series

Basic Information

- **Short name:** pc88
- **Type:** Computer
- **Supported extensions:** 3
 - .d88
 - .u88
 - .m3u

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



NEC パーソナルコンピュータ PC-9800シリーズ PC-9801

NEC PC-9800 Series

Basic Information

- **Short name:** pc98
- **Type:** Computer
- **Supported extensions:** 18
 - .d98
 - .zip
 - .98d
 - .fdi
 - .fdd
 - .2hd
 - .tfd
 - .d88
 - .88d
 - .hdm
 - .xdf
 - .dup
 - .cmd
 - .hdi

- .thd
- .nhd
- .hdd
- .hdn

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Sam Coupé

Basic Information

- **Short name:** samcoupe
- **Type:** Computer
- **Supported extensions: 4**
 - .dsk
 - .mgt
 - .sbt
 - .sad

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *SimCoupe*



Spectravideo

Basic Information

- **Short name:** spectravideo
- **Type:** Computer
- **Supported extensions:** 11
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col
 - .dsk
 - .cas
 - .sg
 - .sc
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*
- *openMSX*
- *Emulicious*



Tano Dragon

Basic Information

- **Short name:** tanodragon
- **Type:** Computer
- **Supported extensions:** 16
 - .cas
 - .c10
 - .cue
 - .k7

- .dsk
- .vdk
- .dmk
- .rom
- .wav
- .bas
- .asc
- .jvc
- .os9
- .ccc
- .sna
- .dgn

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *XRoar*



TI-99/4A

computer

Texas Instruments TI-99

Basic Information

- **Short name:** ti99
- **Type:** Computer
- **Supported extensions:** 1
 - .ctg

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



Thomson TO8

Basic Information

- **Short name:** to8
- **Type:** Computer
- **Supported extensions:** 6
 - .fd

- .sap
- .k7
- .rom
- .m7
- .m5

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Tandy TRS-80

Basic Information

- **Short name:** trs-80
- **Type:** Computer
- **Supported extensions:** 6
 - .dsk
 - .cas

- .cmd
- .hex
- .bds
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *XRoar*



Commodore VIC-20

Basic Information

- **Short name:** vic20
- **Type:** Computer
- **Supported extensions:** 37
 - .bin
 - .cmd
 - .crt
 - .d2m
 - .d4m

- .d64
- .d6z
- .d71
- .d7z
- .d80
- .d81
- .d82
- .d8z
- .g41
- .g4z
- .g64
- .g6z
- .gz
- .lnx
- .m3u
- .nbz
- .nib
- .p00
- .prg
- .t64
- .tap
- .vfl
- .vsf
- .x64
- .x6z
- .20
- .40
- .60
- .a0
- .b0
- .rom
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Denise*
- *VICE*



Sharp X1

Basic Information

- **Short name:** x1
- **Type:** Computer
- **Supported extensions:** 11
 - .dx1
 - .2d
 - .2hd
 - .tfd
 - .d88

- .88d
- .hdm
- .xdf
- .dup
- .cmd
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



SHARP 68000

Sharp X68000

Basic Information

- **Short name:** x68000
- **Type:** Computer
- **Supported extensions:** 12
 - .dim
 - .img
 - .d88
 - .88d
 - .hdm
 - .dup
 - .2hd
 - .xdf
 - .hdf
 - .cmd
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



Infocom Z-machine

Basic Information

- **Short name:** zmachine
- **Type:** Computer
- **Supported extensions:** 10
 - .dat
 - .z1
 - .z2
 - .z3
 - .z4
 - .z5
 - .z6
 - .z7
 - .z8
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *Frotz*





Sinclair ZX81

Basic Information

- **Short name:** zx81
- **Type:** Computer
- **Supported extensions:** 3
 - .p
 - .tZX
 - .t81

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Clock Signal*
- *EightyOne*
- *ZEsarUX*





Sinclair ZX Spectrum

Basic Information

- **Short name:** zxspectrum
- **Type:** Computer
- **Supported extensions:** 12
 - .sna
 - .szx
 - .z80
 - .tap
 - .tZX
 - .gz
 - .udi
 - .mgt
 - .img
 - .trd
 - .scl
 - .dsk

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Fuse*
- *EightyOne*
- *ZEsarUX*

ChaiLove

Basic Information

- **Short name:** chailove
- **Type:** Game Engine
- **Supported extensions: 2**
 - .chai
 - .chailove

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*

DOOM

Basic Information

- **Short name:** doom
- **Type:** Game Engine
- **Supported extensions: 3**
 - .wad
 - .iwad
 - .pwad

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ZDoom*



Lutro

Basic Information

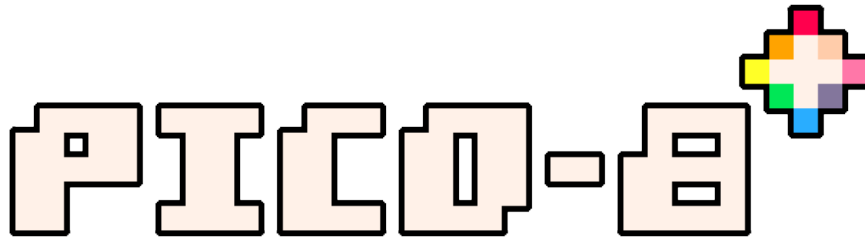
- **Short name:** lutro
- **Type:** Game Engine
- **Supported extensions:** 2
 - .lutro
 - .lua

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



PICO-8

Basic Information

- **Short name:** pico8
- **Type:** Game Engine
- **Supported extensions: 2**
 - .p8
 - .png

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *PICO-8*





ScummVM

Basic Information

- **Short name:** `scummvm`
- **Type:** Game Engine
- **Supported extensions:** 2
 - `.scummvm`
 - `.svm`

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*





Solarus

Basic Information

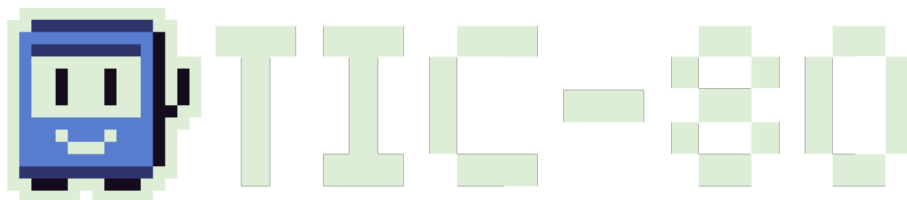
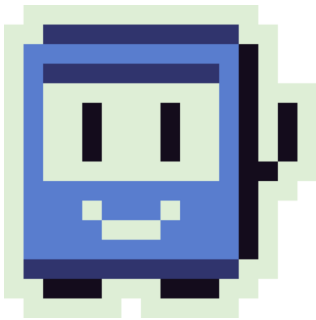
- **Short name:** solarus
- **Type:** Game Engine
- **Supported extensions:** 1
 - .solarus

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *Solarus*



TIC-80

Basic Information

- **Short name:** tic80
- **Type:** Game Engine
- **Supported extensions: 1**
 - .tic

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *TIC-80*



3DO

Basic Information

- **Short name:** 3do
- **Type:** Console
- **Supported extensions:** 5
 - .iso
 - .bin
 - .chd
 - .cue
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *4DO*



Nintendo 64 DD

Basic Information

- **Short name:** 64dd
- **Type:** Console
- **Supported extensions:** 7
 - .n64
 - .v64
 - .z64
 - .bin
 - .u1
 - .n64
 - .zip

Notes

BIOS

This system requires BIOS files to work properly.

Emulators

- *RetroArch*
- *Mupen64Plus*
- *ares*
- *Rosalie's Mupen GUI*
- *Project64*





Bally Astrocade

Basic Information

- **Short name:** astrocade
- **Type:** Arcade
- **Supported extensions:** 2
 - .chd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



ATARI 2600

Atari 2600

Basic Information

- **Short name:** atari2600
- **Type:** Console
- **Supported extensions:** 5
 - .a26
 - .bin
 - .gz
 - .rom
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Stella*
- *MAME*





Atari 5200

Basic Information

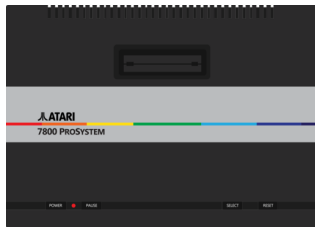
- **Short name:** atari5200
- **Type:** Console
- **Supported extensions:** 14
 - .xfd
 - .atr
 - .atr.gz
 - .xfd.gz
 - .atx
 - .cdm
 - .cas
 - .car
 - .bas
 - .bin
 - .dcm
 - .a52
 - .xex
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Atari800*
- *Altirra*
- *MAME*



ATARI 1800

Atari 7800

Basic Information

- **Short name:** atari7800
- **Type:** Console
- **Supported extensions:** 3
 - .a78
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *A7800*
- *MAME*



Atari Jaguar

Basic Information

- **Short name:** atarijaguar
- **Type:** Console
- **Supported extensions:** 7
 - .j64
 - .jag
 - .rom
 - .abs

- .cof
- .bin
- .prg

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *BigPEmu*



JAGUAR CD

Atari Jaguar CD

Basic Information

- **Short name:** atarijaguarc
- **Type:** Console
- **Supported extensions:** 7
 - .j64
 - .jag
 - .rom

- .abs
- .cof
- .bin
- .prg

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *BigPEmu*



Atari Lynx

Basic Information

- **Short name:** atarilynx
- **Type:** Console
- **Supported extensions:** 3
 - .lnx
 - .o

- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mednafen*
- *MAME*



Atari XE

Basic Information

- **Short name:** atarixe
- **Type:** Console
- **Supported extensions:** 14
 - .xfd
 - .atr
 - .atr.gz

- .xfd.gz
- .atx
- .cdm
- .cas
- .car
- .bas
- .bin
- .dcm
- .a52
- .xex
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Atari800*
- *Altirra*
- *MAME*





Philips CD-i

Basic Information

- **Short name:** cdimono1
- **Type:** Console
- **Supported extensions:** 3
 - .chd
 - .cue
 - .iso

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*





Commodore CDTV

Basic Information

- **Short name:** cdtv
- **Type:** Console
- **Supported extensions: 19**
 - .adf
 - .adz
 - .dms
 - .fdi
 - .ipf
 - .hdf
 - .hdz
 - .lha
 - .slave
 - .info
 - .cue
 - .ccd
 - .chd
 - .nrg
 - .mds
 - .iso
 - .uae
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Fairchild Channel F

Basic Information

- **Short name:** channelf
- **Type:** Console
- **Supported extensions:** 4
 - .bin
 - .rom
 - .chf
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



ColecoVision

Basic Information

- **Short name:** colecovision
- **Type:** Console
- **Supported extensions:** 4
 - .col
 - .cv
 - .bin
 - .rom

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *ColEm*
- *MAME*



Sega Dreamcast

Basic Information

- **Short name:** dreamcast
- **Type:** Console
- **Supported extensions:** 10
 - .cdi
 - .gdi
 - .chd

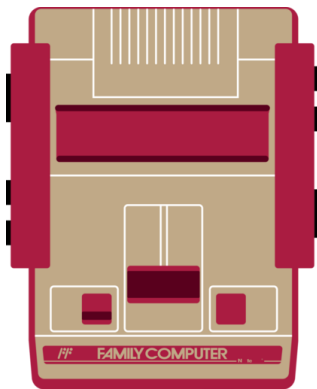
- .cue
- .bin
- .elf
- .lst
- .dat
- .m3u
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Flycast*
- *redream*



FAMILY COMPUTER

ファミリーコンピュータ

Nintendo Family Computer

Basic Information

- **Short name:** famicom
- **Type:** Console
- **Supported extensions:** 5
 - .nes

- .fds
- .unf
- .unif
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Nestopia*
- *Mesen*



Nintendo Famicom Disk System

Basic Information

- **Short name:** fds
- **Type:** Console
- **Supported extensions:** 5
 - .nes
 - .fds
 - .unf
 - .unif
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Nestopia*
- *Mesen*



Nintendo Game & Watch

Basic Information

- **Short name:** gameandwatch
- **Type:** Console
- **Supported extensions:** 2
 - .mgw
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



Sega Game Gear

Basic Information

- **Short name:** gamegear
- **Type:** Console
- **Supported extensions: 4**
 - .sms
 - .bin
 - .rom
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *MAME*
- *Emulicious*



GAME BOY

Nintendo Game Boy

Basic Information

- **Short name:** gb
- **Type:** Console
- **Supported extensions:** 8
 - .gb
 - .gbc
 - .bin
 - .rom
 - .dmg
 - .cgb
 - .sgb
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *SameBoy*
- *Emulicious*
- *Mesen*



GAME BOY COLOR

Nintendo Game Boy Color

Basic Information

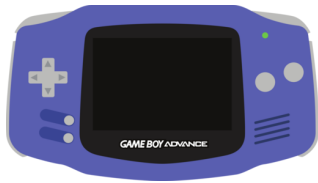
- **Short name:** gbc
- **Type:** Console
- **Supported extensions:** 8
 - .gb
 - .gbc
 - .bin
 - .rom
 - .dmg
 - .cgb
 - .sgb
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *SameBoy*
- *Emulicious*
- *Mesen*



Nintendo Game Boy Advance

Basic Information

- **Short name:** gba
- **Type:** Console
- **Supported extensions:** 4
 - .gba
 - .agb
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *mGBA*
- *Mednafen*
- *VisualBoyAdvance-M*



Nintendo GameCube

Basic Information

- **Short name:** gc
- **Type:** Console
- **Supported extensions:** 10
 - .elf
 - .iso

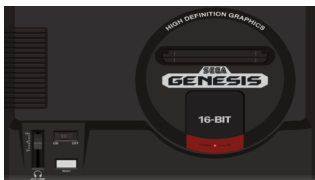
- .gcm
- .dol
- .tgc
- .wbfs
- .ciso
- .gcZ
- .wad
- .rvZ

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Dolphin*



Sega Genesis

Basic Information

- **Short name:** genesis
- **Type:** Console
- **Supported extensions: 13**
 - .mdx
 - .md
 - .smd
 - .gen
 - .bin
 - .cue
 - .iso
 - .sms
 - .gg
 - .sg
 - .68k
 - .chd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *BlastEm*
- *Mednafen*





Sega Mega Drive

Basic Information

- **Short name:** megadrive
- **Type:** Console
- **Supported extensions:** 13
 - .mdx
 - .md
 - .smd
 - .gen
 - .bin
 - .cue
 - .iso
 - .sms
 - .gg
 - .sg
 - .68k
 - .chd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *BlastEm*
- *Mednafen*



Sega 32X

Basic Information

- **Short name:** sega32x
- **Type:** Console
- **Supported extensions:** 13
 - .mdx
 - .md
 - .smd
 - .gen

- .bin
- .cue
- .iso
- .sms
- .gg
- .sg
- .68k
- .chd
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *BlastEm*
- *Mednafen*



Sega CD

Basic Information

- **Short name:** segacd
- **Type:** Console
- **Supported extensions:** 13
 - .mdx
 - .md
 - .smd
 - .gen
 - .bin
 - .cue
 - .iso
 - .sms
 - .gg
 - .sg
 - .68k
 - .chd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *BlastEm*
- *Mednafen*





Amstrad GX4000

Basic Information

- **Short name:** gx4000
- **Type:** Computer
- **Supported extensions:** 10
 - .cdt
 - .cpr
 - .cpc
 - .dsk
 - .kcr
 - .m3u
 - .sna
 - .tap
 - .voc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Caprice32*
- *Arnold*
- *MAME*



Mattel Electronics Intellivision

Basic Information

- **Short name:** intellivision
- **Type:** Console
- **Supported extensions:** 5
 - .int
 - .rom
 - .bin
 - .itv
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



SEGA
Master System

Sega Master System

Basic Information

- **Short name:** mastersystem
- **Type:** Console
- **Supported extensions:** 4
 - .sms
 - .bin
 - .rom
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *MAME*
- *Emulicious*



SEGA SG-1000

Othello Multivision

Basic Information

- **Short name:** multivision
- **Type:** Console
- **Supported extensions:** 12
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col
 - .dsk

- .cas
- .sg
- .sc
- .m3u
- .bin
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *MAME*
- *MasterGear*



Nintendo 64

Basic Information

- **Short name:** n64
- **Type:** Console
- **Supported extensions:** 7
 - .n64
 - .v64
 - .z64
 - .bin
 - .u1
 - .n64
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mupen64Plus*
- *ares*
- *Rosalie's Mupen GUI*
- *Project64*



NINTENDO DS

Nintendo DS

Basic Information

- **Short name:** nds
- **Type:** Console
- **Supported extensions:** 2
 - .nds
 - .bin

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *DeSmuME*
- *melonDS*



NEO·GEO CD

SNK Neo Geo CD

Basic Information

- **Short name:** neogeocd
- **Type:** Console
- **Supported extensions: 1**
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



Nintendo Entertainment System

Basic Information

- **Short name:** nes
- **Type:** Console
- **Supported extensions:** 5
 - .nes
 - .fds
 - .unf
 - .unif
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Nestopia*
- *Mesen*



NEOGEO POCKET

SNK Neo Geo Pocket

Basic Information

- **Short name:** ngp
- **Type:** Console
- **Supported extensions:** 3
 - .ngp
 - .ngc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *MAME*



NEOGEO POCKET COLOR

SNK Neo Geo Pocket Color

Basic Information

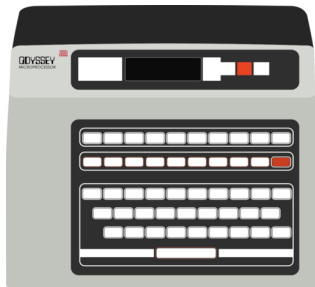
- **Short name:** ngpc
- **Type:** Console
- **Supported extensions:** 3
 - .ngp
 - .ngc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *MAME*



ODYSSEY²

Magnavox Odyssey²

Basic Information

- **Short name:** odyssey2
- **Type:** Console
- **Supported extensions:** 2
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



VIDEO PAC

Philips Videopac G7000

Basic Information

- **Short name:** videopac
- **Type:** Console
- **Supported extensions:** 2
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



NEC TurboGrafx-16

Basic Information

- **Short name:** tg16
- **Type:** Console
- **Supported extensions:** 9
 - .pce
 - .sgx
 - .cue
 - .ccd
 - .chd

- .iso
- .img
- .bin
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Mesen*



TURBO GRAFX CD

NEC TurboGrafx-CD

Basic Information

- **Short name:** tg-cd
- **Type:** Console
- **Supported extensions:** 9
 - .pce
 - .sgx
 - .cue
 - .ccd
 - .chd
 - .iso

- .img
- .bin
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Mesen*



NEC PC Engine

Basic Information

- **Short name:** pcengine
- **Type:** Console
- **Supported extensions: 9**
 - .pce
 - .sgx
 - .cue
 - .ccd
 - .chd
 - .iso
 - .img
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Mesen*





NEC PC Engine CD

Basic Information

- **Short name:** pcenginecd
- **Type:** Console
- **Supported extensions:** 9
 - .pce
 - .sgx
 - .cue
 - .ccd
 - .chd
 - .iso
 - .img
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Mesen*



NEC PC-FX

Basic Information

- **Short name:** pcfx
- **Type:** Console
- **Supported extensions:** 4
 - .cue
 - .ccd
 - .toc
 - .chd

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mednafen*



Pokémon mini

Nintendo Pokémon Mini

Basic Information

- **Short name:** pokemini
- **Type:** Console
- **Supported extensions:** 1
 - .min

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*



Sony PlayStation 2

Basic Information

- **Short name:** ps2
- **Type:** Console
- **Supported extensions:** 17
 - .elf
 - .iso
 - .ciso
 - .chd
 - .cso
 - .bin
 - .mdf
 - .nrg
 - .dump
 - .gz
 - .img
 - .m3u
 - .z

- .z2
- .bz2
- .ima
- .zip

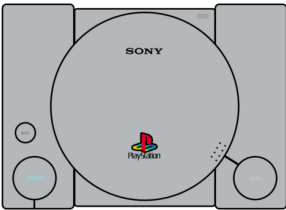
Notes

BIOS

This system requires BIOS files to work properly.

Emulators

- *RetroArch*
- *PCSX2*
- *Play!*



Sony PlayStation

Basic Information

- **Short name:** psx
- **Type:** Console
- **Supported extensions: 14**
 - .bin
 - .cue
 - .toc
 - .m3u
 - .ccd
 - .exe
 - .pbp
 - .chd
 - .img
 - .mdf
 - .cbn
 - .iso
 - .z
 - .znx

Notes

BIOS

This system requires BIOS files to work properly.

Emulators

- *RetroArch*
- *DuckStation*
- *Mednafen*



Sega Saturn

Basic Information

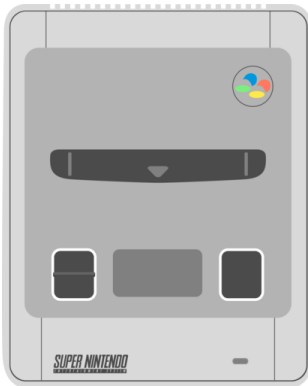
- **Short name:** saturn
- **Type:** Console
- **Supported extensions:** 8
 - .cue
 - .toc
 - .m3u
 - .ccd
 - .chd
 - .iso
 - .mds
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mednafen*



Super Nintendo Entertainment System

Basic Information

- **Short name:** snes
- **Type:** Console
- **Supported extensions:** 14
 - .bin
 - .bml
 - .bs
 - .bsx

- .dx2
- .fig
- .gd3
- .gd7
- .mgd
- .sfc
- .smc
- .st
- .swc
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Snes9X*
- *ares*
- *Mednafen*
- *Mesen*
- *bsnes*



Nintendo Super Famicom

Basic Information

- **Short name:** sfc
- **Type:** Console
- **Supported extensions:** 14
 - .bin
 - .bml
 - .bs
 - .bsx
 - .dx2
 - .fig
 - .gd3
 - .gd7
 - .mgd
 - .sfc
 - .smc
 - .st
 - .swc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Snes9X*
- *ares*
- *Mednafen*
- *Mesen*
- *bsnes*





Nintendo Satellaview

Basic Information

- **Short name:** satellaview
- **Type:** Console
- **Supported extensions:** 14
 - .bin
 - .bml
 - .bs
 - .bsx
 - .dx2
 - .fig
 - .gd3
 - .gd7
 - .mgd
 - .sfc
 - .smc
 - .st
 - .swc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Snes9X*
- *ares*
- *Mednafen*
- *Mesen*
- *bsnes*



スーファミターボ *SUFAMI TURBO* スーパーファミコン

Bandai SuFami Turbo

Basic Information

- **Short name:** sufami
- **Type:** Console
- **Supported extensions:** 14
 - .bin
 - .bml
 - .bs
 - .bsx
 - .dx2
 - .fig
 - .gd3
 - .gd7
 - .mgd
 - .sfc
 - .smc
 - .st
 - .swc
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Snes9X*
- *ares*
- *Mednafen*
- *Mesen*
- *bsnes*



SEGA SG-1000

Sega SG-1000

Basic Information

- **Short name:** sg-1000
- **Type:** Console
- **Supported extensions:** 12
 - .rom
 - .ri
 - .mx1
 - .mx2
 - .col

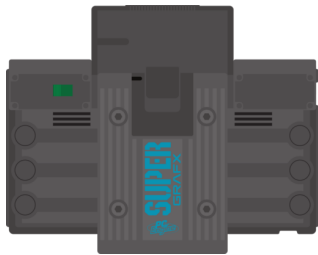
- .dsk
- .cas
- .sg
- .sc
- .m3u
- .bin
- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *MAME*
- *MasterGear*



SUPER GRAFX

NEC SuperGrafx

Basic Information

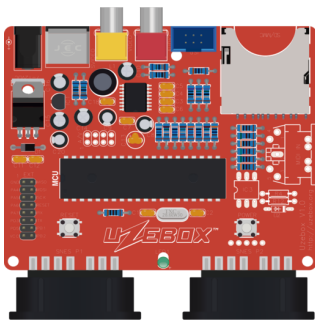
- **Short name:** supergrafx
- **Type:** Console
- **Supported extensions: 9**
 - .pce
 - .sgx
 - .cue
 - .ccd
 - .chd
 - .iso
 - .img
 - .bin
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*
- *Mesen*





Uzebox

Basic Information

- **Short name:** uzebox
- **Type:** Console
- **Supported extensions:** 1
 - .uze

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *CUzeBox*



GCE Vectrex

Basic Information

- **Short name:** vectrex
- **Type:** Console
- **Supported extensions:** 3
 - .bin
 - .vec
 - .gam

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



VIRTUAL BOY

Nintendo Virtual Boy

Basic Information

- **Short name:** virtualboy
- **Type:** Console
- **Supported extensions:** 3
 - .vb
 - .vboy
 - .bin

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Mednafen*



Bandai WonderSwan

Basic Information

- **Short name:** wonderswan
- **Type:** Console
- **Supported extensions:** 3
 - .ws
 - .wsc
 - .pc2

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*



Bandai WonderSwan Color

Basic Information

- **Short name:** wonderswancolor
- **Type:** Console
- **Supported extensions:** 3
 - .ws
 - .wsc
 - .pc2

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *ares*
- *Mednafen*



Microsoft Xbox

Basic Information

- **Short name:** xbox
- **Type:** Console
- **Supported extensions: 1**
 - .iso

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *xemu*



Arcade

Basic Information

- **Short name:** arcade
- **Type:** Arcade
- **Supported extensions:** 2
 - .chd
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



Atomiswave

Basic Information

- **Short name:** atomiswave
- **Type:** Arcade
- **Supported extensions: 10**
 - .cdi
 - .gdi
 - .chd
 - .cue
 - .bin
 - .elf
 - .lst
 - .dat
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Flycast*
- *redream*



DAPHNE

Daphne Arcade LaserDisc

Basic Information

- **Short name:** daphne
- **Type:** Arcade
- **Supported extensions: 1**
 - .daphne

Notes

There are no special notes for this system. Games should work out of the box.

Emulators





Sega NAOMI

Basic Information

- **Short name:** naomi
- **Type:** Arcade
- **Supported extensions:** 10
 - .cdi
 - .gdi
 - .chd
 - .cue
 - .bin
 - .elf
 - .lst
 - .dat
 - .m3u
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Flycast*
- *redream*



NAOMI

GD-ROM SYSTEM

Sega NAOMI GD-ROM

Basic Information

- **Short name:** naomigd
- **Type:** Arcade
- **Supported extensions:** 10
 - .cdi
 - .gdi
 - .chd
 - .cue
 - .bin
 - .elf
 - .lst
 - .dat
 - .m3u

- .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Flycast*
- *redream*



NEO GEO

SNK Neo Geo

Basic Information

- **Short name:** neogeo
- **Type:** Arcade
- **Supported extensions:** 1
 - .zip

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *MAME*



NINTENDO 3DS

Nintendo 3DS

Basic Information

- **Short name:** n3ds
- **Type:** Modern Console
- **Supported extensions:** 7
 - .3ds
 - .3dsx
 - .elf
 - .axf
 - .cci
 - .cxi
 - .app

Notes

Encrypted games

Game files might be encrypted, which have to be decrypted before they can be run by emulators.

Emulators

- *RetroArch*
- *Citra*



PLAYSTATION 3

Sony PlayStation 3

Basic Information

- **Short name:** ps3
- **Type:** Modern Console
- **Supported extensions:** 1
 - PARAM.sfo

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RPCS3*



PlayStation Portable

Sony PlayStation Portable

Basic Information

- **Short name:** psp
- **Type:** Modern Console
- **Supported extensions:** 5
 - .elf
 - .iso
 - .cso
 - .prx
 - .pbp

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *PPSSPP*



PSVITA

Sony PlayStation Vita

Basic Information

- **Short name:** psvita
- **Type:** Modern Console
- **Supported extensions: 1**
 - .vpk

Notes

There are no special notes for this system. Games should work out of the box.

Emulators



Nintendo Switch

Basic Information

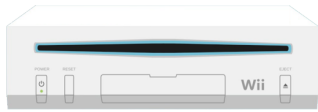
- **Short name:** switch
- **Type:** Modern Console
- **Supported extensions:** 5
 - .nca
 - .nro
 - .nso
 - .nsp
 - .xci

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *yuzu*
- *Ryujinx*



wii

Nintendo Wii

Basic Information

- **Short name:** wii
- **Type:** Modern Console
- **Supported extensions:** 10
 - .elf
 - .iso
 - .gcm
 - .dol
 - .tgc
 - .wbfs
 - .ciso
 - .gcx
 - .wad

– .rvz

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *RetroArch*
- *Dolphin*



Nintendo Wii U

Basic Information

- **Short name:** wiiu
- **Type:** Modern Console
- **Supported extensions:** 3
 - .wud
 - .wux
 - .rpx

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *Cemu*



XBOX 360

Microsoft Xbox 360

Basic Information

- **Short name:** xbox360
- **Type:** Modern Console
- **Supported extensions:** 2
 - .iso
 - .xex

Notes

There are no special notes for this system. Games should work out of the box.

Emulators

- *Xenia*

1.7 Emulators

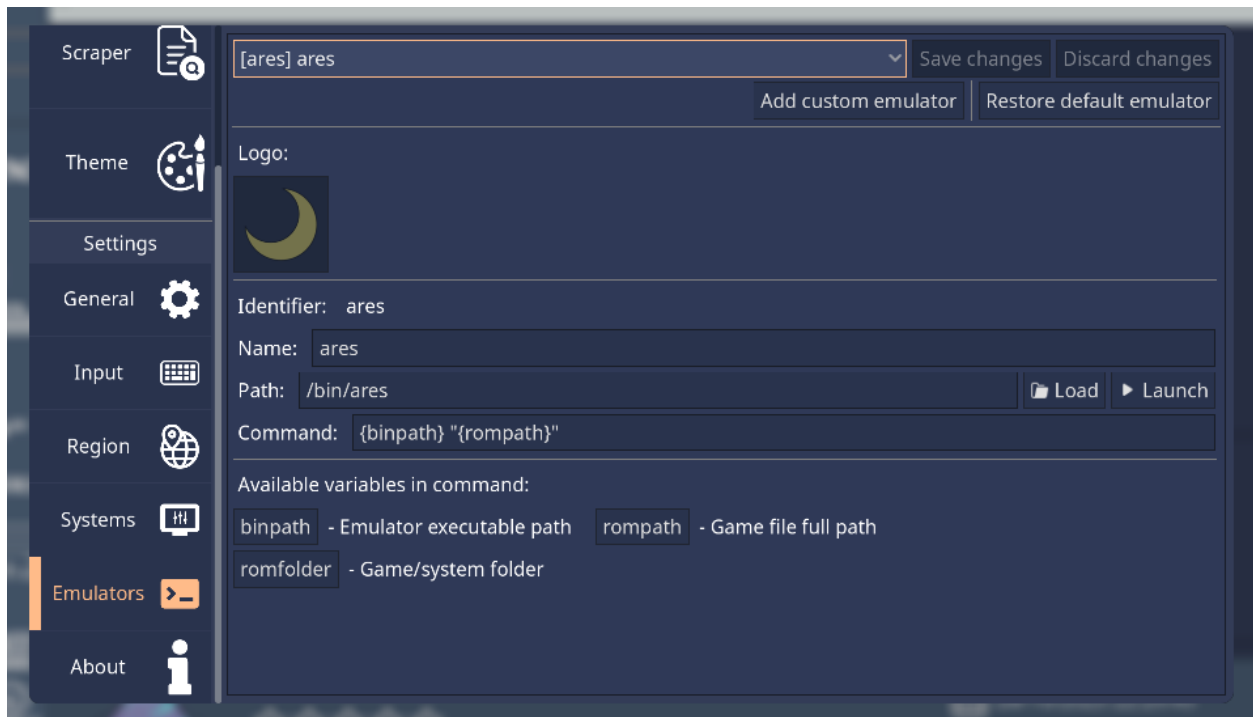
An emulator is a tool required to play games outside of a system's original hardware.

1.7.1 Adding custom emulators

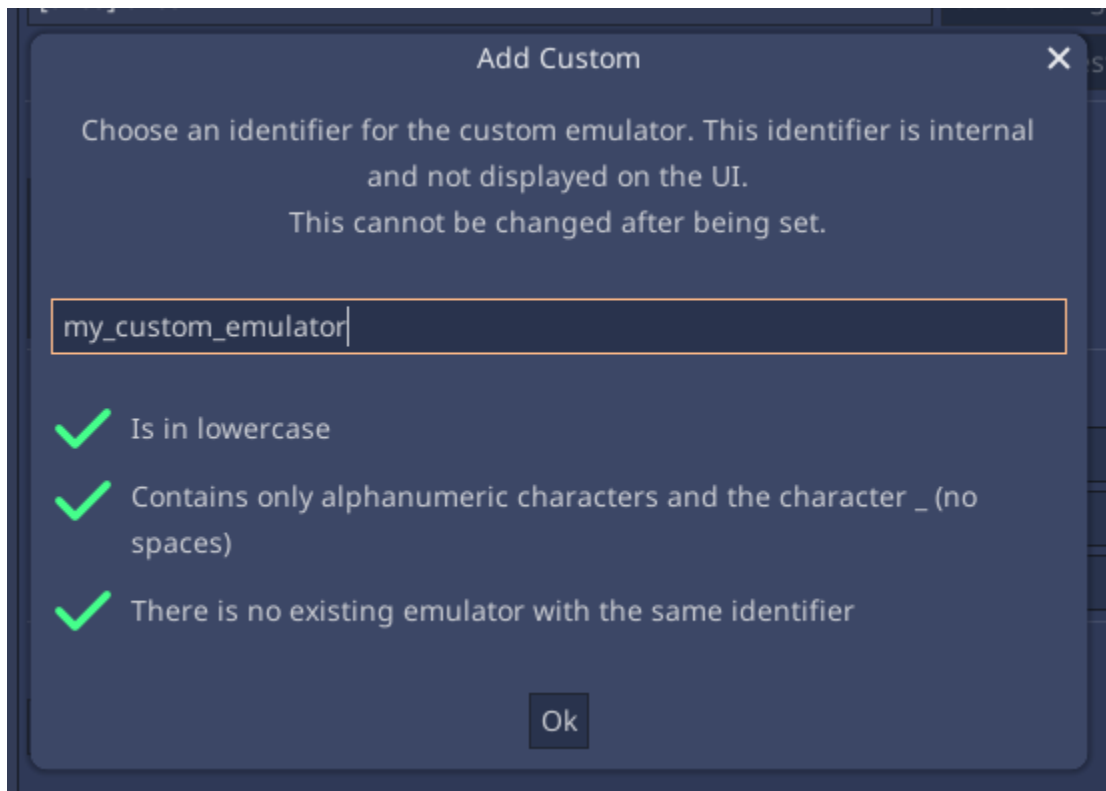
There are two ways to achieve this:

Application

Open the **Settings** ( / ) panel, and navigate to the **Emulators** section. Click on the **Add custom emulator**.



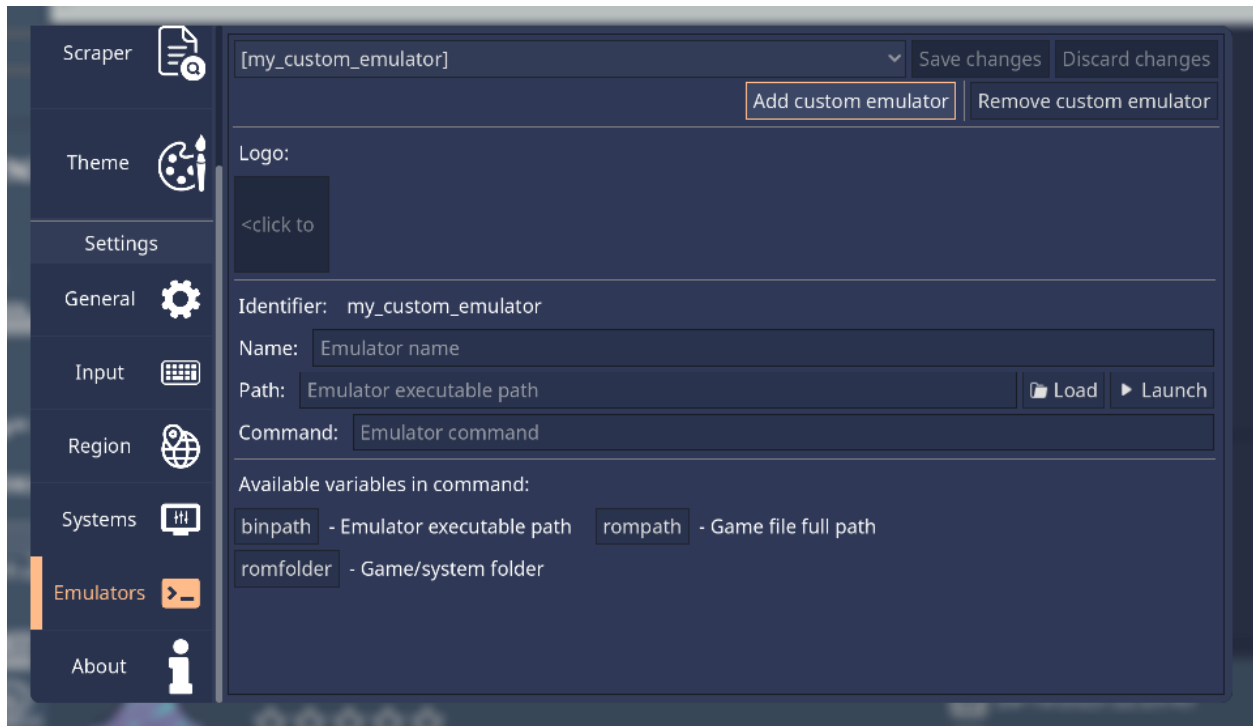
This will bring a popup to choose a short emulator name. This is what's used internally to uniquely identify the emulator.



It must follow the following rules to be valid:

- Be lowercase
- Use only alphanumeric characters (a-z and 0-9) and the underscore character
- Not be an existing emulator short name

After that you must now set all the information relevant to that emulator in each available field, such as it's name, executable path, and command to run.



For the command, you can use variables as placeholders, such as the emulator's path and the game file location. Use the existing buttons to place them, or write them manually in the form of "{variable_name}". The available variables are:

- {binpath}: The emulator's executable path
- {rompath}: The full game file path

Note: Don't forget that paths may have spaces. Surround variables with quotes " to avoid issues.

Manually

Edit the `rh_emulators.json` file in the configuration directory. Add a dictionary entry to the existing array, with the emulator information:



```
[
  {
    "name": "my_custom_emulator",
    "fullname": "My Custom Emulator",
    "binpath": "/path/to/emulator",
    "command": "{binpath} -e \"{rompath}\""
  }
]
```

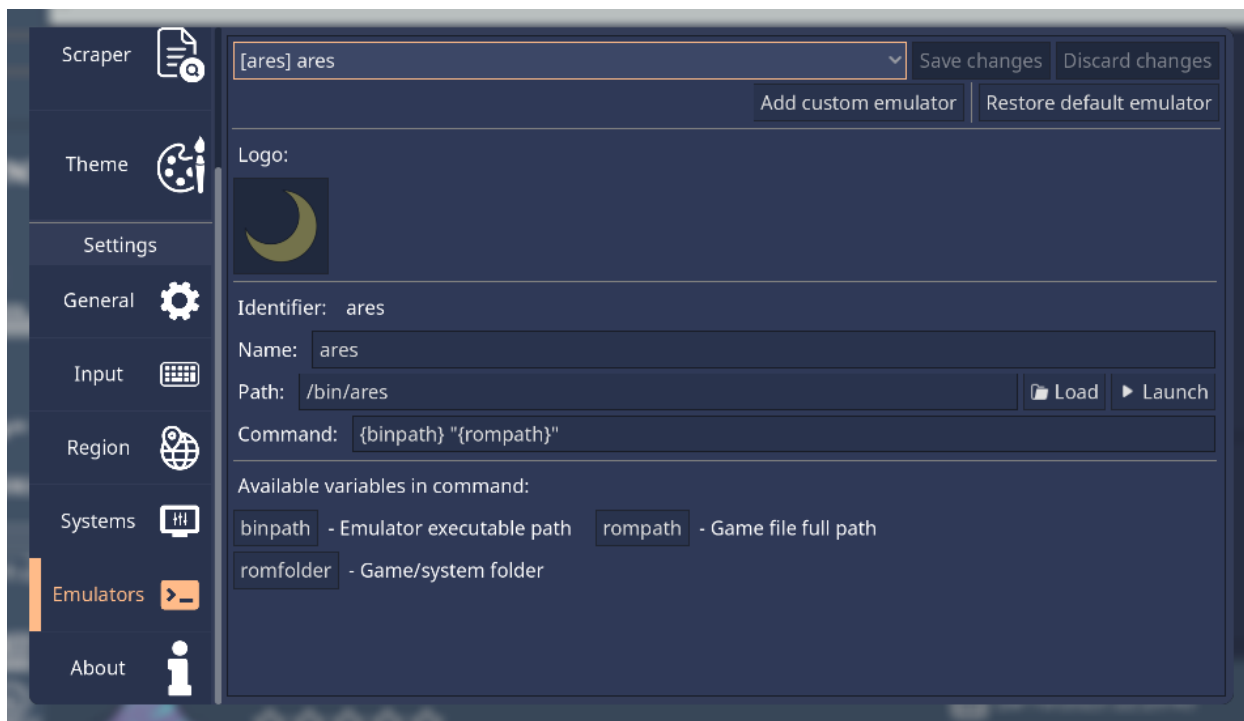
For more information on the necessary keys and their values, see the [rh_emulators.json specification](#) section.

1.7.2 Changing emulator settings

There are two ways to achieve this:

Application

Open the **Settings** ( / ) panel, and navigate to the **Emulators** section.



Then modify these fields to your liking. Remember to save changes when you're done.

You can also revert to the default settings by clicking the **Restore default emulator** button.

Manually

Edit the `rh_emulators.json` file in the configuration directory. Add a dictionary entry to the existing array, with the `name` field set to the name of the emulator you want to modify, and then changing the other key values:

```
[
  {
    "name": "mupen64plus",
    "fullname": "Mupen64Plus (Bleeding Edge)",
    "binpath": "/bin/mupen64plus-custom"
  }
]
```

For more information on the existing keys and their values, see the [rh_emulators.json specification](#) section.

1.7.3 Removing default emulators

There are two ways to achieve this:

Application

Note: This is still not implemented in the app. For now you must follow instructions on the **Manually** section.

Manually

Edit the `rh_emulators.json` file in the configuration directory. Add a dictionary entry to the existing array, with the `name` field set to the name of the emulator you want to remove, and then add the special `#delete` key to it:

```
[
  {
    "name": "mupen64plus",
    "#delete": true
  }
]
```

For more information, see the *rh_emulators.json specification* section.

1.7.4 rh_emulators.json specification

This page details the JSON specification of the emulators file, `rh_emulators.json`. This file is applied on top of the default emulators specification file, which can be checked [here](#).

Warning: Always close RetroHub before editing this file, otherwise your changes may be overridden!

Warning: Ensure the file is valid before using it. You can use a JSON validator such as [JSONLint](#).

This JSON consists of an array of dictionary entries. Each entry details a custom emulator or modifications to an existing emulator:

```
[
  {
    "name": "retroarch",
    "binpath": "/usr/local/bin/custom-retroarch",
    "corepath": "~/.config/retroarch/cores"
  },
  {
    "name": "dolphin",
    "#delete": true
  },
  {
    "name": "my_custom_Emulator",
```

(continues on next page)

(continued from previous page)

```

        "fullname": "My Custom Emulator",
        "binpath": [
            {
                "windows": [
                    "C:/Program Files/My Custom Emulator/
↪MyCustomEmulator.exe"
                ],
            },
            {
                "macos": [
                    "/Applications/MyCustomEmulator.app/Contents/
↪MacOS/MyCustomEmulator"
                ],
            },
            {
                "linux": [
                    "/bin/my_custom_emulator"
                ],
            },
        ],
        "command": "{binpath} \"{rompath}\""
    }
]

```

Each entry has specific keys for emulator information, detailed below:

- name - String (**required**)

Short name for the emulator

```
"name": "mupen64plus"
```

- fullname - String (**required**)

Emulator name displayed to the user

```
"fullname": "Mupen64Plus"
```

- binpath - *Path* (**required**)

Path to the emulator executable.

```
"binpath": "/usr/bin/mupen64plus"
```

- command: String (**required**)

Command to run the emulator. It can use some pre-defined variables to fill in information:

- {binpath} - Path to the emulator executable
- {rompath} - Path to the game file

```
"command": "{binpath} --file \"{rompath}\""
```

Note: Game files may have spaces in the file name. Ensure you surround the {rompath} variable with quotes " to

avoid such issues.

- `#delete` - Boolean (*optional*)

If this key exists, RetroHub will delete this emulator, making it unavailable to the user. Must be combined with a `name` key to know which emulator to delete.

```
"name": "mupen64plus",
"#delete": true
```

RetroArch configs

RetroArch is a more complex emulator that needs more information to properly work:

- `corepath` - *Path* (**required**)

Path to the RetroArch cores folder.

```
"corepath": "/usr/lib/libretro"
```

- `cores` - Array of Dictionaries (**required**)

Contains the definition of cores as Dictionary entries:

- `name` - String (**required**)

Short name for the core

- `fullname` - String (**required**)

Core name displayed to the user

- `file` - *Path* (**required**)

Core file name

```
"cores": [
  {
    "name": "mupen64plus-next",
    "fullname": "Mupen64Plus-Next",
    "file": [
      {
        "windows": "mupen64plus_next_libretro.dll",
        "macos": "mupen64plus_next_libretro.dylib",
        "linux": "mupen64plus_next_libretro.so"
      }
    ]
  }
]
```

Paths

Paths point to files in the user's computer. There are peculiarities to each OS, which RetroHub handles for you. Here are some notes valid for all operating systems:

- Path separators are always /, instead of \\. (e.g C:/Program Files/My Emulator/MyEmulator.exe)
- You can go to the previous folder by typing .. (e.g C:/Program Files/./Program Files (x86).)
- You can use ~ to refer to the user's home folder `` (e.g. ~/Desktop/Emulators/MyEmulator.exe)
- Absolute paths start with / on macOS and Linux, and the drive letter (e.g. C:/) on Windows. Otherwise, it's a relative path, which is relative to the RetroHub executable.

Since paths are highly OS dependent, RetroHub uses a special syntax to define them. For example, the `binpath` is supposed to be a string, but is defined as a dictionary with keys `windows`, `macos` and `linux`:

```
"name": "mupen64plus",
"binpath": [
  {
    "windows": [
      "mupen64plus/mupen64plus-ui-console.exe",
      "Emulators/mupen64plus/mupen64plus-ui-console.exe",
      "../mupen64plus/mupen64plus-ui-console.exe"
    ],
  },
  {
    "macos": [
      "/Applications/mupen64plus.app/Contents/MacOS/mupen64plus",
      "/usr/local/bin/mupen64plus"
    ],
  },
  {
    "linux": [
      "/bin/m64p",
      "/bin/mupen64plus",
      "/usr/bin/m64p",
      "/usr/bin/mupen64plus",
      "/var/lib/flatpak/exports/bin/io.github.m64p.m64p",
      "~/local/share/flatpak/exports/bin/io.github.m64p.m64p"
    ],
  }
]
```

RetroHub will search these paths, according to the current OS, and return the first valid one. If no valid one is found, RetroHub will use the first defined path.

For example, on Windows, with the following definition, and with RetroHub finding the emulator on the highlighted line:

```
"binpath": [
  {
    "windows": [
      "C:/Program Files/My Custom Emulator/MyCustomEmulator.exe",
      "C:/Program Files (x86)/My Custom Emulator/MyCustomEmulator.exe",
      "D:/Program Files/My Custom Emulator/MyCustomEmulator.exe",
    ],
  }
]
```

(continues on next page)

(continued from previous page)

```
        "D:/Program Files (x86)/My Custom Emulator/MyCustomEmulator.exe"  
    ]  
}  
]
```

It will be converted to:

```
"binpath": "C:/Program Files (x86)/My Custom Emulator/MyCustomEmulator.exe"
```

Note: Currently, it's not possible to set these alternative paths from the application. If you wish to use this feature, you must edit the file manually.

1.7.5 Emulator list



4DO

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- 3DO



A7800

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari 7800*



Altirra

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari 800*
- *Atari 5200*
- *Atari XE*



ares

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo 64 DD*
- *ColecoVision*
- *Nintendo Family Computer*
- *Nintendo Famicom Disk System*
- *Sega Game Gear*
- *Nintendo Game Boy*
- *Nintendo Game Boy Color*
- *Nintendo Game Boy Advance*
- *Sega Genesis*
- *Sega Mega Drive*
- *Sega 32X*
- *Sega CD*
- *Sega Master System*
- *Othello Multivision*
- *Nintendo 64*
- *Nintendo Entertainment System*
- *SNK Neo Geo Pocket*
- *SNK Neo Geo Pocket Color*
- *NEC TurboGrafx-16*
- *NEC TurboGrafx-CD*
- *NEC PC Engine*
- *NEC PC Engine CD*
- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*
- *Sega SG-1000*
- *NEC SuperGrafx*
- *Bandai WonderSwan*
- *Bandai WonderSwan Color*

Arnold

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Amstrad CPC*
- *Amstrad GX4000*

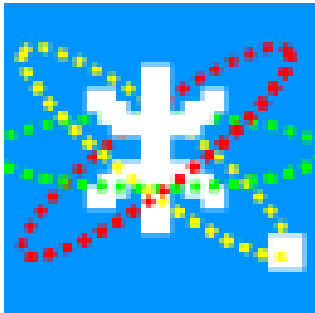
Atari800

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari 800*
- *Atari 5200*
- *Atari XE*



B-em

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *BBC Micro*



BeebEm

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *BBC Micro*



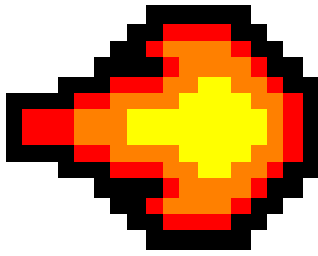
BigPEmu

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari Jaguar*
- *Atari Jaguar CD*



BlastEm

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sega Genesis*
- *Sega Mega Drive*
- *Sega 32X*
- *Sega CD*



bsnes

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*



Caprice32

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Amstrad CPC*
- *Amstrad GX4000*



Cemu

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Wii U*



Citra

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo 3DS*



Clock Signal

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sinclair ZX81*



ColEm

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *ColecoVision*

CUzeBox

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Uzebox*



Denise

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Commodore 64*
- *Commodore VIC-20*



DeSmuME

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo DS*



Dolphin

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo GameCube*
- *Nintendo Wii*



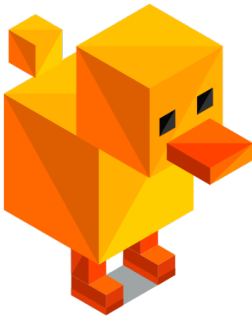
DOSBox

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *DOS (PC)*
- *IBM PC*



DuckStation

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sony PlayStation*

EightyOne

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sinclair ZX81*
- *Sinclair ZX Spectrum*



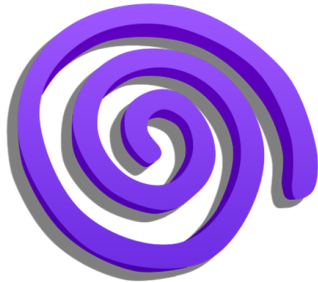
Emulicious

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *MSX*
- *MSX2*
- *MSX Turbo R*
- *Spectravideo*
- *Sega Game Gear*
- *Nintendo Game Boy*
- *Nintendo Game Boy Color*
- *Sega Master System*



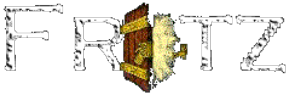
Flycast

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sega Dreamcast*
- *Atomiswave*
- *Sega NAOMI*
- *Sega NAOMI GD-ROM*



Frotz

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Infocom Z-machine*



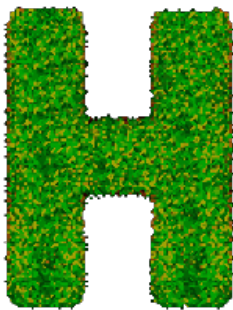
Fuse

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sinclair ZX Spectrum*



Hatari

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari ST*



MAME

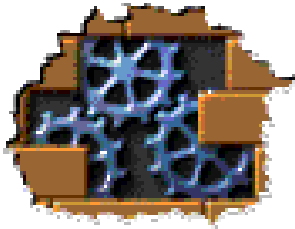
Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Amstrad CPC*
- *Apple II*
- *Atari 800*
- *MSX*
- *MSX2*
- *MSX Turbo R*
- *NEC PC-8000 / PC-8800 Series*
- *Spectravideo*
- *Texas Instruments TI-99*
- *Sharp X1*
- *Sharp X68000*
- *Bally Astrocade*
- *Atari 2600*
- *Atari 5200*
- *Atari 7800*
- *Atari Lynx*
- *Atari XE*
- *Philips CD-i*
- *Fairchild Channel F*

- *ColecoVision*
- *Nintendo Game & Watch*
- *Sega Game Gear*
- *Amstrad GX4000*
- *Mattel Electronics Intellivision*
- *Sega Master System*
- *Othello Multivision*
- *SNK Neo Geo CD*
- *SNK Neo Geo Pocket*
- *SNK Neo Geo Pocket Color*
- *Magnavox Odyssey²*
- *Philips Videopac G7000*
- *Sega SG-1000*
- *GCE Vectrex*
- *Arcade*
- *SNK Neo Geo*



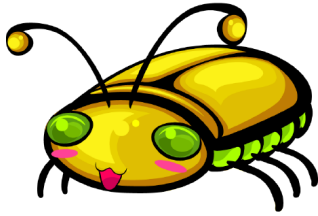
MasterGear

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Othello Multivision*
- *Sega SG-1000*



Mednafen

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Apple II*
- *Atari Lynx*
- *Nintendo Family Computer*
- *Nintendo Famicom Disk System*
- *Sega Game Gear*
- *Nintendo Game Boy*
- *Nintendo Game Boy Color*
- *Nintendo Game Boy Advance*
- *Sega Genesis*
- *Sega Mega Drive*
- *Sega 32X*
- *Sega CD*
- *Sega Master System*
- *Nintendo Entertainment System*
- *SNK Neo Geo Pocket*
- *SNK Neo Geo Pocket Color*
- *NEC TurboGrafx-16*
- *NEC TurboGrafx-CD*
- *NEC PC Engine*
- *NEC PC Engine CD*
- *NEC PC-FX*
- *Sony PlayStation*

- *Sega Saturn*
- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*
- *NEC SuperGrafx*
- *Nintendo Virtual Boy*
- *Bandai WonderSwan*
- *Bandai WonderSwan Color*



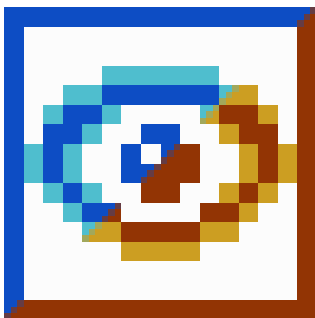
melonDS

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo DS*



Mesen

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Family Computer*
- *Nintendo Famicom Disk System*
- *Nintendo Game Boy*
- *Nintendo Game Boy Color*
- *Nintendo Entertainment System*
- *NEC TurboGrafx-16*
- *NEC TurboGrafx-CD*
- *NEC PC Engine*
- *NEC PC Engine CD*
- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*
- *NEC SuperGrafx*



mGBA

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Game Boy Advance*



Mupen64Plus

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo 64 DD*
- *Nintendo 64*



Nestopia

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Family Computer*
- *Nintendo Famicom Disk System*
- *Nintendo Entertainment System*



openMSX

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *MSX*
- *MSX2*
- *MSX Turbo R*
- *Spectravideo*

Oricutron

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Tangerine Oric*



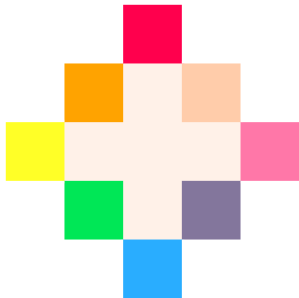
PCSX2

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sony PlayStation 2*



PICO-8

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *PICO-8*



Play!

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sony PlayStation 2*



PPSSPP

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sony PlayStation Portable*



Project64

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo 64 DD*
- *Nintendo 64*



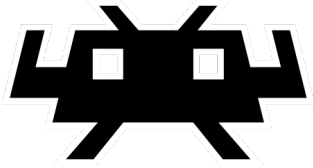
redream

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sega Dreamcast*
- *Atomiswave*
- *Sega NAOMI*
- *Sega NAOMI GD-ROM*



RetroArch

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Commodore Amiga*
- *Commodore Amiga 600*
- *Commodore Amiga 1200*
- *Commodore Amiga CD32*
- *Amstrad CPC*
- *Apple II*
- *Atari 800*
- *Atari ST*
- *Commodore 64*
- *DOS (PC)*
- *Macintosh*
- *Thomson MO/TO Series*
- *MSX*
- *MSX2*
- *MSX Turbo R*
- *Palm OS*
- *IBM PC*
- *NEC PC-8000 / PC-8800 Series*
- *NEC PC-9800 Series*
- *Sam Coupé*

- *Spectravideo*
- *Texas Instruments TI-99*
- *Thomson TO8*
- *Commodore VIC-20*
- *Sharp X1*
- *Sharp X68000*
- *Sinclair ZX81*
- *Sinclair ZX Spectrum*
- *ChaiLove*
- *DOOM*
- *Lutro*
- *ScummVM*
- *TIC-80*
- *3DO*
- *Nintendo 64 DD*
- *Bally Astrocade*
- *Atari 2600*
- *Atari 5200*
- *Atari 7800*
- *Atari Jaguar*
- *Atari Lynx*
- *Atari XE*
- *Philips CD-i*
- *Commodore CDTV*
- *Fairchild Channel F*
- *ColecoVision*
- *Sega Dreamcast*
- *Nintendo Family Computer*
- *Nintendo Famicom Disk System*
- *Nintendo Game & Watch*
- *Sega Game Gear*
- *Nintendo Game Boy*
- *Nintendo Game Boy Color*
- *Nintendo Game Boy Advance*
- *Nintendo GameCube*
- *Sega Genesis*

- *Sega Mega Drive*
- *Sega 32X*
- *Sega CD*
- *Amstrad GX4000*
- *Mattel Electronics Intellivision*
- *Sega Master System*
- *Othello Multivision*
- *Nintendo 64*
- *Nintendo DS*
- *SNK Neo Geo CD*
- *Nintendo Entertainment System*
- *SNK Neo Geo Pocket*
- *SNK Neo Geo Pocket Color*
- *Magnavox Odyssey²*
- *Philips Videopac G7000*
- *NEC TurboGrafx-16*
- *NEC TurboGrafx-CD*
- *NEC PC Engine*
- *NEC PC Engine CD*
- *NEC PC-FX*
- *Nintendo Pokémon Mini*
- *Sony PlayStation 2*
- *Sony PlayStation*
- *Sega Saturn*
- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*
- *Sega SG-1000*
- *NEC SuperGrafx*
- *Uzebox*
- *GCE Vectrex*
- *Nintendo Virtual Boy*
- *Bandai WonderSwan*
- *Bandai WonderSwan Color*
- *Arcade*

- *Atomiswave*
- *Sega NAOMI*
- *Sega NAOMI GD-ROM*
- *SNK Neo Geo*
- *Nintendo 3DS*
- *Sony PlayStation Portable*
- *Nintendo Wii*



Rosalie's Mupen GUI

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo 64 DD*
- *Nintendo 64*



RPCS3

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sony PlayStation 3*



Ryujinx

Notes

Initial Configuration

Ryujinx requires a few configuration steps before it can be used, check [here](#).

Supported systems

- *Nintendo Switch*



SameBoy

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Game Boy*
- *Nintendo Game Boy Color*

SimCoupe

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sam Coupé*



Snes9X

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Super Nintendo Entertainment System*
- *Nintendo Super Famicom*
- *Nintendo Satellaview*
- *Bandai SuFami Turbo*



Solarus

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Solarus*



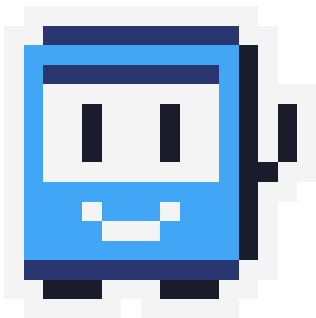
Stella

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Atari 2600*



TIC-80

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *TIC-80*



VisualBoyAdvance-M

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Nintendo Game Boy Advance*



VICE

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Commodore 64*
- *Commodore VIC-20*



xemu

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Microsoft Xbox*



Xenia

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Microsoft Xbox 360*

XRoar

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Tandy Color Computer*
- *Dragon 32*
- *Tano Dragon*
- *Tandy TRS-80*



yuzu

Notes

Initial Configuration

yuzu requires a few configuration steps before it can be used, check [here](#).

Supported systems

- *Nintendo Switch*



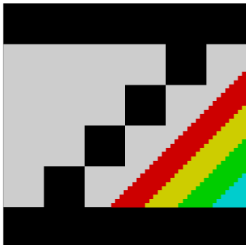
ZDoom

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *DOOM*



ZEsrUX

Notes

There are no special notes for this emulator. Games should work out of the box.

Supported systems

- *Sinclair ZX81*
- *Sinclair ZX Spectrum*

1.8 Integrations

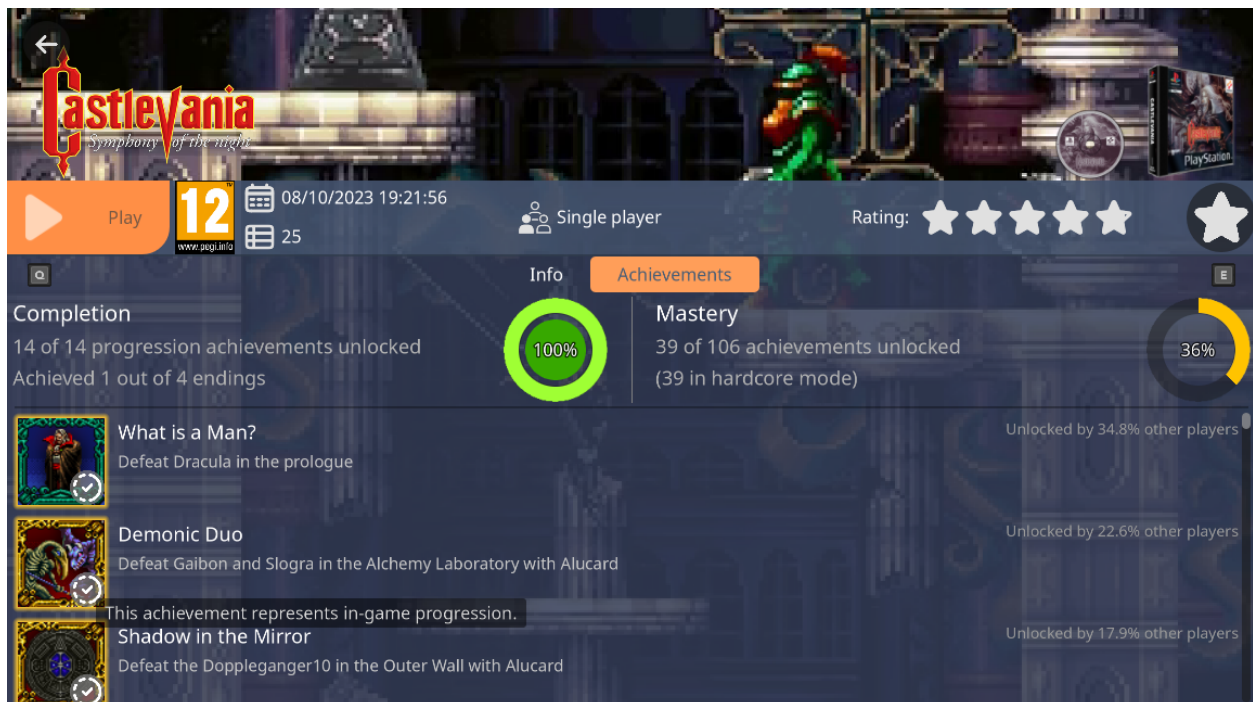
Integrations are optional services that you can enable to enrich your experience by adding additional features, be it on the main application and/or on supported themes.

1.8.1 RetroAchievements





RetroAchievements is a service that introduces achievements on classic systems, such as the SNES, Genesis, PlayStation, and many others. It also tracks your progression, statistics, and allows you to compete with other players.

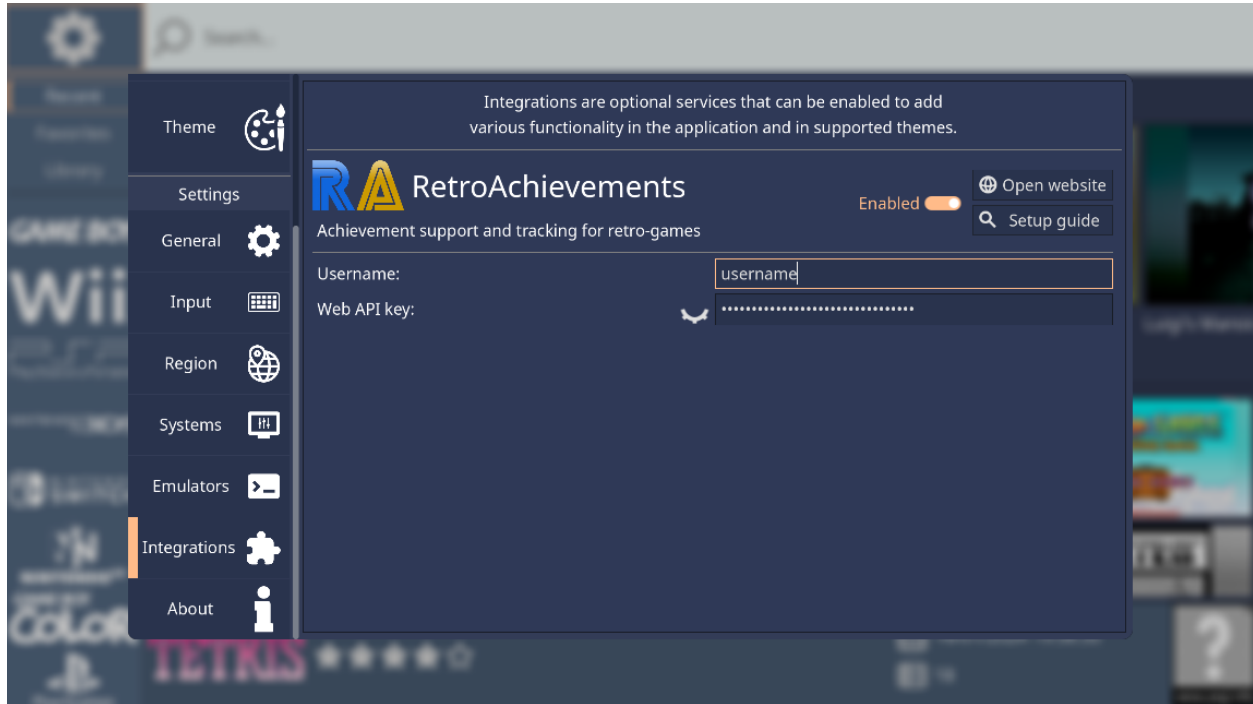
RetroHub can fetch your personal achievement information and progression, providing it for supported themes.



Setup

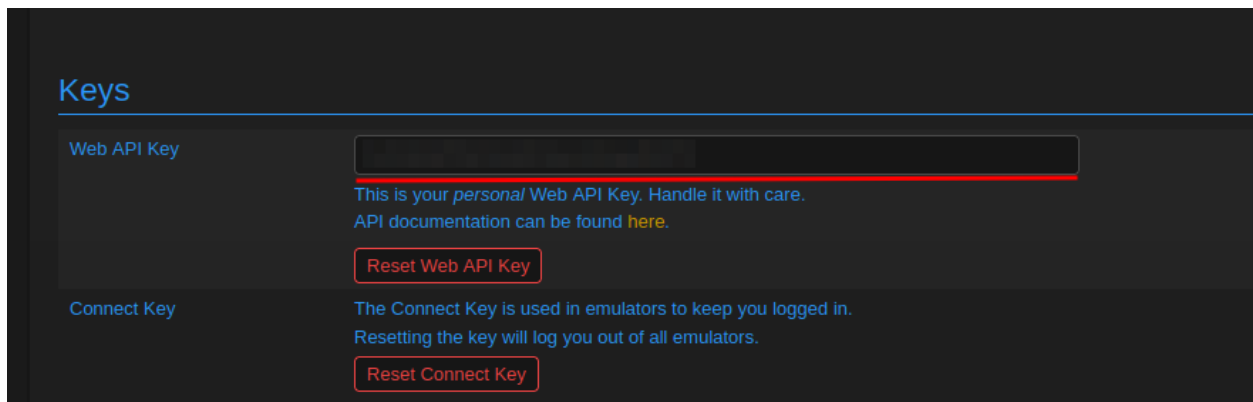
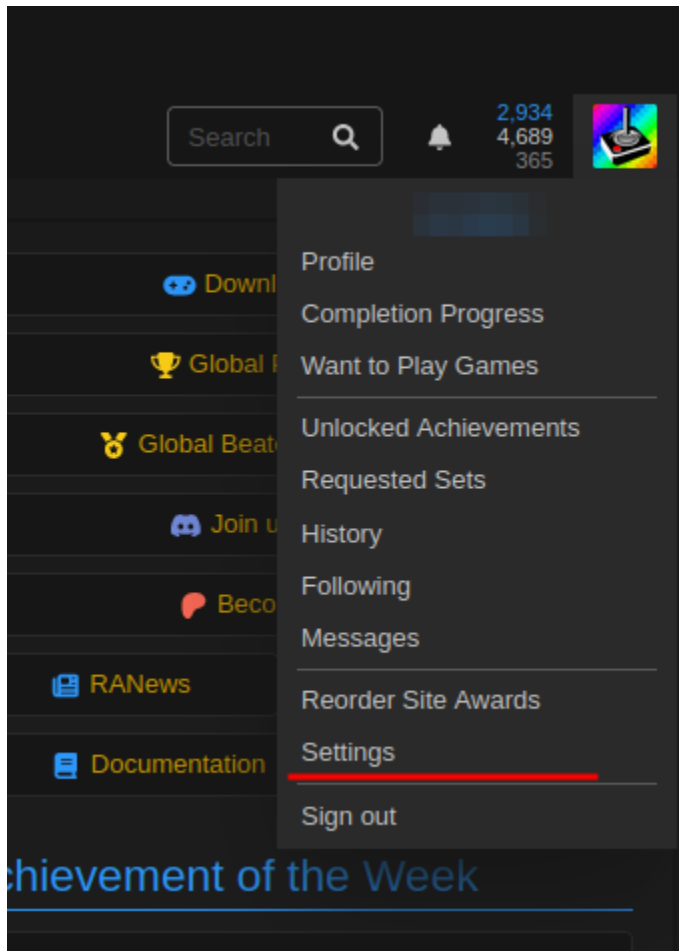
Warning: You must have a RetroAchievements account to use this integration. You can [sign up](#) for free. You also need to [setup your emulators properly](#), and ensure your game files are supported by RetroAchievements.

Open the **Settings** ( / ) panel, and navigate to the **Integrations** section. Find the RetroAchievements section.



The integration must be enabled for themes to be able to use it. You must also fill in your RetroAchievements username and web API key.

The web API key can be found in your settings panel on the RetroAchievements website.



THEME DEVELOPMENT

This section explains all the details on how to develop themes, as well as guides and tips.

2.1 Tutorial

This tutorial will guide you on the creation of a simple theme, from scratch.

RetroHub requires you to code every aspect of your theme. This includes how you present information, and interact with it. Therefore, it requires much more effort to create even a simple theme, when compared to other frontends where most of your theme is interpreted text from XML/JSON files.

The advantage, however, is that there's no restrictions to what your theme can be. This tutorial will guide you on creating a very simple UI layout, but you'll soon notice how you can control every aspect of how your theme looks and behaves.

2.1.1 Before starting

RetroHub runs using the Godot Engine. Therefore, a theme is essentially a Godot project, or scene. It's expected that you're familiar with the engine before proceeding with this tutorial. While it will guide you step-by-step, it won't explain any core concept important from Godot.

If you never coded in Godot before, we suggest you take some time learning it before starting with theme development. A few good resources to start with:

- [Your first 2D game](#): Step-by-step tutorial on how to create 2D content
- [Your first 3D game](#): Step-by-step tutorial on how to create 3D content, if you wish to create 3D themes.
- [Using signals](#): Lesson which teaches signals, a core concept of Godot which RetroHub depends on heavily.

2.1.2 Contents

Preparing the project

Before starting, you need to setup the necessary tools, and prepare the project to properly develop themes.

What you'll need

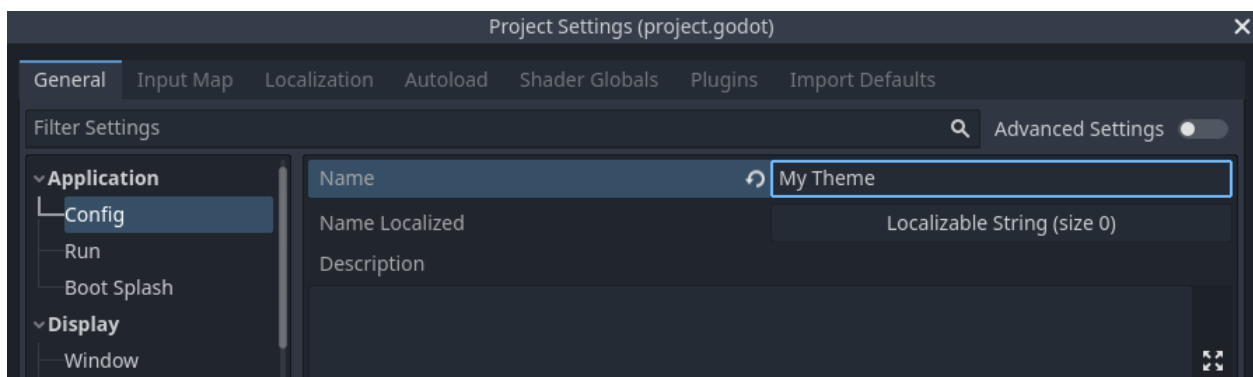
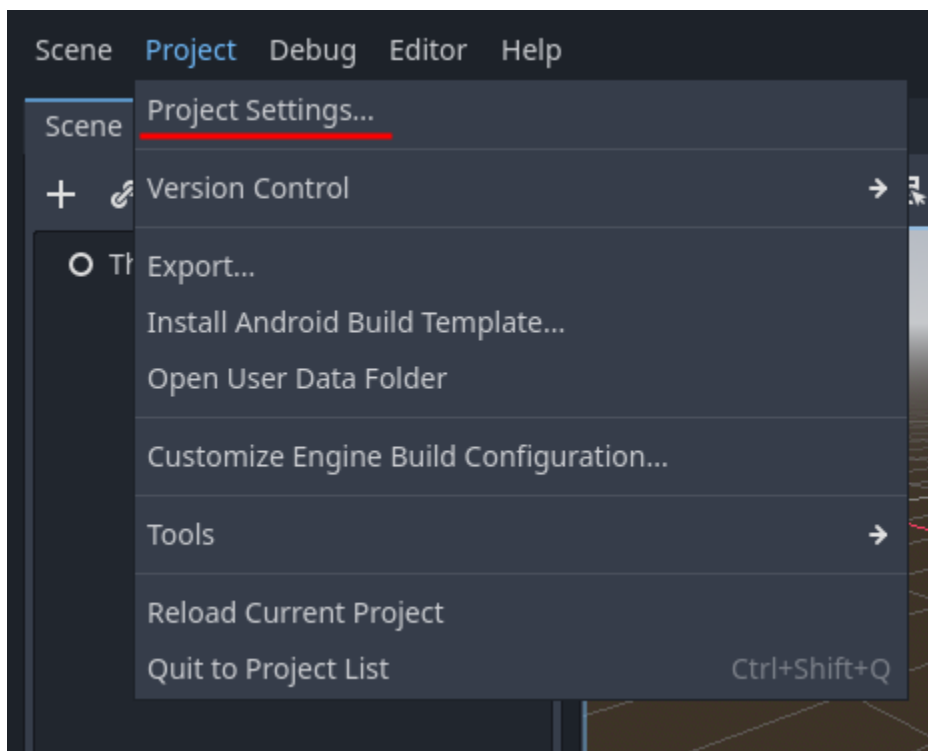
Firstly, you'll need to get the official [Godot Editor v4.2.1-stable](#). Although RetroHub uses a custom version of Godot, the themes can be developed on the official version.

Next, you need a project pre-configured with RetroHub's settings, such as input keys, addons, etc. . . For that, we created a bootstrap project, which is a base that can be used for developing any theme. Download [retrohub_bootstrap_theme](#) and open it in Godot.

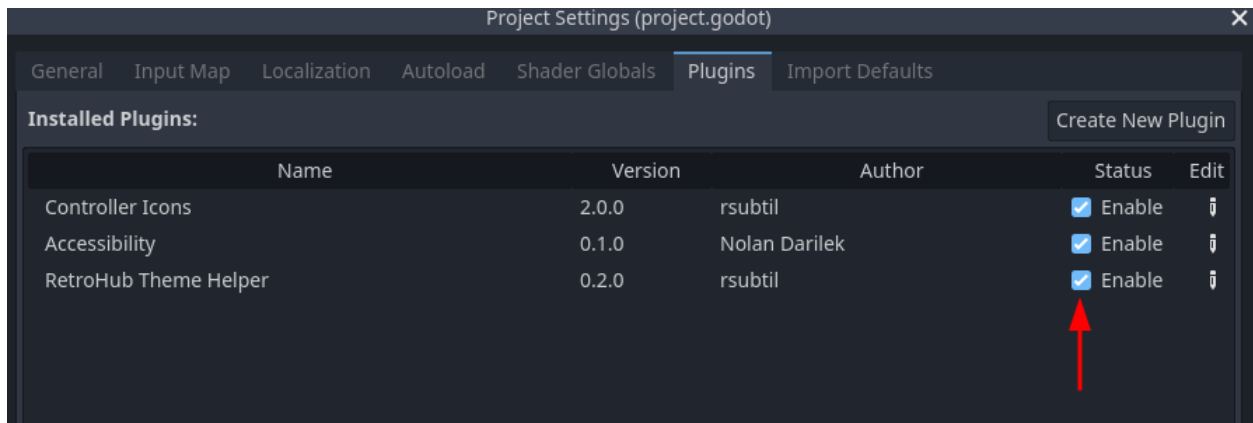
This project starts with many pre-applied configurations for RetroHub, but most notably it has an addon called `retrohub-theme-helper`. This addon not only exposes RetroHub's API methods and variables, it also provides various tools to help in theme development.

Configuration

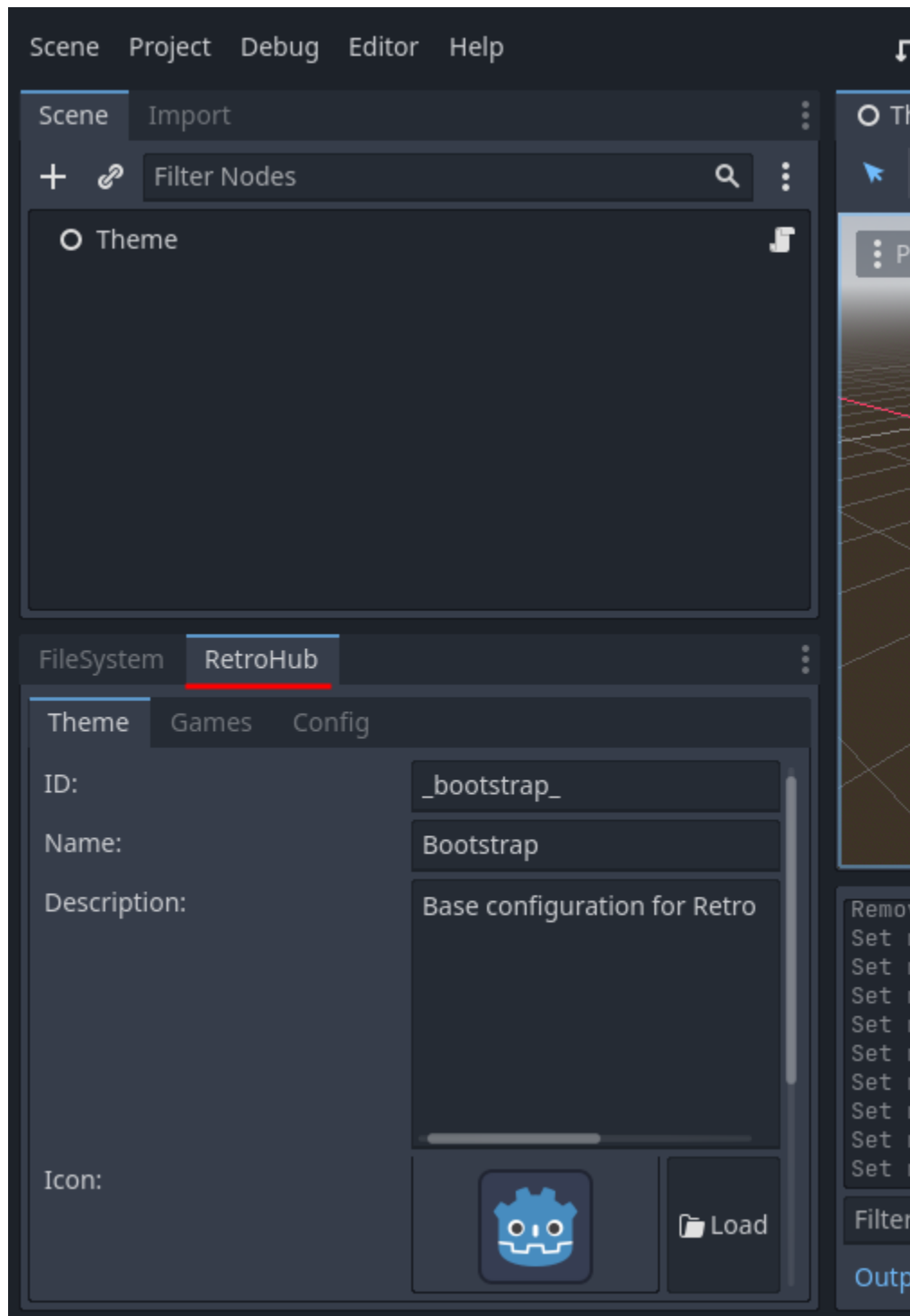
The first time you open the project, it will take some time to import resources. After that, you should change some settings to your liking. Start by setting your project name by going to **Project > Project Settings...**



- Ensure the RetroHub Theme Helper is enabled in the **Plugins** tab.

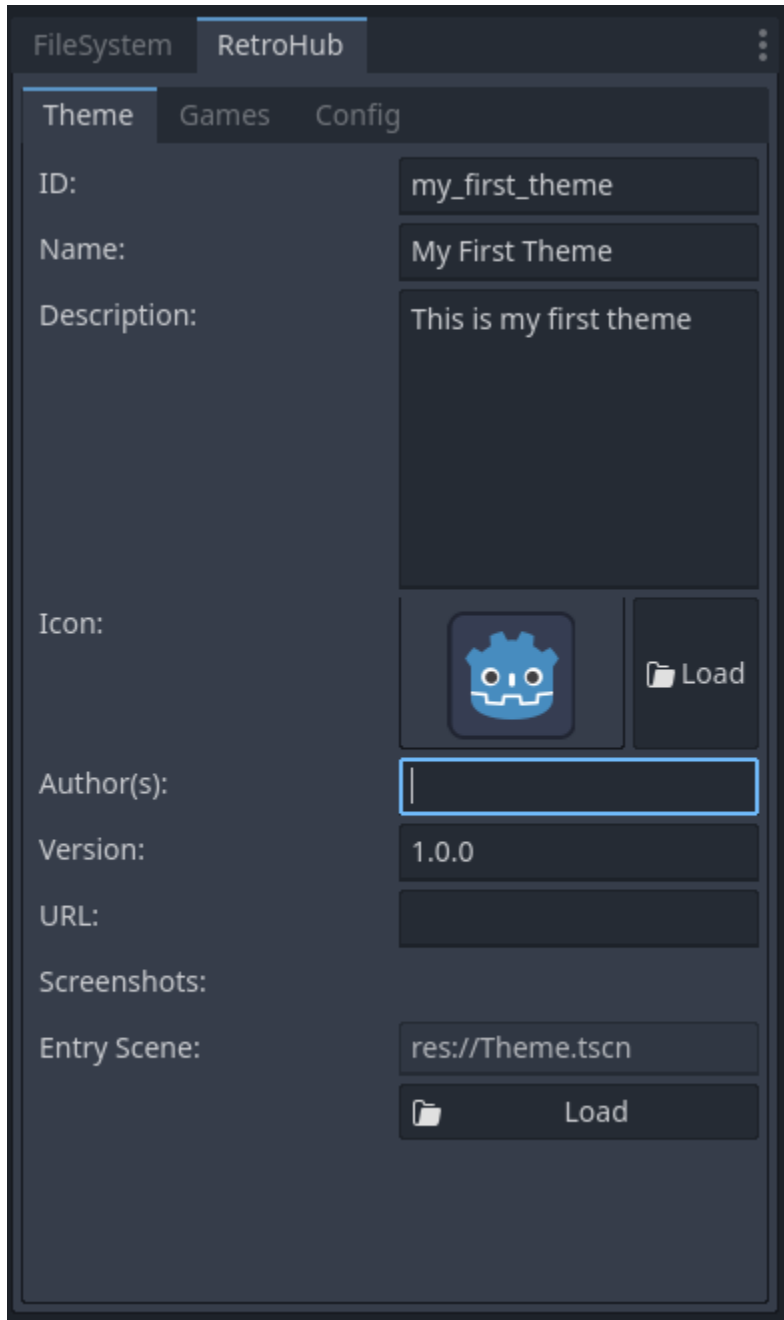


- Close the project settings, and move to the **RetroHub** tab, next to the **FileSystem** tab.



- Ensure the Theme tab is selected, and fill the information as follow:
 - **ID:** my_first_theme
 - **Name:** My First Theme
 - **Description:** This is my first theme.
 - **Icon:** Load the default icon.svg

- **Author(s):** *your name*
- **Version:** 1.0.0
- **URL:** *a personal URL of your choice, or leave the default value*
- **Screenshots:** *leave empty*
- **Entry Scene:** Load the main scene `Theme.tscn`



The screenshot shows the RetroHub application window with the 'Theme' tab selected. The window has a dark theme. The 'Theme' tab is active, with 'Games' and 'Config' tabs also visible. The configuration fields are as follows:

- ID:** my_first_theme
- Name:** My First Theme
- Description:** This is my first theme
- Icon:** A blue gear icon with a face, next to a 'Load' button.
- Author(s):** An empty text input field.
- Version:** 1.0.0
- URL:** An empty text input field.
- Screenshots:** An empty area.
- Entry Scene:** res://Theme.tscn, with a 'Load' button below it.

The ID is an identifier for your theme, and should be unique. In this instance it's not problematic, but in a release theme you'll need to get a unique ID to distinguish between all themes.

Entry scene is the scene that will be loaded first when the theme is loaded. It's essentially like setting the main scene when you run your project.

The remaining fields are just information about your theme.

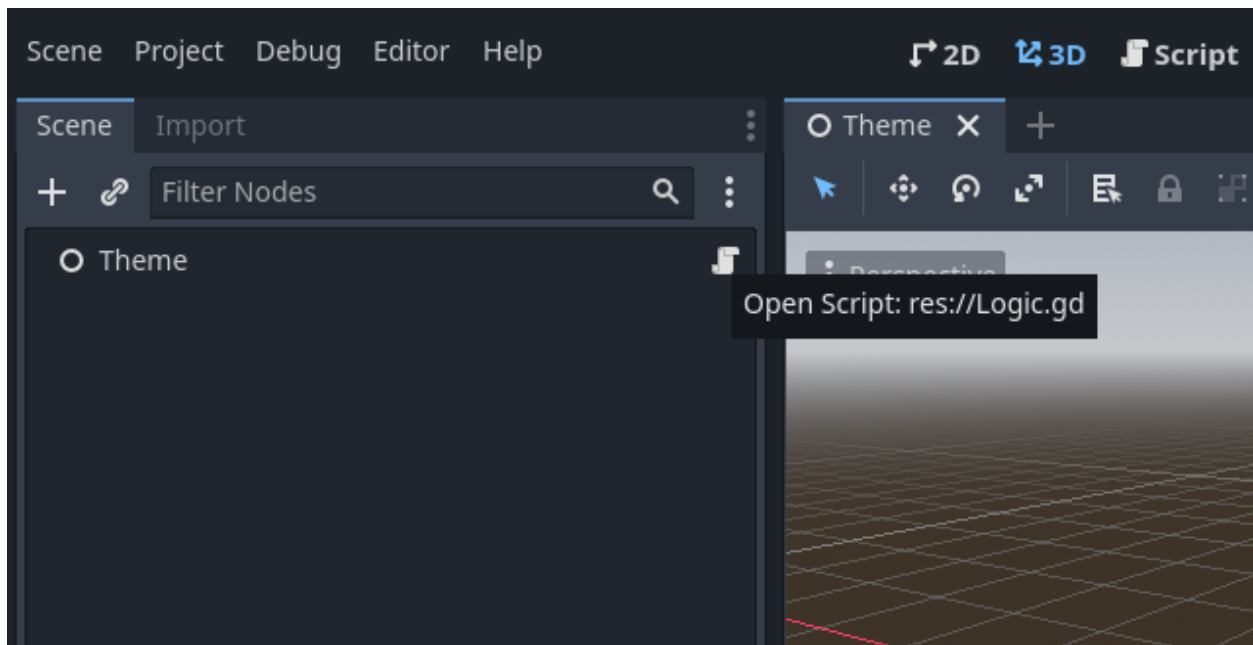
These settings are saved whenever Godot performs a save operation, such as exiting the editor or launching a scene. You can trigger a save at any time with the shortcut Ctrl + S.

All this information is stored in a `theme.json` file at the root of your project.

Structure

This project starts with a base structure set in place. We recommend following it for this tutorial, but you're free to structure your project as you prefer.

The main scene is called `Theme.tscn`. It contains just a scene root, with a `Logic.gd` script attached. This script connects to plenty of RetroHub signals, and has detailed comments explaining each signal logic.



Your theme should connect to these signals, as these are the mechanism RetroHub uses to present the user's gaming library to a theme. There's a lot of content here already, so don't worry about understanding what they all do. The most important ones are `system_received/game_received`; these signals together expose the full user's library, for system and game data respectively. In the next section we will take a better look on how they work.

System & Game Data

RetroHub shares information through data classes. There are two main ones you'll be working with: [RetroHubSystemData](#) for system information, and [RetroHubGameData](#) for game data. These arrive from `system_received` and `game_received` signals respectively.

If you look at the `Logic.gd` script, you'll notice there's some code for reading info from these data classes:

```
## Called when RetroHub has information of a game system available.
## It's entirely up to you how to display that system information.
## RetroHub only sends information from systems with detected games.
##
## System information always arrives before game information.
```

(continues on next page)

(continued from previous page)

```

func _on_system_received(data: RetroHubSystemData):
    print("System received: %s [%s]" % [data.fullname, data.name])

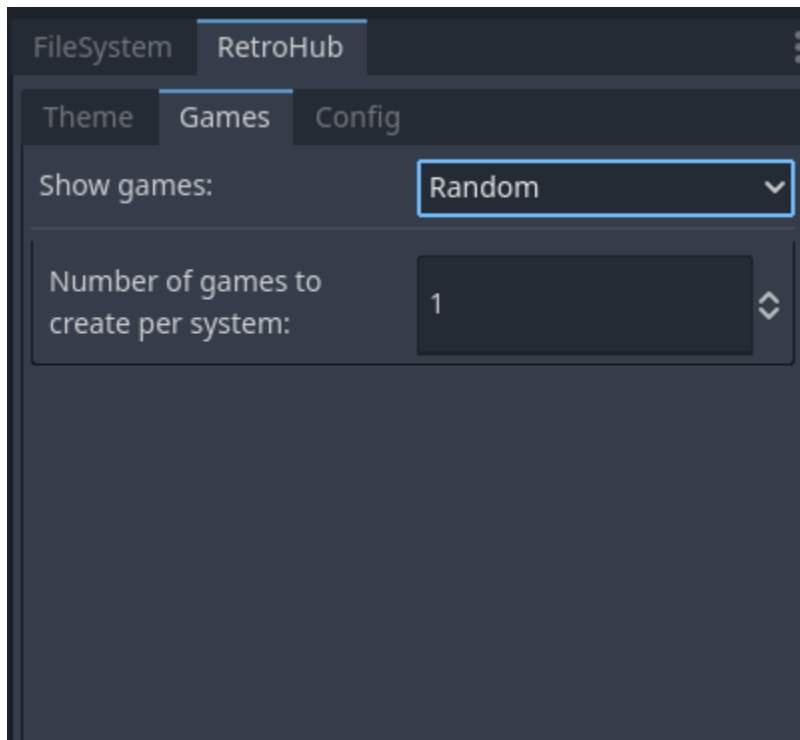
## Called when RetroHub has information of a game available.
## It's entirely up to you how to display that game information.
##
## Game information always arrives after system information.
func _on_game_received(data: RetroHubGameData):
    print("Game received: %s [%s]" % [data.name, data.system.name])

```

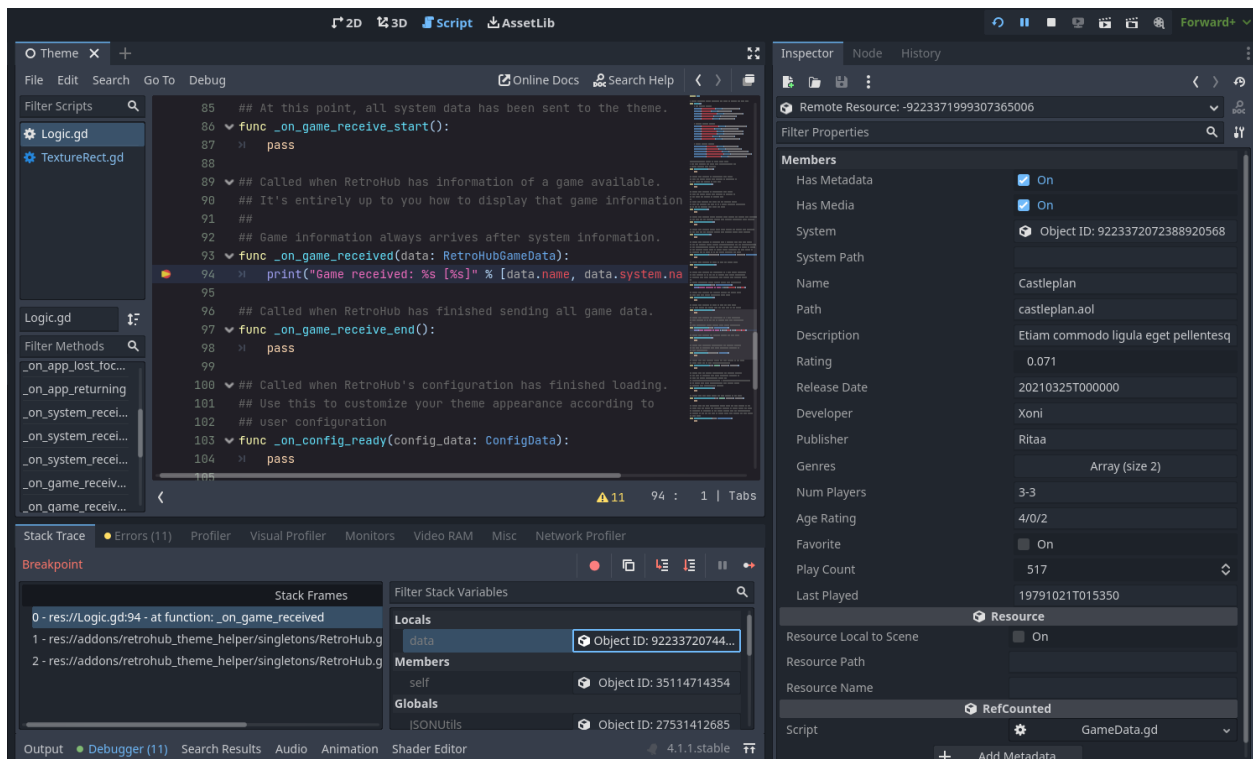
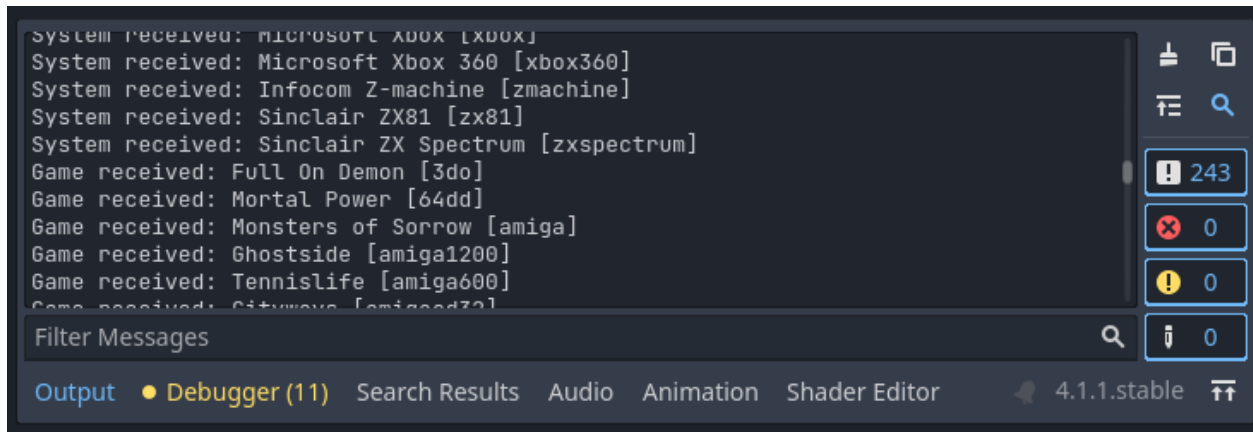
You should check these classes' definitions, either in app or from the API reference. You'll be using fields from it constantly to present information to the user.

If you run the project now, however, no information will be received. This is because the theme helper settings are set by default to not have any information available, i.e. the user has an empty library.

Change the settings (**RetroHub > Games**) to **Random**, and set the number of games per system to **1**. This setting will use real system data, but generate random game information. That way you can test for every system available without requiring you to own games for each system.



Run the project now and you'll notice you get some text output with information now. If you want to check each data class, you can breakpoint on each method and analyze the content of the data classes.



These two data classes are the most important. In this next section, we'll learn how to use them and how to present their information to the user through a simple UI.

Note: From now on, the tutorial will assume this setting is set to **Random**. However, if you have your own gaming library already, you can set to **Local** instead to test the theme with your gaming library.

Building UI

Now that we have system and game information, it's time to parse them and present to the user.

This is the part where themes will massively diverge. How you present this information to the user is entirely up to you. You can create a UI dropdown list for the user to select a game. Or [create a pretty retro console style selection screen](#). Or [spawn a 3D cartridge object the user has to insert in a console](#). Or you can make a [DOOM clone where you shoot the game to launch it](#). The possibilities are endless!

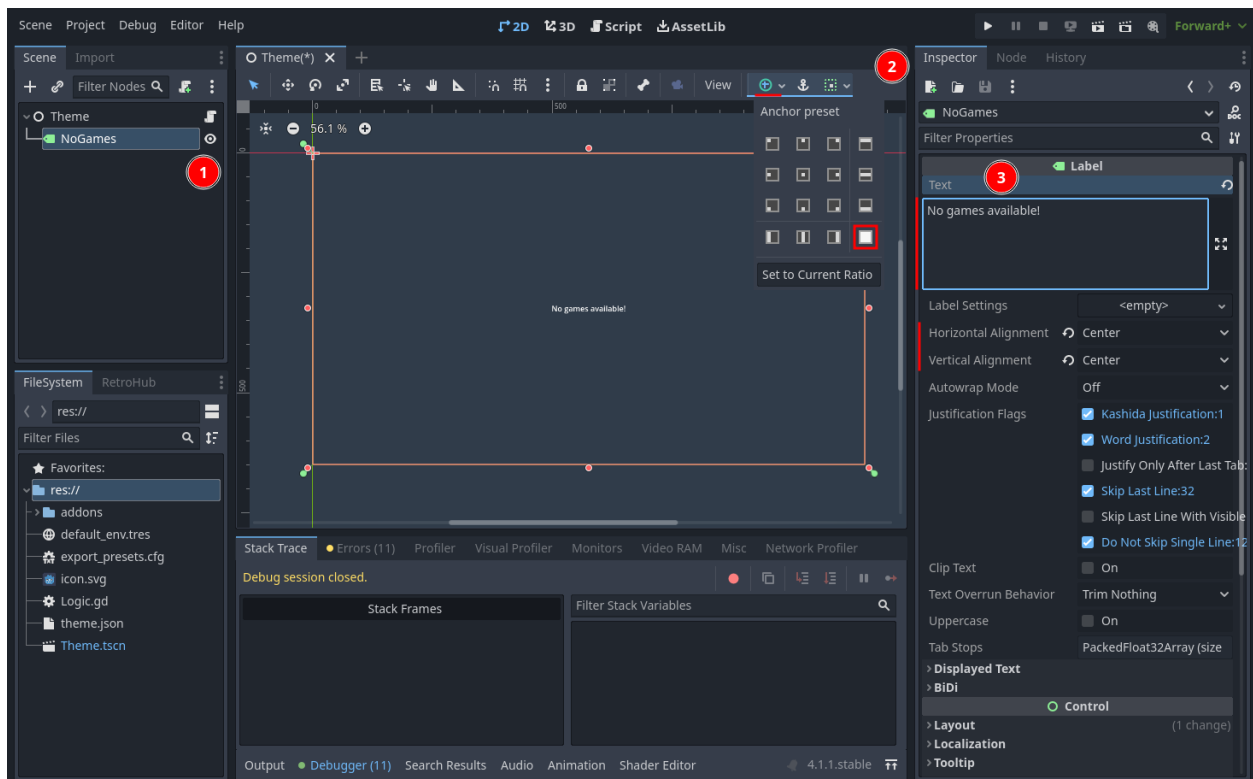
For this tutorial, we will make a simple UI where games are grouped by system, and the user can view some game metadata before launching it.

Handling empty libraries

Before we begin, we have to ensure we handle an edge case properly; the user might not have any game available. In that instance, RetroHub will never send system/game data.

So, we can have our scene show by default a message for when no games are available, and remove it as soon as system/game data is received.

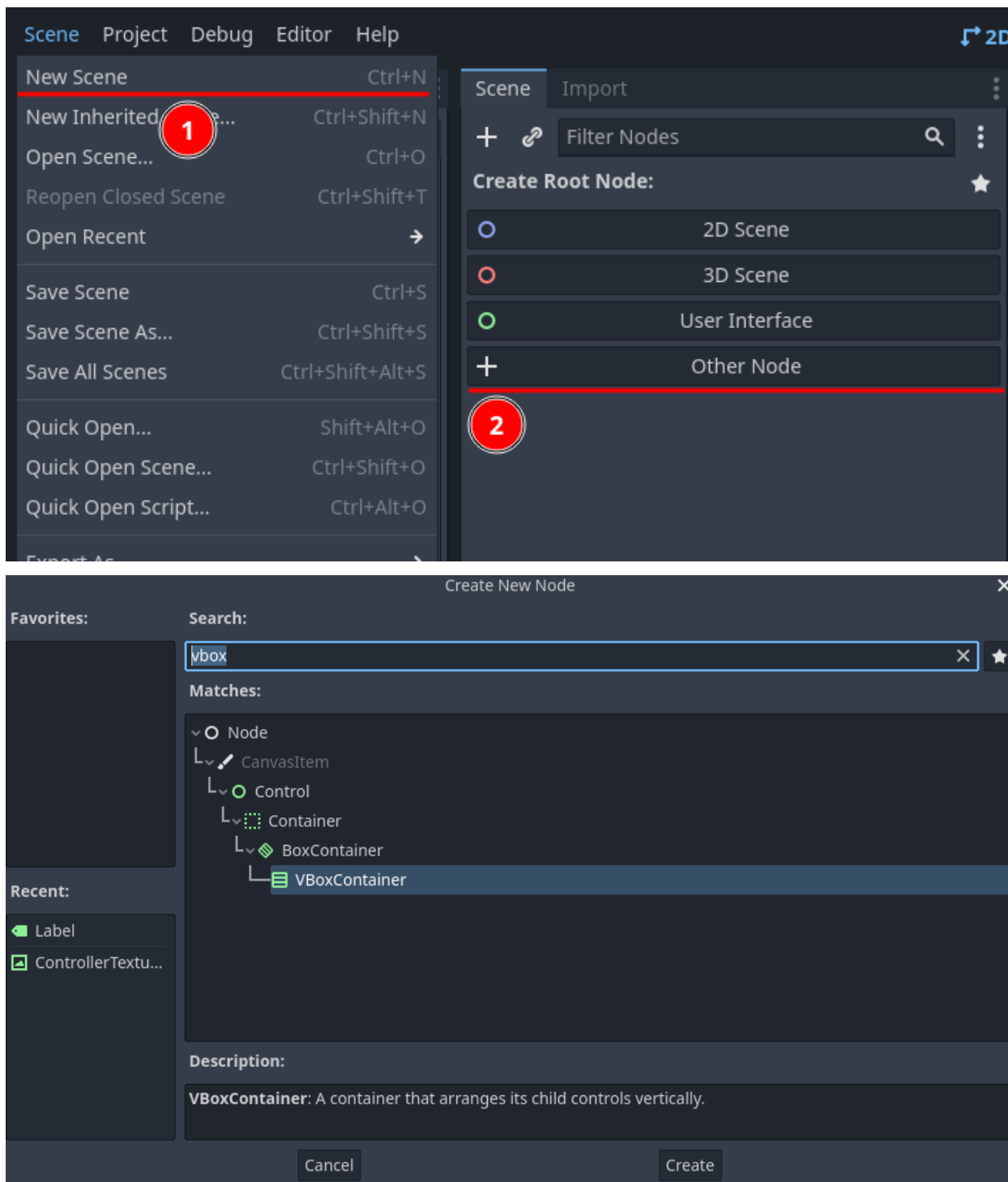
Add a new **Label** to the theme, called **NoGames**. Set it's layout to fill the entire screen, center the content, and write the warning message, such as "No games available!"



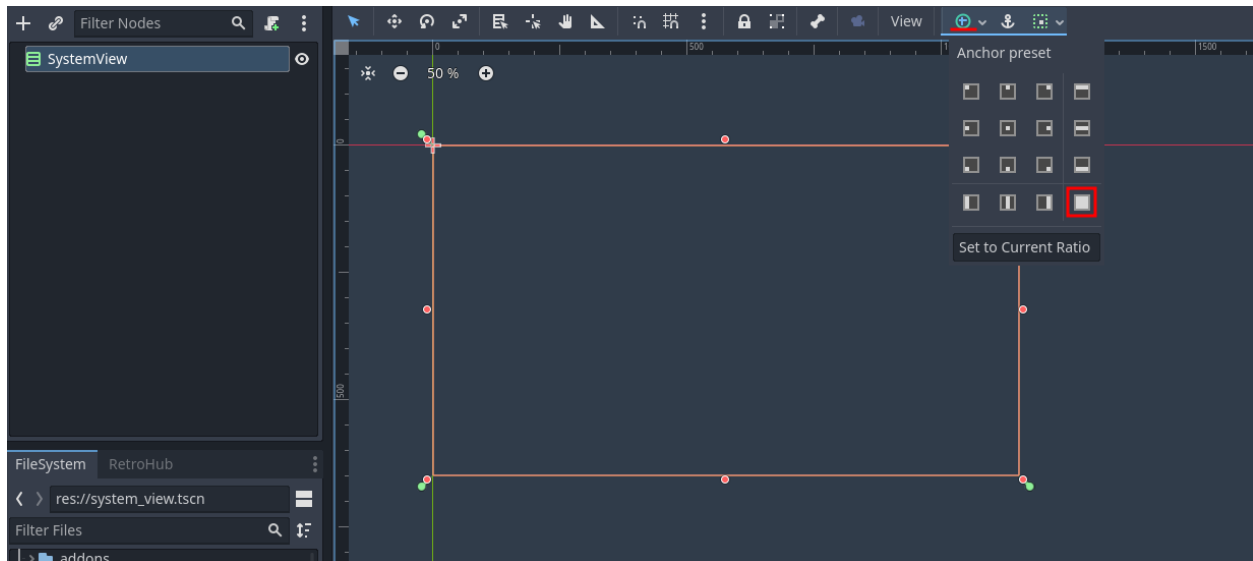
This is enough to warn the user. Ensure this is what appears if you try running the project with the theme helper settings games setting to **None**.

Creating system views

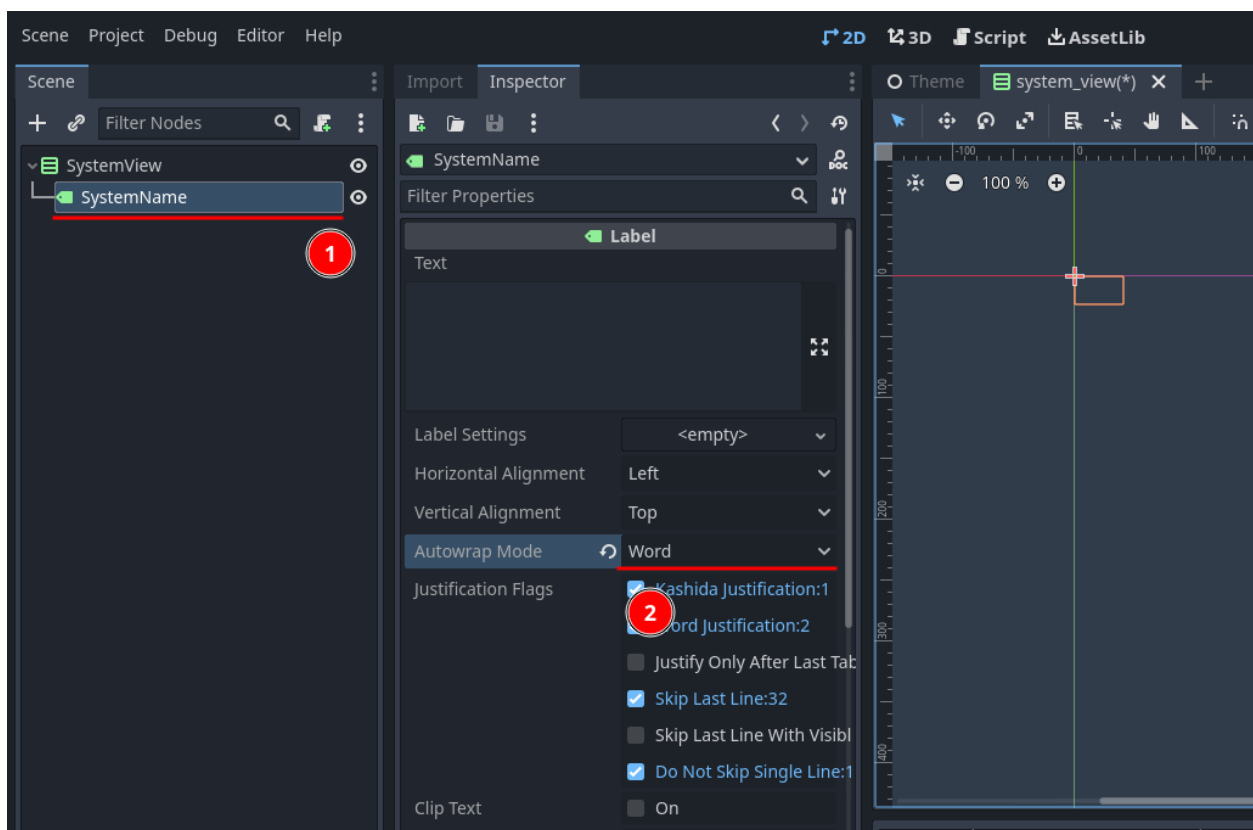
Let's start by properly showing each system name. Let's create a new scene called `system_view.tscn`, inheriting from **VBoxContainer**:



Set it's layout to fill the entire screen as well.



For now we will just show each system full name. Add a label to it called **SystemName**, set its **Autowrap** property to **Word**, and leave it empty (the content will be changed in code).



Now add a script to the root **SystemView**, which will be called **system_view.gd**. Let's add a way for easily setting the system name from outside this scene:

```
extends VBoxContainer

@onready var system_name_label := $SystemName
```

(continues on next page)

(continued from previous page)

```

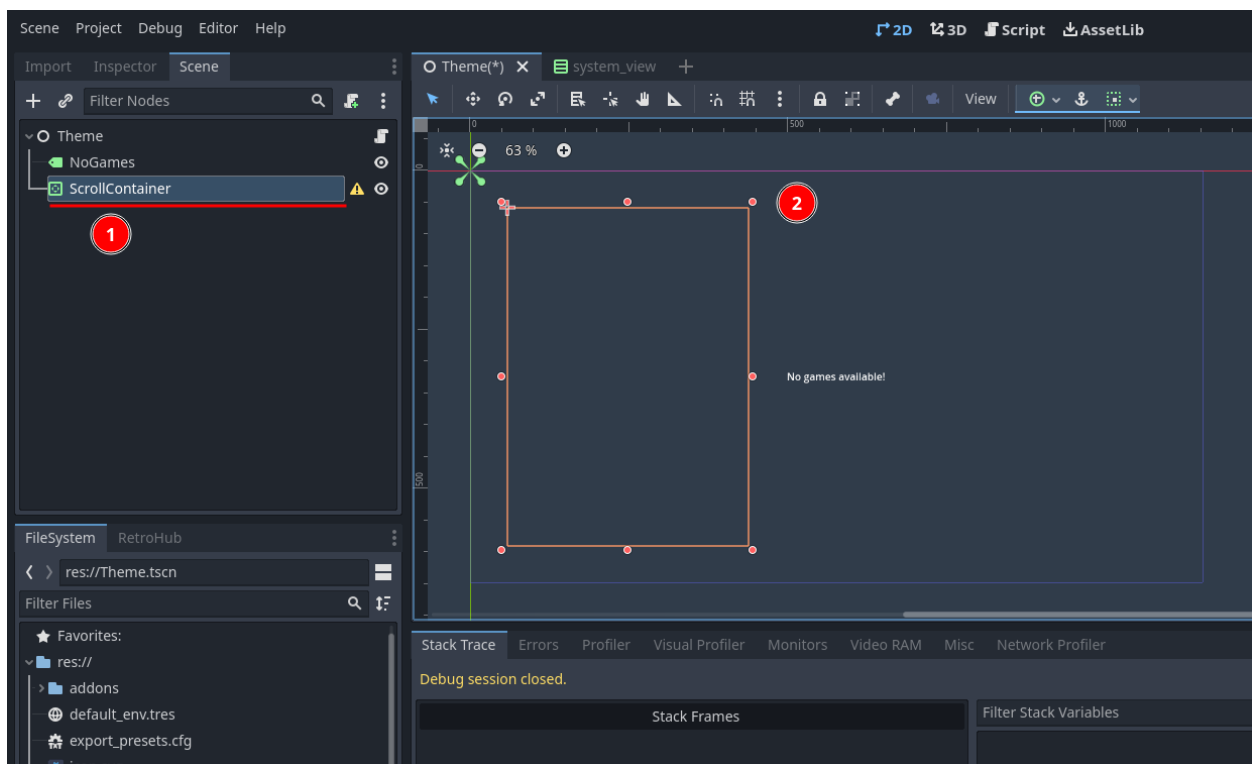
var system_data : RetroHubSystemData:
    set(value):
        system_data = value

func _ready():
    system_name_label.text = system_data.fullname

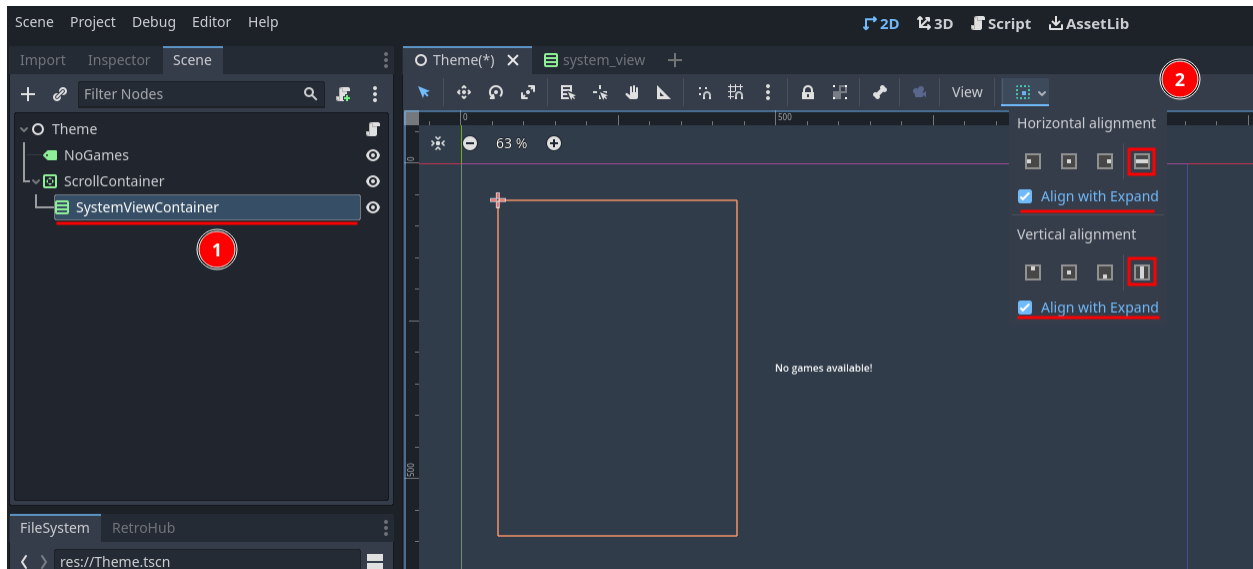
```

Now, we need to instance this object at runtime when each system is received. Go back to `Theme.tscn`. We may need to instance a lot of scene views and we want them to be presented neatly. For that we're gonna create a **VBoxContainer** to add them vertically, and a **ScrollContainer** to create scroll bars if the content doesn't fit on screen.

Start with the **ScrollContainer**, and give it a good amount of vertical size, while only occupying horizontally about 30% of the screen, to leave space for game metadata later on.



Then add a **VBoxContainer** to it. Rename it to **SystemViewContainer** so it can be accessed in the script. Lastly, set its size flags to expand so it occupies all the space the scroll container provides.



Now, edit the script under the **Theme** root node. Add references to the nodes we need: the **NoGames** label so we can remove it when there's content, and the **SystemViewContainer** to add **SystemView** instances to it:

```
extends Node

@onready var no_games_label := $NoGames
@onready var system_view_container := $ScrollContainer/SystemViewContainer

# _ready function, called everytime the theme is loaded, and only once
func _ready():
    # App related signals
    RetroHub.app_initializing.connect(_on_app_initializing)
    ...
```

Then move to the existing `_on_system_receive_start` function. This function is called right before RetroHub starts sending all system data, so it's the perfect place to remove our **NoGames** label:

```
## Called when RetroHub is about to send all system data.
func _on_system_receive_start():
    no_games_label.queue_free()
```

Now, let's start creating our **SystemView** instances. Edit the `_on_system_received` function right below:

```
## Called when RetroHub has information of a game system available.
## It's entirely up to you how to display that system information.
## RetroHub only sends information from systems with detected games.
##
## System information always arrives before game information.
func _on_system_received(data: RetroHubSystemData):
    var system_view = preload("res://system_view.tscn").instance()
    system_view.system_data = data
    system_view_container.add_child(system_view)
```

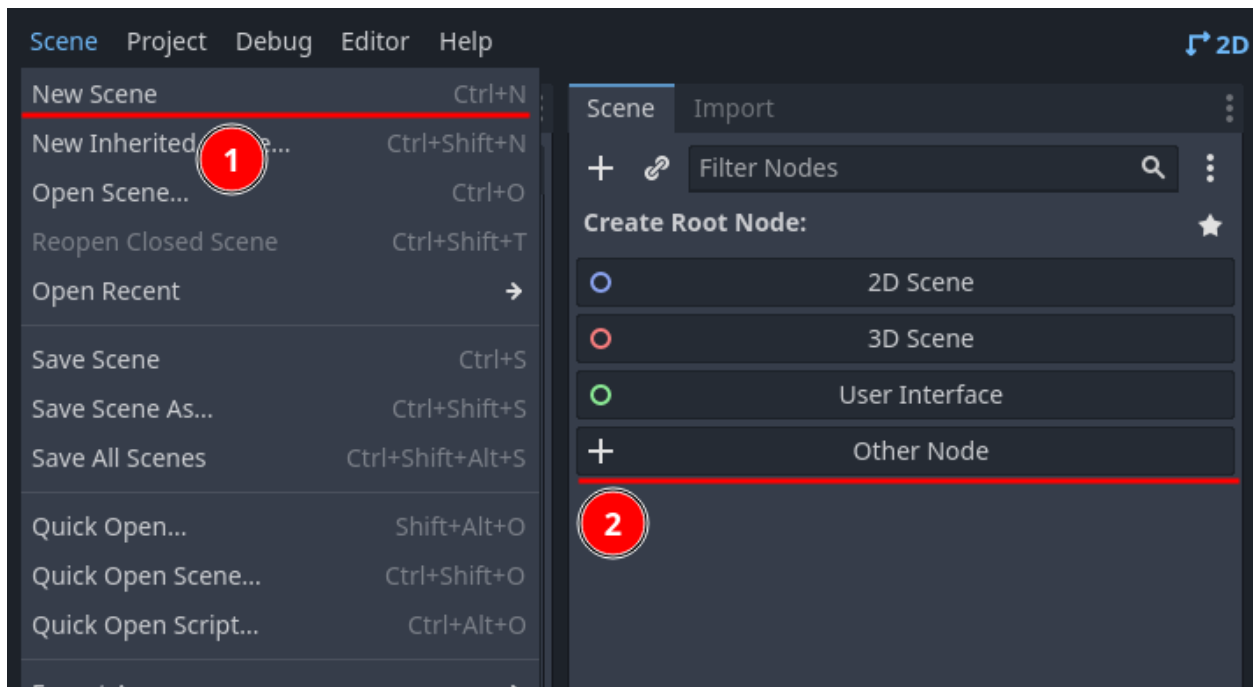
Go ahead and run your project. You'll now see a list of system "pretty names". The warning label about an empty library is gone as well.

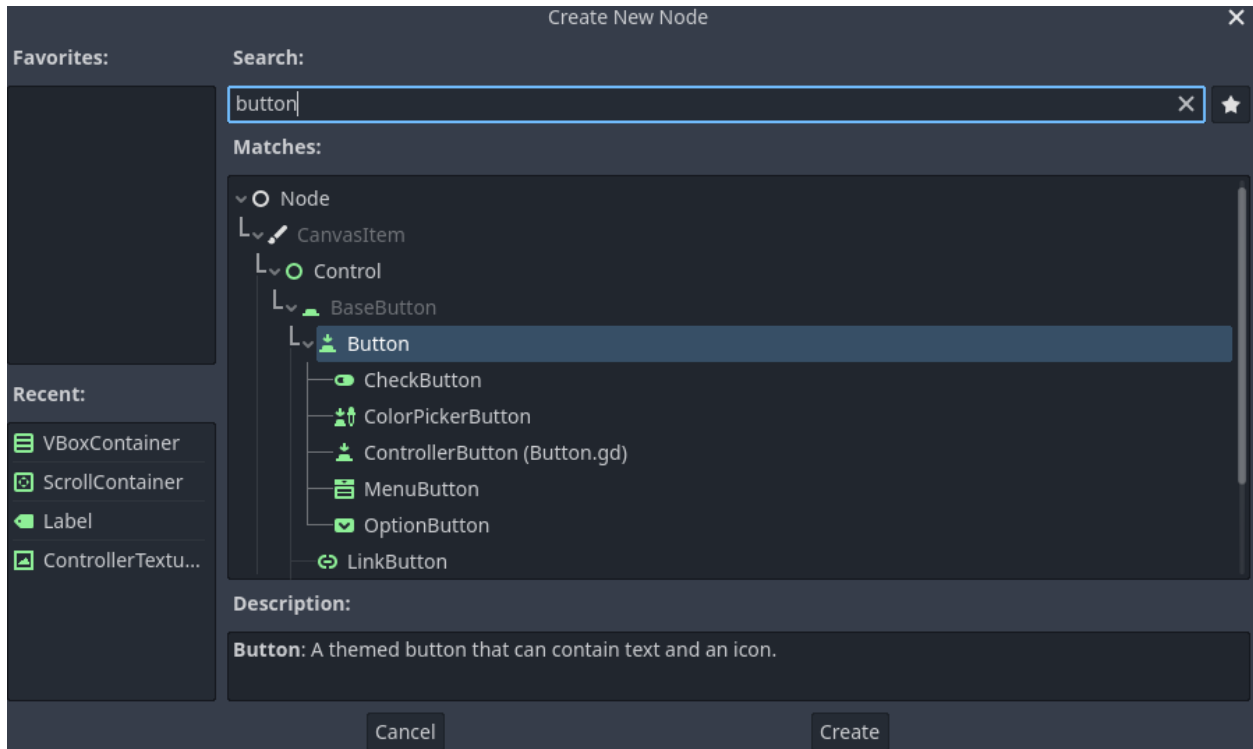


Creating game entries

We have a scene that handles system information. So, we can have it also handle any game data from that system. Let's make it so that each game data shows up as a button under each system name.

Create a new scene called **game_entry.tscn**, this time inheriting from a **Button**.





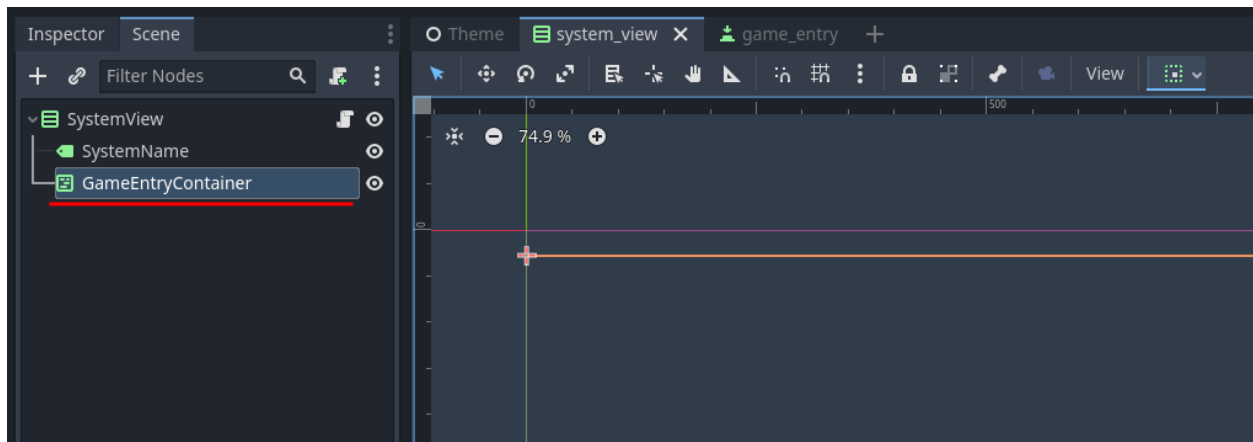
Add a script to it, which will be called **game_entry.gd**. The process is similar to what we did in the system view; we receive data, and set the label to some information:

```
extends Button

var game_data : RetroHubGameData:
    set(value):
        game_data = value

func _ready():
    text = game_data.name
```

Let's prepare the **SystemView** scene to properly handle **GameEntry** instances. Add a **HFlowContainer** after the **SystemName** label, and rename it to **GameEntryContainer**. This container looks better when rearranging children with different sizes, which happens as games have titles with different lengths.



Now, instead of instancing game entries from the **Theme.tscn** like we did for system views, we'll do a trick so our code is more organized. RetroHub works by connecting to signals, and the main **Logic.gd** script connects to all of them; but that doesn't mean we can't connect from other places! So, let's connect to the `game_received` signal in our **SceneView.gd** instead, and instance the game entries there:

```
extends VBoxContainer

@onready var system_name_label := $SystemName
@onready var game_entry_container := $GameEntryContainer

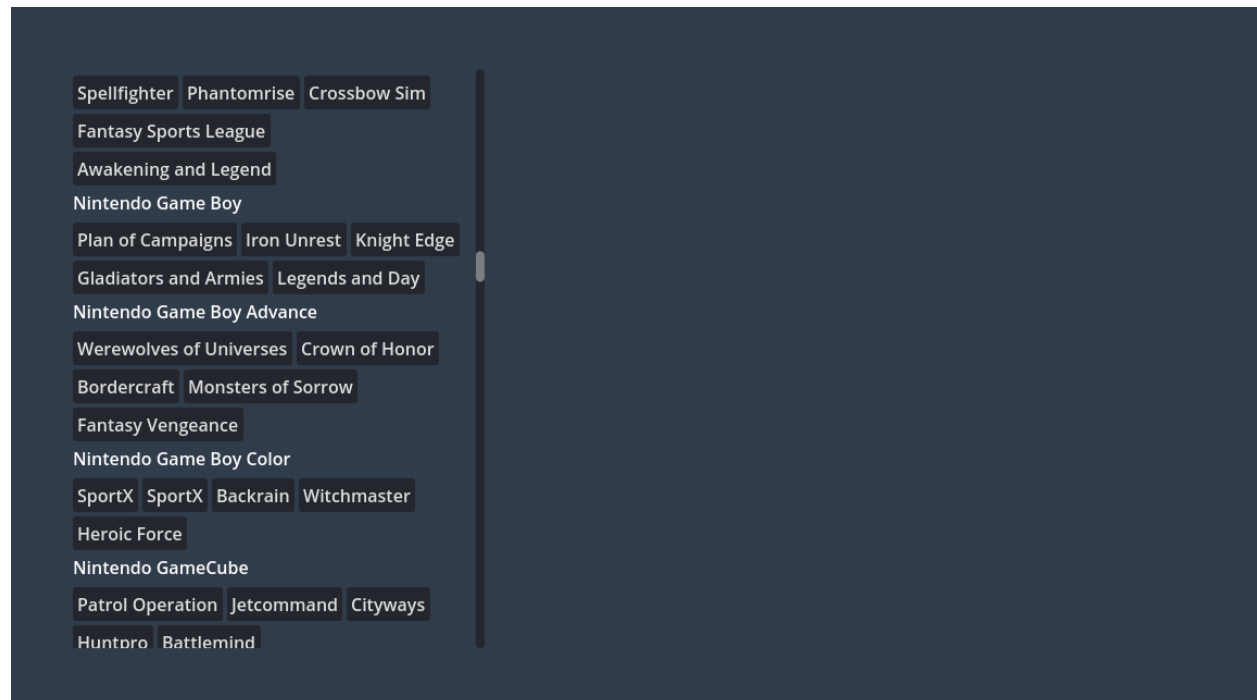
var system_data : RetroHubSystemData:
    set(value):
        system_data = value

func _ready():
    system_name_label.text = system_data.fullname
    RetroHub.game_received.connect(_on_game_received)

func _on_game_received(game_data: RetroHubGameData):
    if game_data.system == system_data:
        var game_entry = preload("res://game_entry.tscn").instantiate()
        game_entry.game_data = game_data
        game_entry_container.add_child(game_entry)
```

Every game data keeps a reference of the system it comes from, so you can use this to check whether this game data belongs to the current system view.

If you now run the project, you'll now see buttons below each system name, with each game title. Try increasing the amount of games received to 5 or more, and see how the buttons get rearranged!



We are very close to having a theme that can launch games already! We could do it right now with the right API call, but let's wait until next section to create the metadata viewer, so we have a proper UI element for launching games.

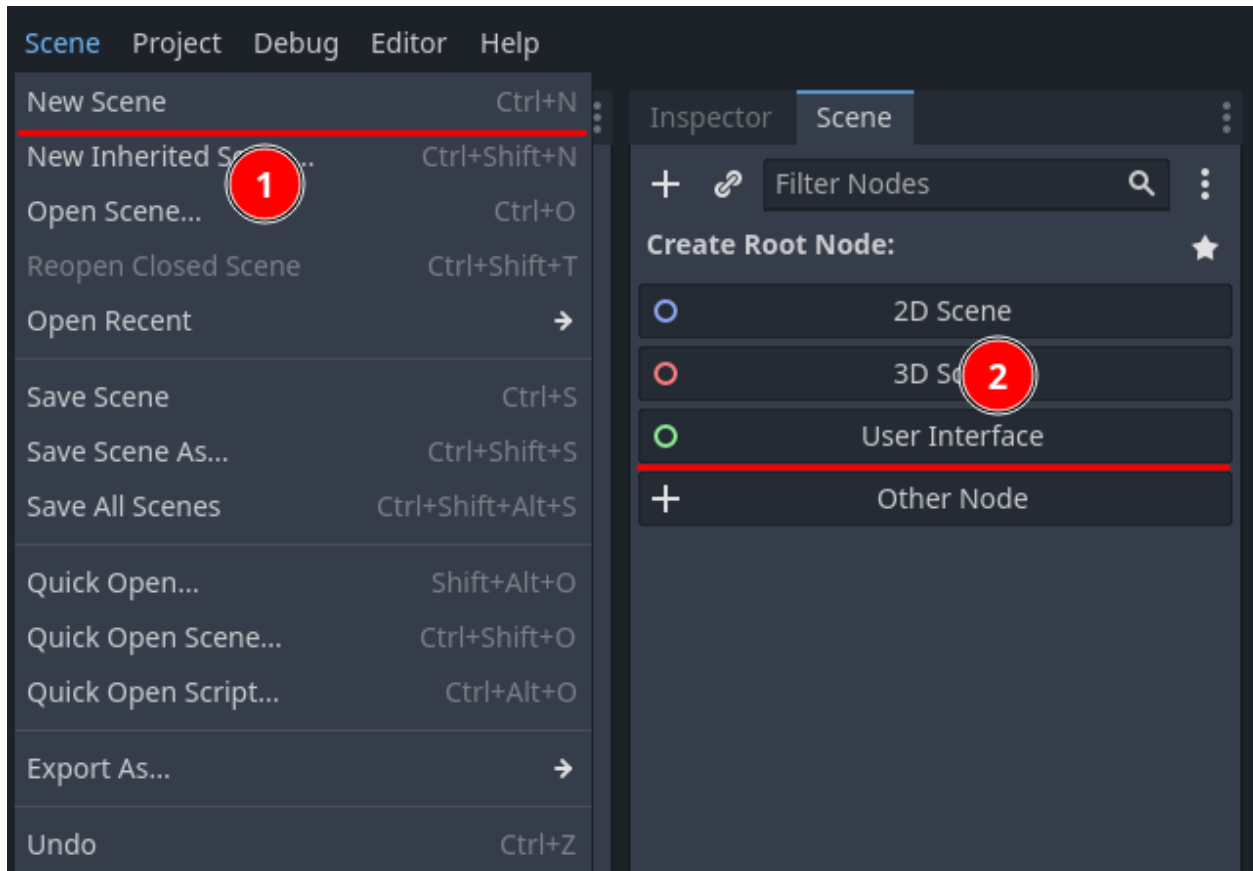
Warning: Don't forget that the base code connects to the `game_received` signal from **Logic.gd** and prints each game name for debugging purposes. As we now have a proper way to handle game data, you can now remove that connection to stop getting all that output data to the console.

Presenting metadata

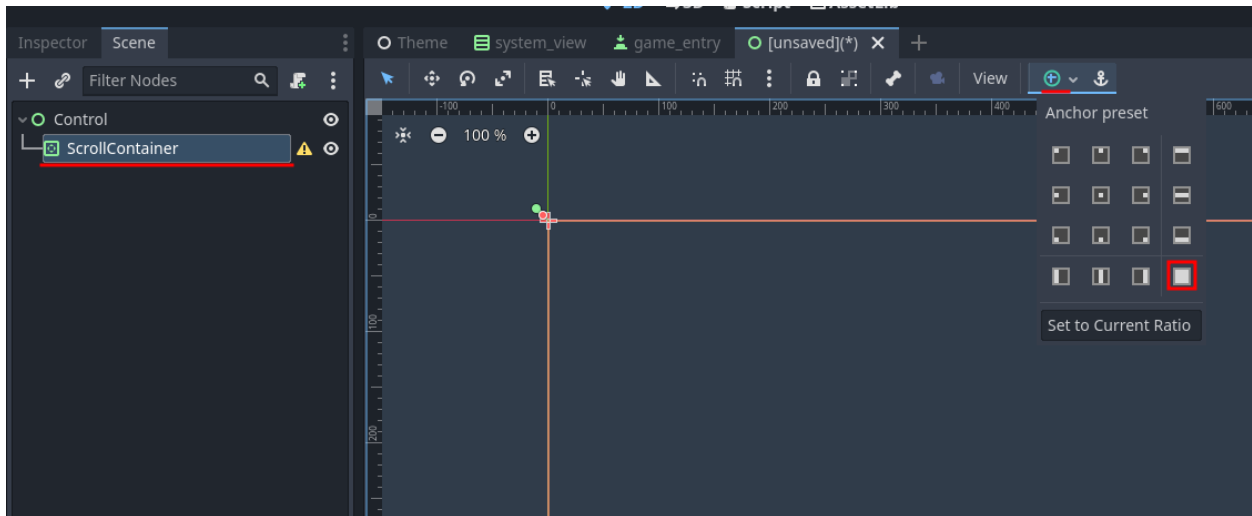
We built a simple UI, but it's already showing games for the user to pick and launch. Let's make it more complete by showing information about the game, such as title, description, and rating, when the user picks them.

Creating metadata viewer

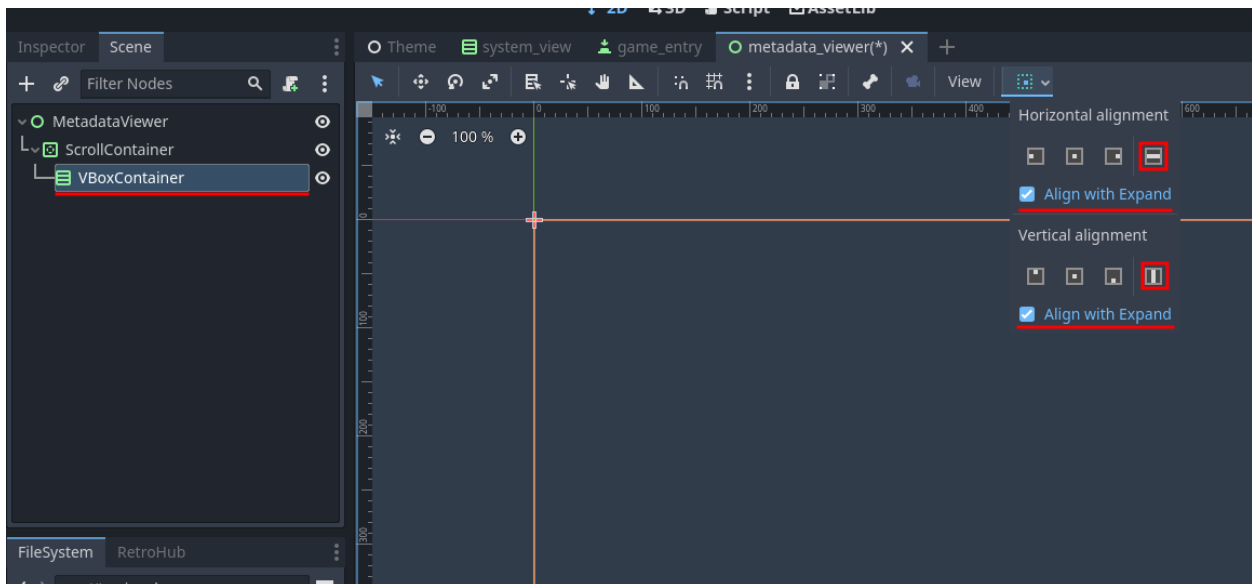
Let's create a new scene, called **metadata_viewer.tscn**. Make it inherit from **User Interface** (*aka Control*); this node will be added to the main theme, so there's no need to handle layout in this scenario.



Similarly to what we did for system views, let's add a **ScrollContainer** to add scroll bars if the content doesn't fit on screen. Add a **ScrollContainer**, and set its layout to **Full Rect**.



We will have multiple objects inside this container, so add a **VBoxContainer** to order them vertically. Set it to **Expand** as well.

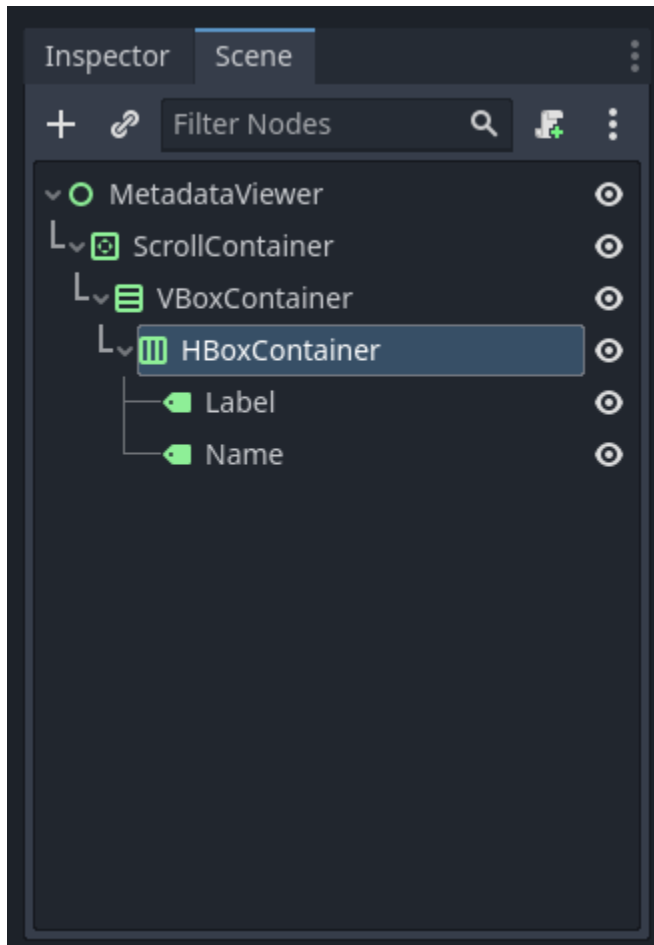


Now we start building the UI for showing all the information we want. *RetroHubGameData* has plenty of variables available, and because building an UI for all of them is time-consuming, we will just focus on a few of them: name, description, release date and rating.

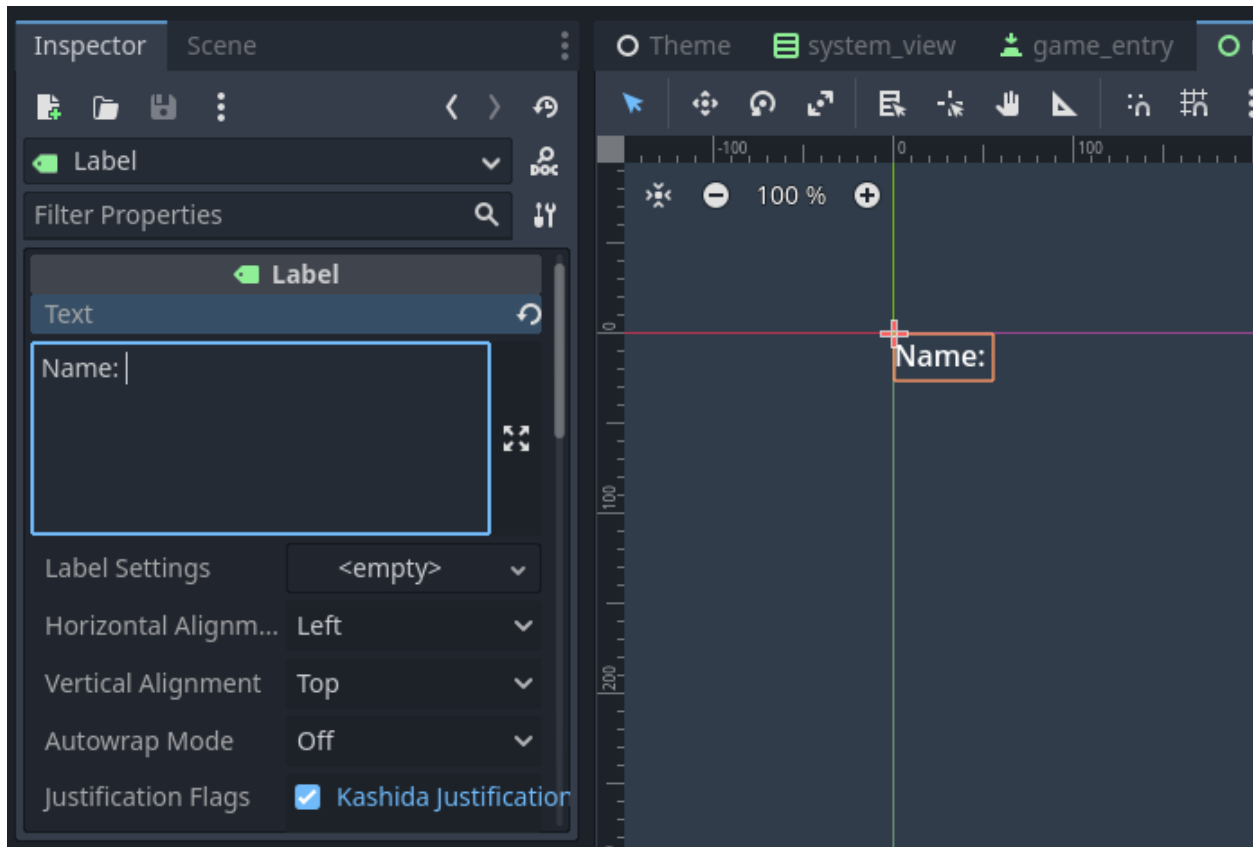
Creating a name field

Let's start by building a base structure for showing the name. We can then duplicate it for the other fields and change each structure a bit for speeding up this process.

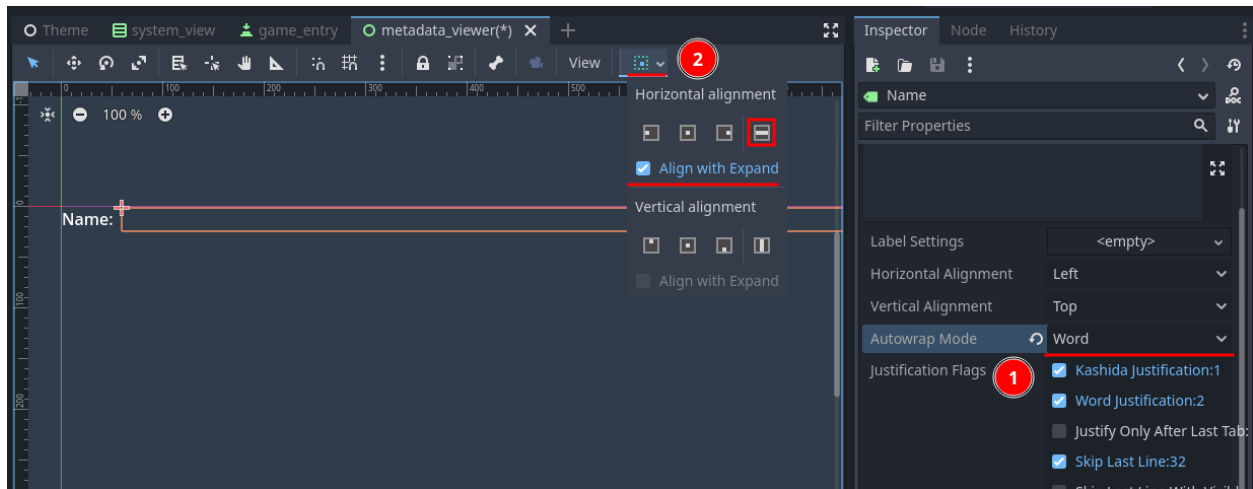
Create a **HBoxContainer**, and inside it create two labels: **Label** (you can leave the default name) and **Name** (it's this one we will change in code).



The **Label** will just describe the name of this field, so set it's content to Name: (add a space at the end so the text next to it isn't "glued" to the left).



The **Name** label will be left empty, as it will be set in code. However we need to change some settings. Set **Autowrap** to **Word** to prevent very long lines of text making this label huge. To ensure it also gets all horizontal space available, set the horizontal size flag to **Expand** as well.



Now would be a good idea to test this before moving on. It's going to take a bit of work to connect this scene to the theme in the end, so we'll cheat for now by generating a random game data in here, and using it instead.

Create the script at the root **MetadataViewer** object, which will be called **metadata_viewer.gd**. We add the nodes we're interested in editing, and change them when game data arrives:

```
extends Control
```

(continues on next page)

(continued from previous page)

```

# Nodes to edit
@onready var name_label := $ScrollContainer/VBoxContainer/HBoxContainer/Name

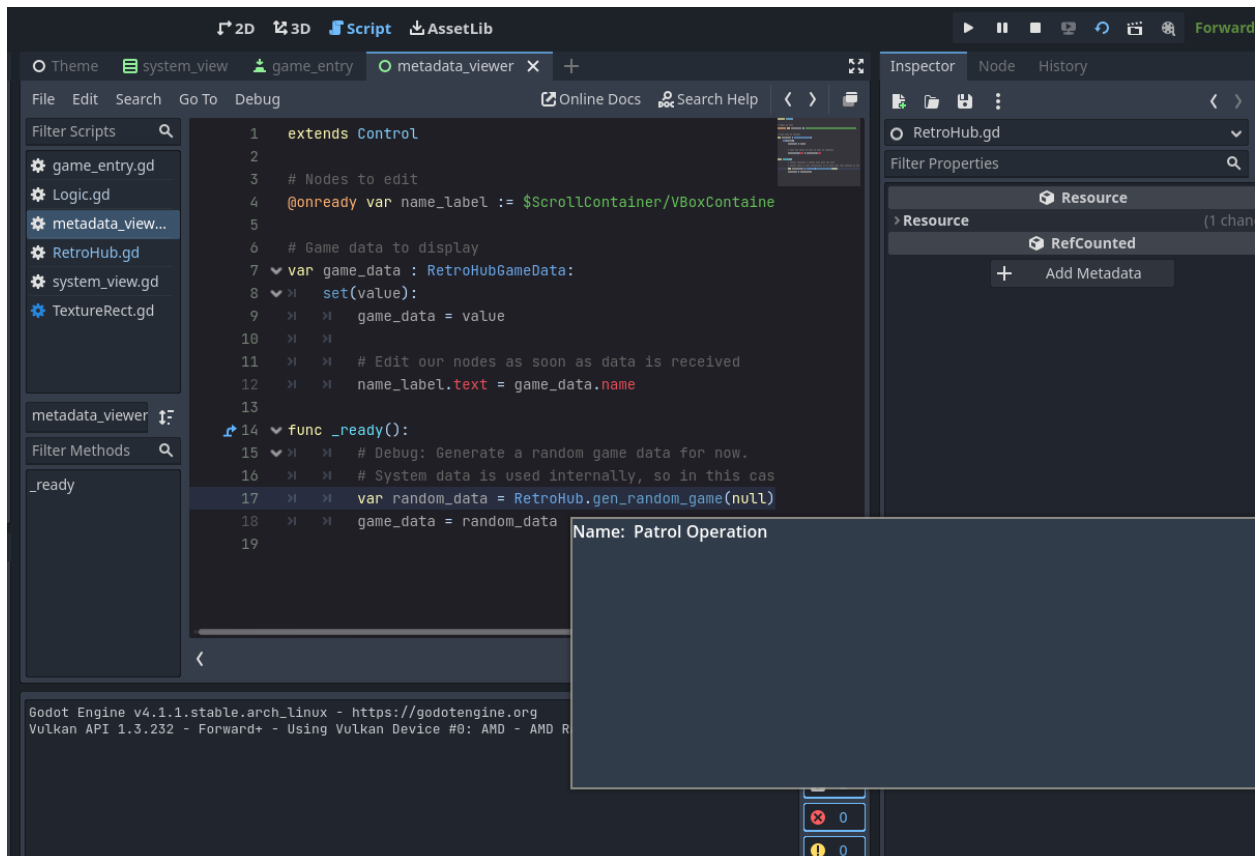
# Game data to display
var game_data : RetroHubGameData:
    set(value):
        game_data = value

        # Edit our nodes as soon as data is received
        name_label.text = game_data.name

func _ready():
    # Debug: Generate a random game data for now.
    # System data is used internally, so in this case it's alright to set to null
    var random_data = RetroHub.gen_random_game(null)
    game_data = random_data

```

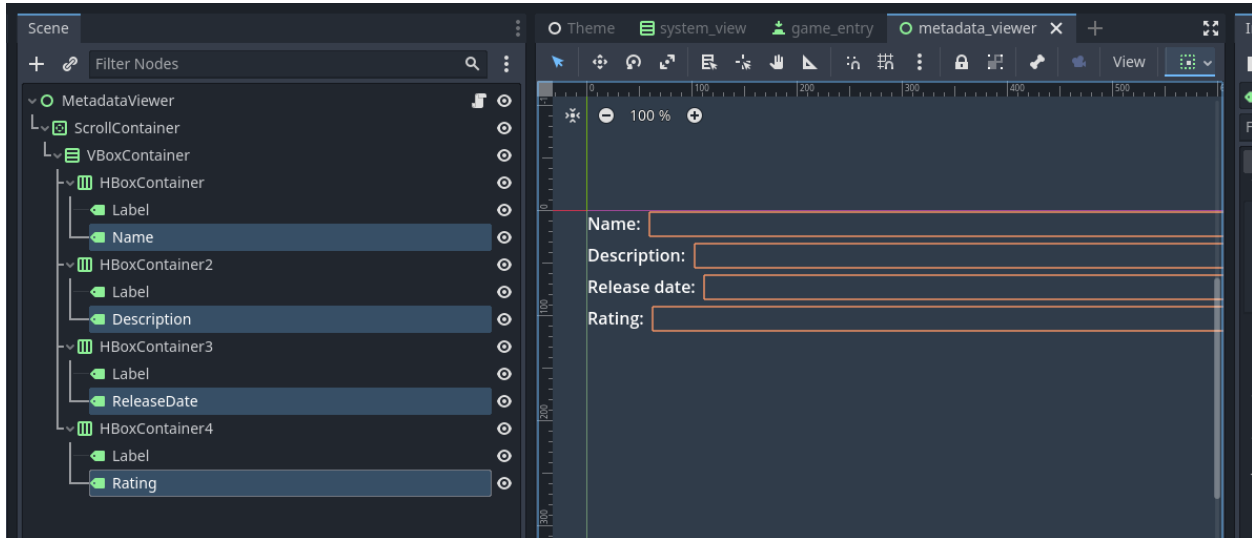
You should now run this scene instead of the whole project. Check that there is a name present when you run it.



Showing the remaining content

Our name object works perfectly! Let's go ahead and create the remaining fields: description, release date and rating.

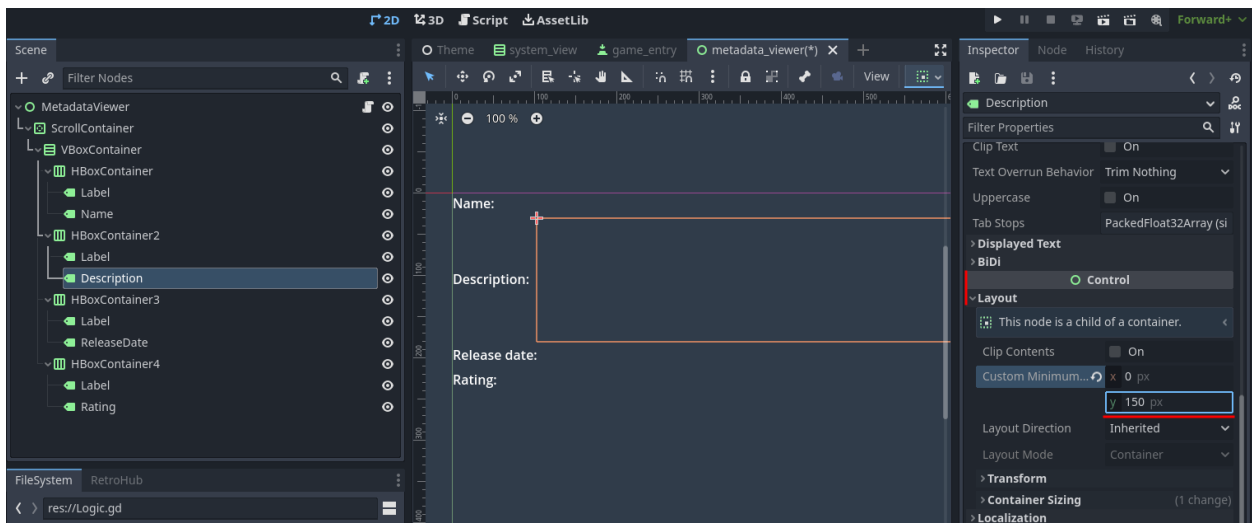
Duplicate the **HBoxContainer** 3 times, so you get 4 nodes in the end. Go to each **Label** and change the name accordingly, and rename **Name** to each field name.



Now let's go through each field which needs some tweaking.

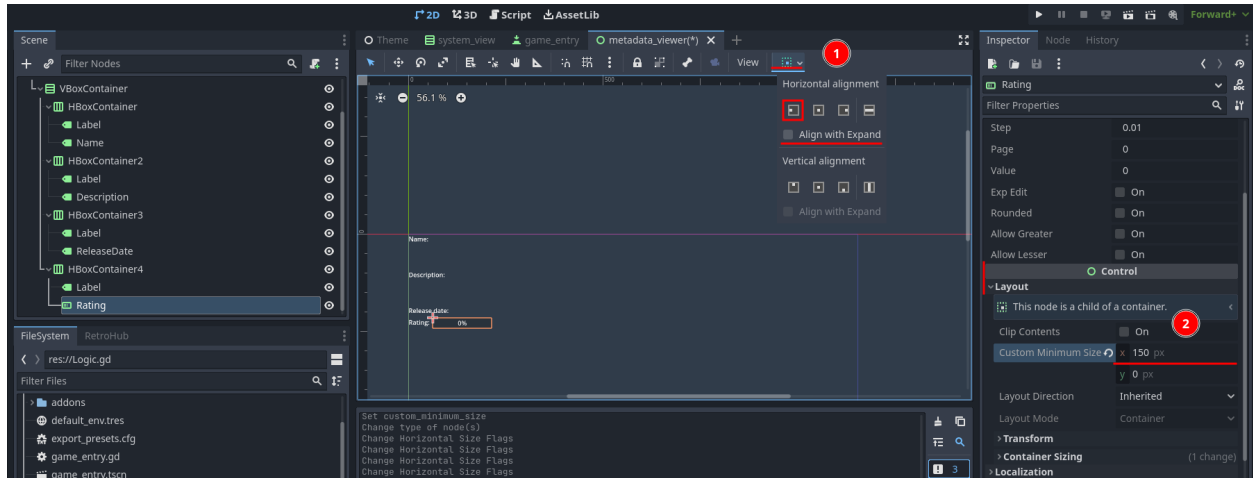
Description

Descriptions are very likely to contain a lot of text. We can increase its vertical size a bit to reflect that. Set the minimum vertical size to **150** pixels.



Rating

Rating will be a value between 0 and 1. So it makes more sense to show this information differently. Change the **Rating** node type to be a **ProgressBar** instead. Because it has the size flag set to expand, it occupies the full width, which is unnecessary and looks bad. Reset it's size flags to **Shrink Begin** and set a minimum horizontal size of **150** pixels.



Now, we can add the new nodes to our **metadata_viewer.gd** script to show all the information:

```
extends Control

# Nodes to edit
@onready var name_label := $ScrollContainer/VBoxContainer/HBoxContainer/Name
@onready var description_label := $ScrollContainer/VBoxContainer/HBoxContainer2/
↳ Description
@onready var release_date_label := $ScrollContainer/VBoxContainer/HBoxContainer3/
↳ ReleaseDate
@onready var rating_progress_bar := $ScrollContainer/VBoxContainer/HBoxContainer4/Rating

# Game data to display
var game_data : RetroHubGameData:
    set(value):
        game_data = value

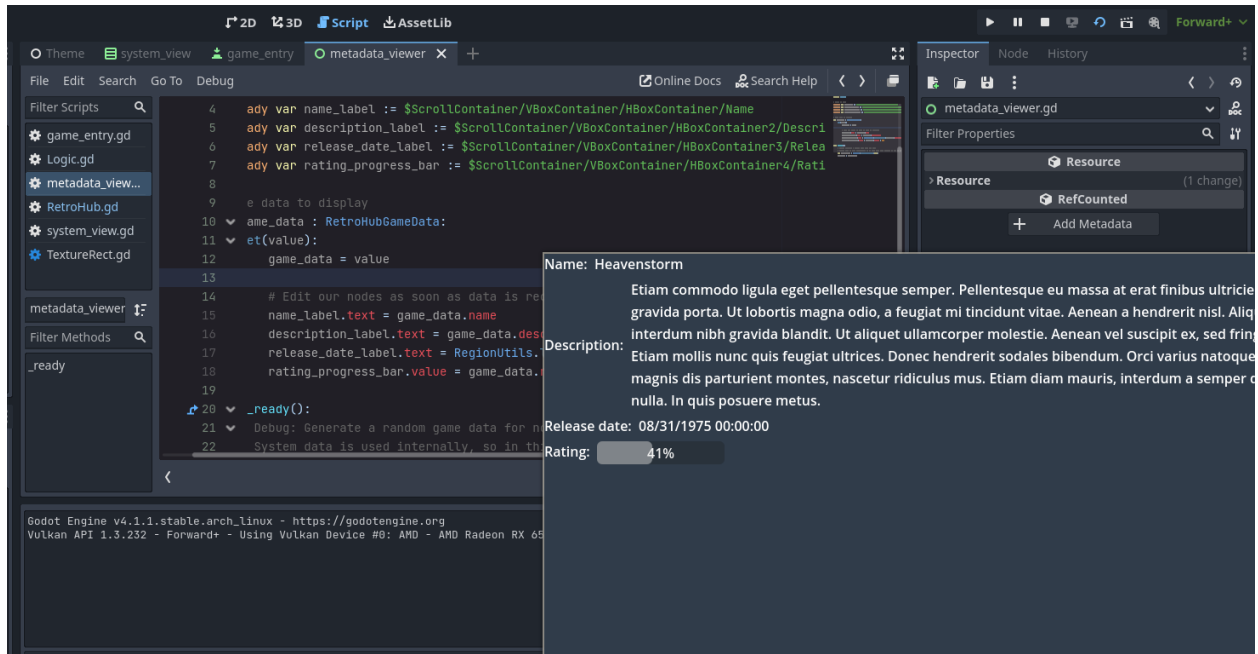
        # Edit our nodes as soon as data is received
        name_label.text = game_data.name
        description_label.text = game_data.description
        release_date_label.text = RegionUtils.localize_date(game_data.release_
↳ date)
        rating_progress_bar.value = game_data.rating * 100

func _ready():
    # Debug: Generate a random game data for now.
    # System data is used internally, so in this case it's alright to set to null
    var random_data = RetroHub.gen_random_game(null)
    game_data = random_data
```

Note: The release date text is in [ISO8601](#) format (e.g. 20190514T145802). To properly display this information, we used methods from [RegionUtils](#). You should rely on this in your theme to take into account the user's region and

always format it properly.

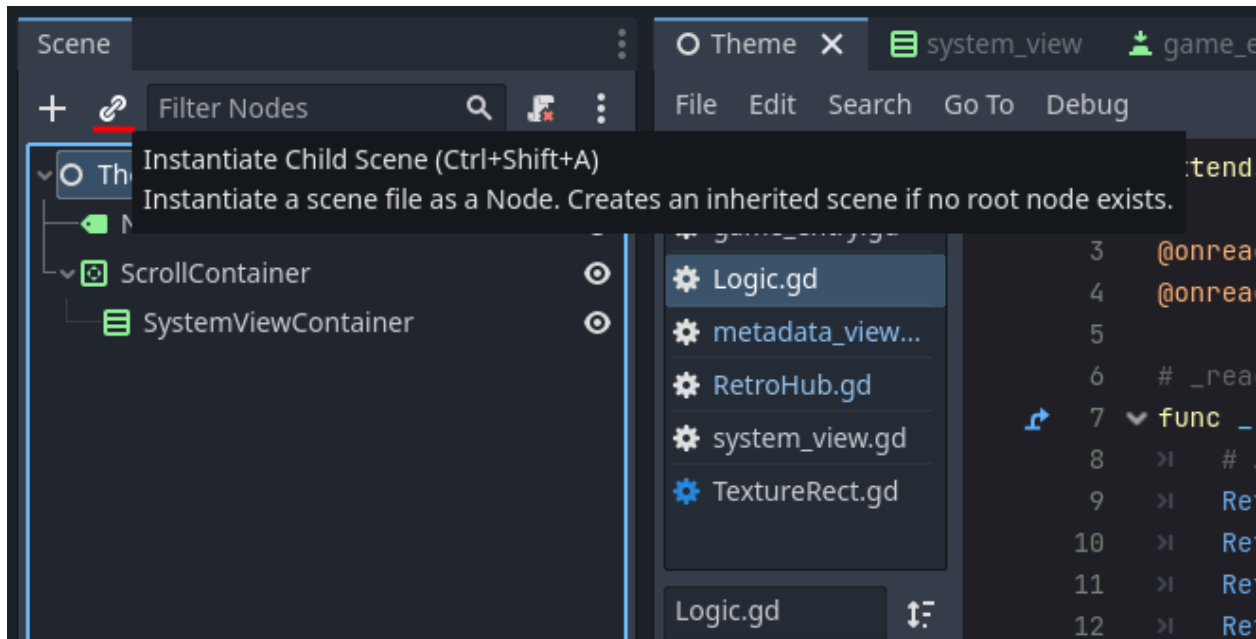
Run the scene and you'll see all the new information displaying properly. Try resizing the window and check that the content resizes properly.



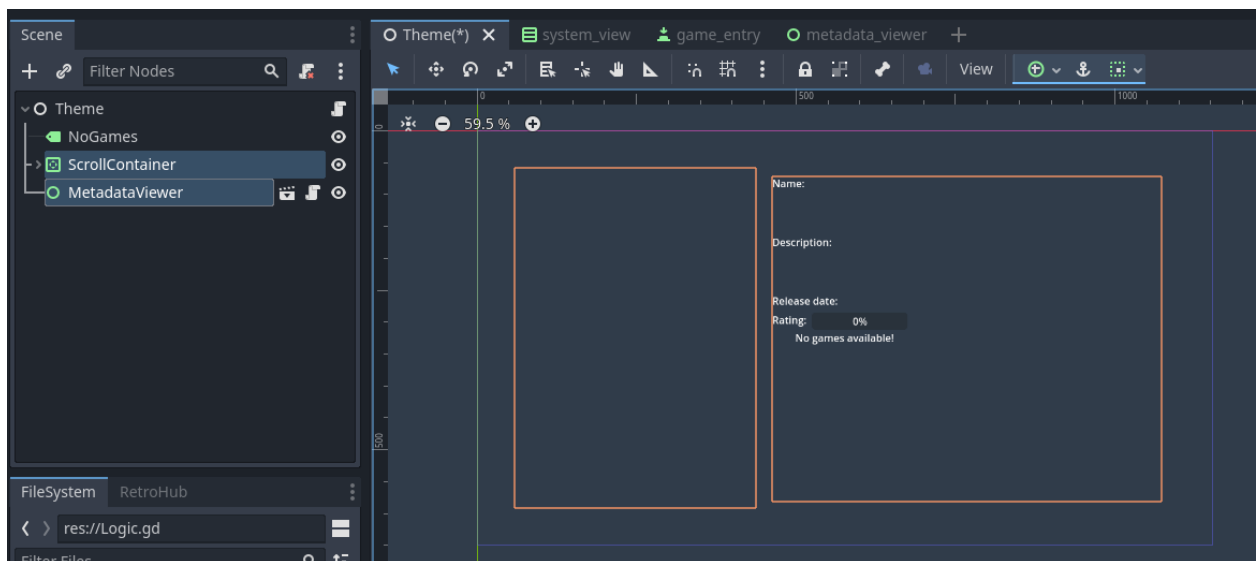
Connecting everything up

Now that we can successfully show game metadata, we need to add it to our main scene, and give it data to show. This last step will be a bit tricky, because the objects with all the game data are our game entries. They can't access the metadata viewer directly, so we'll have to change some things in the code to support this.

Firstly, let's add it to our main scene, which is the easy part. Open the main **Theme.tscn**, and instance our **metadata_viewer.tscn** scene through **Instance Child Scene**.



Move and resize the object so it only occupies the right portion of the screen, being careful not to overlap with the system view.



Now it's time to connect this in code to receive metadata to show. Recall that **SystemView** instances **GameEntry** children, and each entry is assigned it's own game data. Each button has a pressed signal whenever it's pressed, which we need to use anyways. One solution is to use this signal to transport each button's game data (in Godot, you can bind extra variables to signals). Edit **system_view.gd**:

```
...
func _on_game_received(game_data: RetroHubGameData):
    if game_data.system == system_data:
        var game_entry = preload("res://game_entry.tscn").instantiate()
        game_entry.game_data = game_data
        game_entry_container.add_child(game_entry)
        game_entry.pressed.connect(_on_game_entry_pressed.bind(game_data))
```

(continues on next page)

(continued from previous page)

```
func _on_game_entry_pressed(game_data: RetroHubGameData):
    # TODO
    pass
```

We now have a way to check when a button is pressed, and know what game data it has. However, we still can't access the **MetadataViewer** object from the **SystemView** code, so let's propagate this signal further. Create a new `game_selected` signal, and emit it when any game entry button is pressed:

```
extends VBoxContainer

signal game_selected(game_data: RetroHubGameData)

@onready var system_name_label := $SystemName
@onready var game_entry_container := $GameEntryContainer

...

func _on_game_entry_pressed(game_data: RetroHubGameData):
    game_selected.emit(game_data)
```

Alright, that takes care of the problem! **SystemView** now has a signal which will be triggered whenever an entry is pressed, and it exposes that entry's game data. We can now connect to this signal from our root **Theme** scene, although we need to do it in code as we're creating **SystemView** instances in runtime. Edit **Logic.gd**:

```
...
func _on_system_received(data: RetroHubSystemData):
    var system_view = preload("res://system_view.tscn").instantiate()
    system_view.system_data = data
    system_view.game_selected.connect(_on_game_selected)
    system_view_container.add_child(system_view)

## Called when a game entry is selected by the user
func _on_game_selected(game_data: RetroHubGameData):
    pass
```

All that's left now is get a reference to the **MetadataViewer** child, and then pass the game data onto it:

```
extends Node

@onready var no_games_label := $NoGames
@onready var system_view_container := $ScrollContainer/SystemViewContainer
@onready var metadata_viewer := $MetadataViewer

# _ready function, called everytime the theme is loaded, and only once
func _ready():
    # App related signals
    RetroHub.app_initializing.connect(_on_app_initializing)
    ...

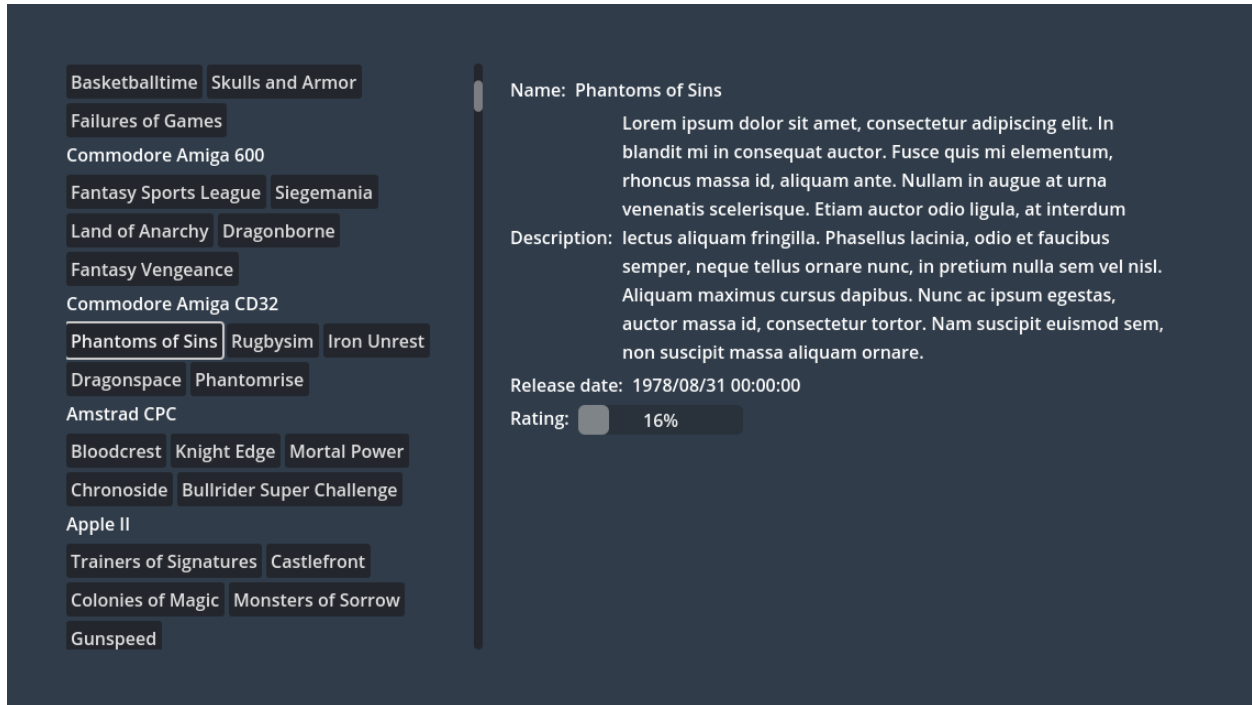
...
```

(continues on next page)

(continued from previous page)

```
## Called when a game entry is selected by the user
func _on_game_selected(game_data: RetroHubGameData):
    metadata_viewer.game_data = game_data
```

And it's done! Run the project now, and click on the game entries. The metadata will change accordingly to show that entry's information!



Note: Don't forget to remove the `_ready` function from `metadata_viewer.gd` which generated random games for testing!

Launching games

Now that we can properly display game information, we're ready to let RetroHub know the user has selected a given game, and wants to launch it.

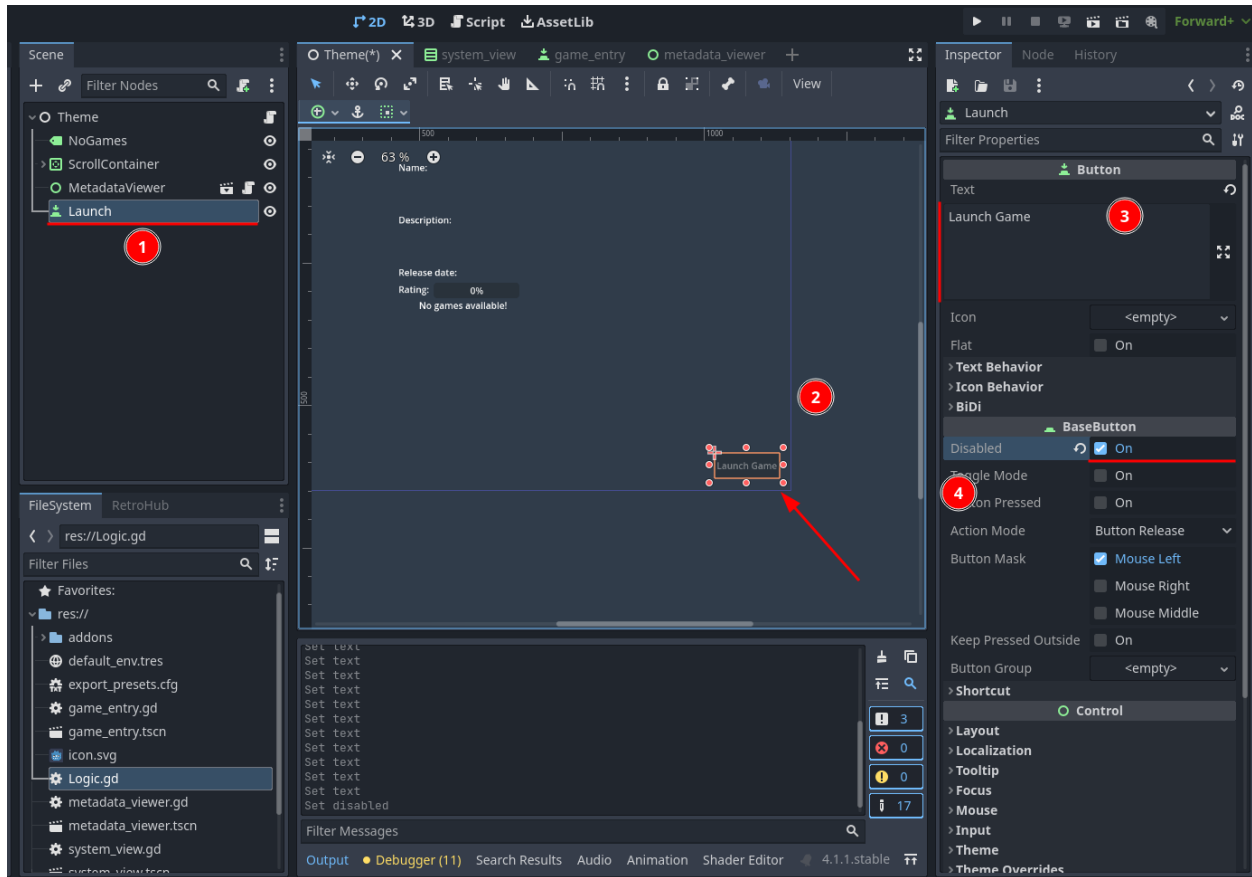
Generally, you should allow users to select a game before launching it, which allows them to edit it's metadata or scrape information from the main app.

We already have a system in place to know which game entry was selected, so we'll use it to signal RetroHub. Edit `Logic.gd`:

```
...
## Called when a game entry is selected by the user
func _on_game_selected(game_data: RetroHubGameData):
    metadata_viewer.game_data = game_data
    RetroHub.set_curr_game_data(game_data)
```

Now, we need to launch the game. If we used the current game entry buttons to do that though, the user would never have the chance of viewing the game information before launching it. So, let's create a new **Button** just for that.

Move it to the right-bottom corner and give it an appropriate label, such as “Launch Game”. We need to have it be disabled by default: when the app launches, no game data is selected, so if the button was active, the user could try to launch a null game data, which would cause problems.



Time to code it's behavior. Let's start by re-enabling it when a game entry is selected, on **Logic.gd**:

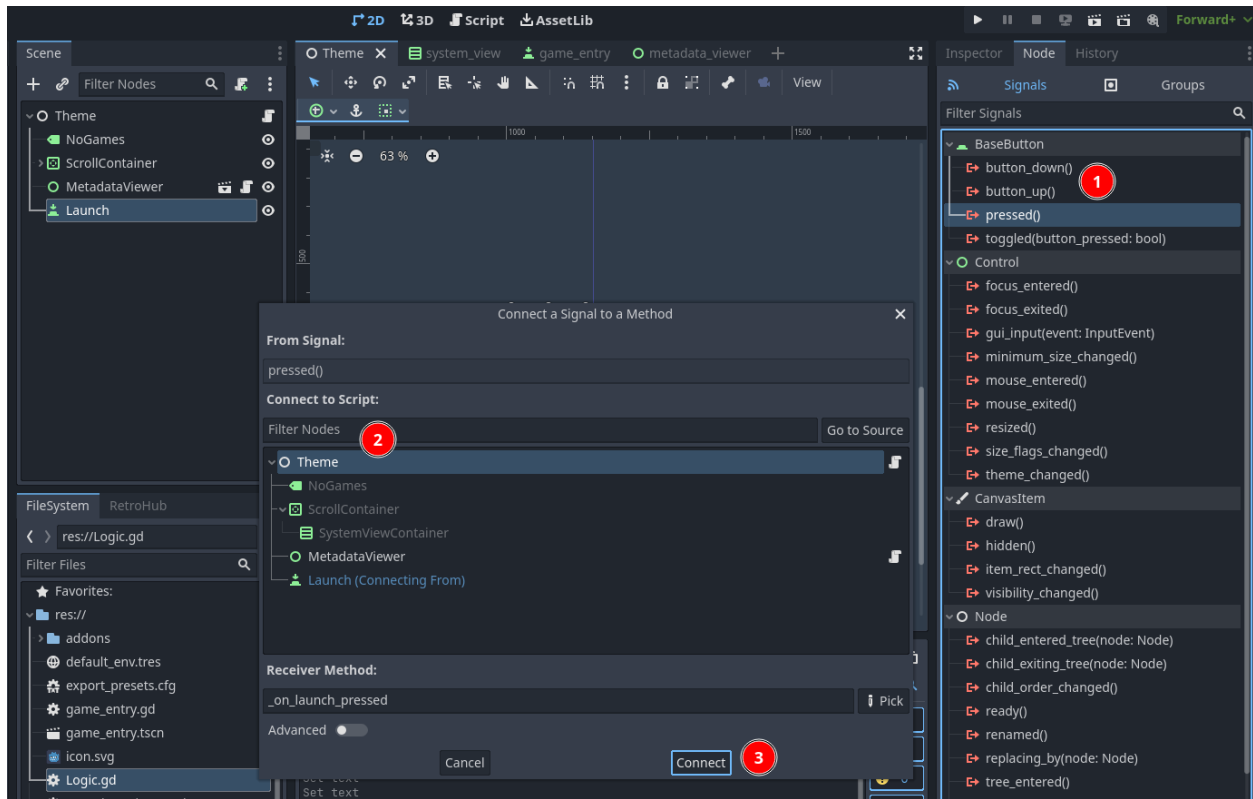
```
extends Node

@onready var no_games_label := $NoGames
@onready var system_view_container := $ScrollContainer/SystemViewContainer
@onready var metadata_viewer := $MetadataViewer
@onready var launch_button := $Launch

...

## Called when a game entry is selected by the user
func _on_game_selected(game_data: RetroHubGameData):
    metadata_viewer.game_data = game_data
    RetroHub.set_curr_game_data(game_data)
    launch_button.disabled = false
```

Since this button is part to the scene, we can connect to the pressed signal from the editor. Ensure it will add the method to the root **Theme** node (thus, on **Logic.gd**).



Now it's just a matter of asking RetroHub to launch the currently selected game data:

```
## Called when the launch button is pressed
func _on_launch_pressed():
    RetroHub.launch_game()
```

You can run the project now, and try launching a game. You'll see that no game will actually launch (this is random data anyways; sorry, I know you wanted to find out what "Roads and Spaceflight" was about), but if you look at the console output, you'll see messages indicating that a game is launched. This means everything is working; if this theme was running under RetroHub right now, it would launch the game!

Before exporting our theme to the final app, however, let's spice things up with some game media in the next section.

Showing game media

We have a functioning theme so far, but it's still too dull, as we're only showing text information.

Games might have some media to show, such as a title screen, game logo, or a short gameplay video. There's even more media for more complex usages, such as box and physical support textures!

Let's change our **MetadataViewer** to show a game's screenshot along the current text info.

Receiving game media data

Contrary to system and game data, media data has to be requested from RetroHub. While system and game data's are "cheap" to instance and have in memory, loading all the existing media would quickly eat all the user's memory. In our case it's also wasteful, since we only want to show media for the currently selected game.

Let's implement this behavior inside **MetadataViewer**. Open **metadata_viewer.gd** and request media, if available:

```
# Media data to display
var media_data : RetroHubGameMediaData

# Game data to display
var game_data : RetroHubGameData:
    set(value):
        game_data = value

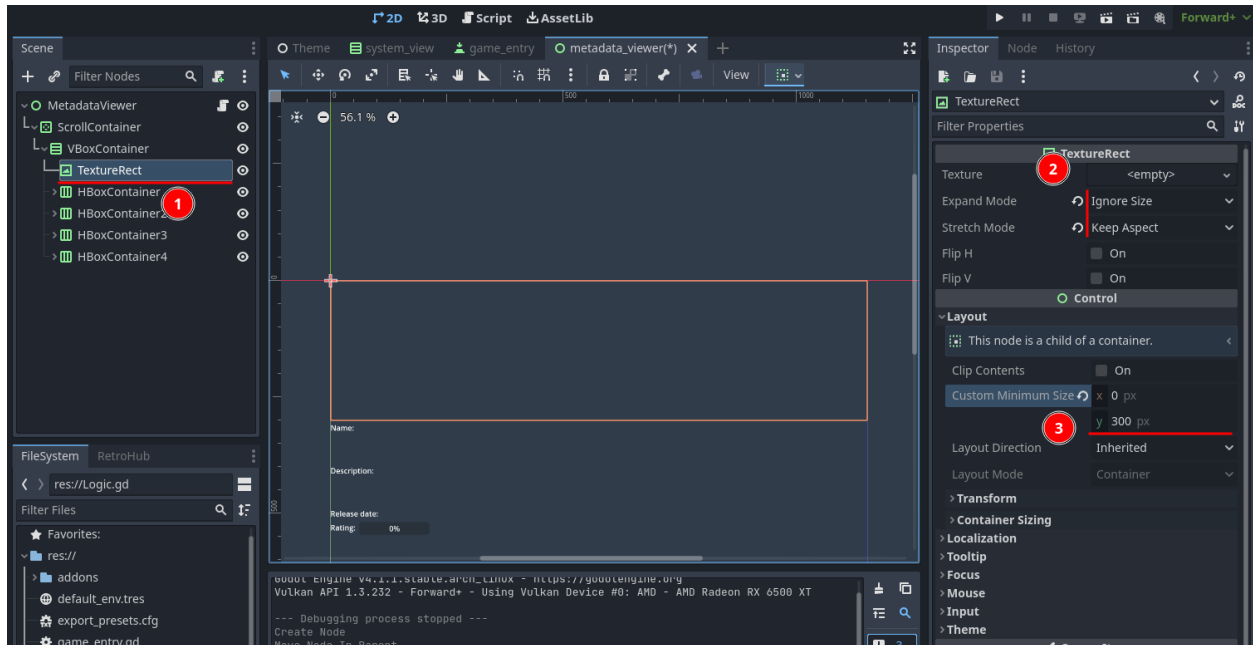
        # Edit our nodes as soon as data is received
        name_label.text = game_data.name
        description_label.text = game_data.description
        release_date_label.text = RegionUtils.localize_date(game_data.release_
↵date)

        rating_progress_bar.value = game_data.rating * 100

        # If media is available, show it as well
        if game_data.has_media:
            media_data = RetroHubMedia.retrieve_media_data(game_data)
```

Note: The reason why `media_data` is global, instead of a local variable inside each game entry, is to ensure the old media data will be deleted when it changes to something else. Media data is expensive to have in memory, so you have to take extra care to ensure it's properly deleted when it's not needed anymore.

To show a screenshot, the media data provides an **ImageTexture** resource. Let's add a node suitable for it. Right before the first name property (**HBoxContainer**), add a **TextureRect** node. Images may come in all sizes, so set the **Expand Mode** to **Ignore Size** to force the image to be scaled (preventing extremely large or small pictures). Additionally, images may have different aspect ratios (4:3 vs 16:9 games, for example), so set it's **Stretch Mode** to **Keep Aspect** to respect it. Lastly, because we now control the image size, set a minimum height of **300** pixels.



Rename the node to **Screenshot**, add it to **metadata_viewer.gd** and set it's texture value when the requested type of media exists:

```
# Nodes to edit
@onready var name_label := $ScrollContainer/VBoxContainer/HBoxContainer/Name
@onready var description_label := $ScrollContainer/VBoxContainer/HBoxContainer2/
↳Description
@onready var release_date_label := $ScrollContainer/VBoxContainer/HBoxContainer3/
↳ReleaseDate
@onready var rating_progress_bar := $ScrollContainer/VBoxContainer/HBoxContainer4/Rating
@onready var screenshot_texture_rect := $ScrollContainer/VBoxContainer/Screenshot
...

# Game data to display
var game_data : RetroHubGameData:
    set(value):
        ...

        # If media is available, show it as well
        if game_data.has_media:
            media_data = RetroHubMedia.retrieve_media_data(game_data)
            if media_data.screenshot:
                screenshot_texture_rect.texture = media_data.screenshot
```

Lastly, we need to take into account game entries that don't have media to show. If that's the case, the image will be empty but still taking space, which looks bad. We can hide it when there's no media available:

```
# Game data to display
var game_data : RetroHubGameData:
    set(value):
        ...

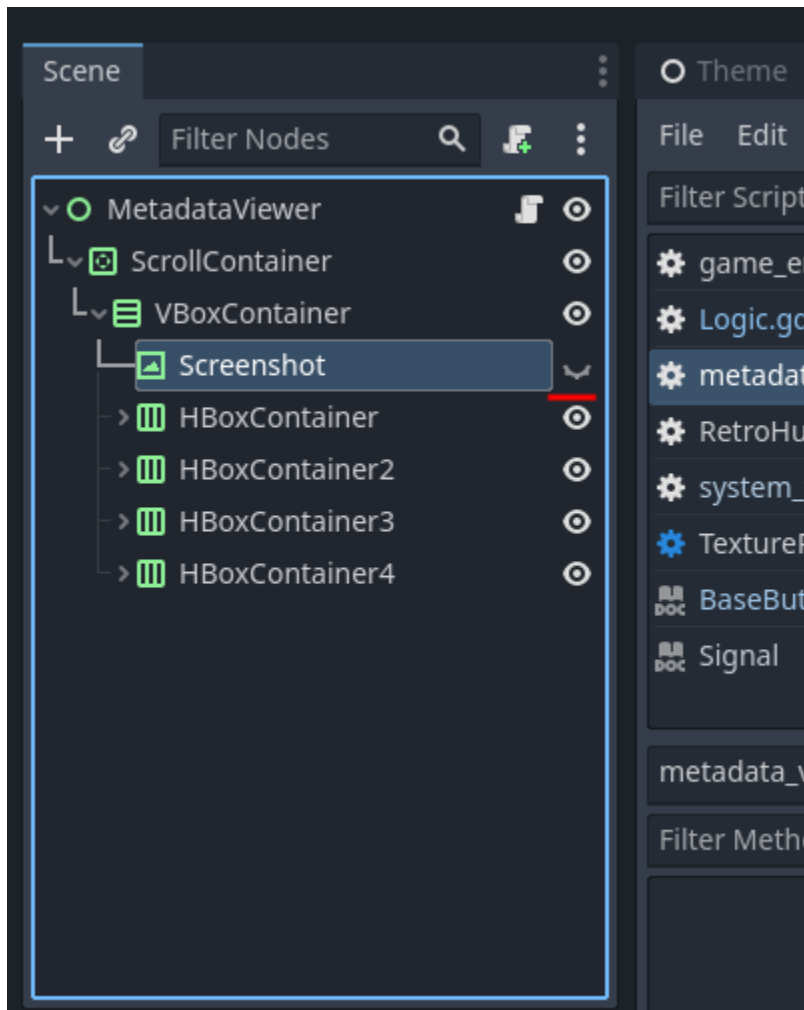
        # If media is available, show it as well
```

(continues on next page)

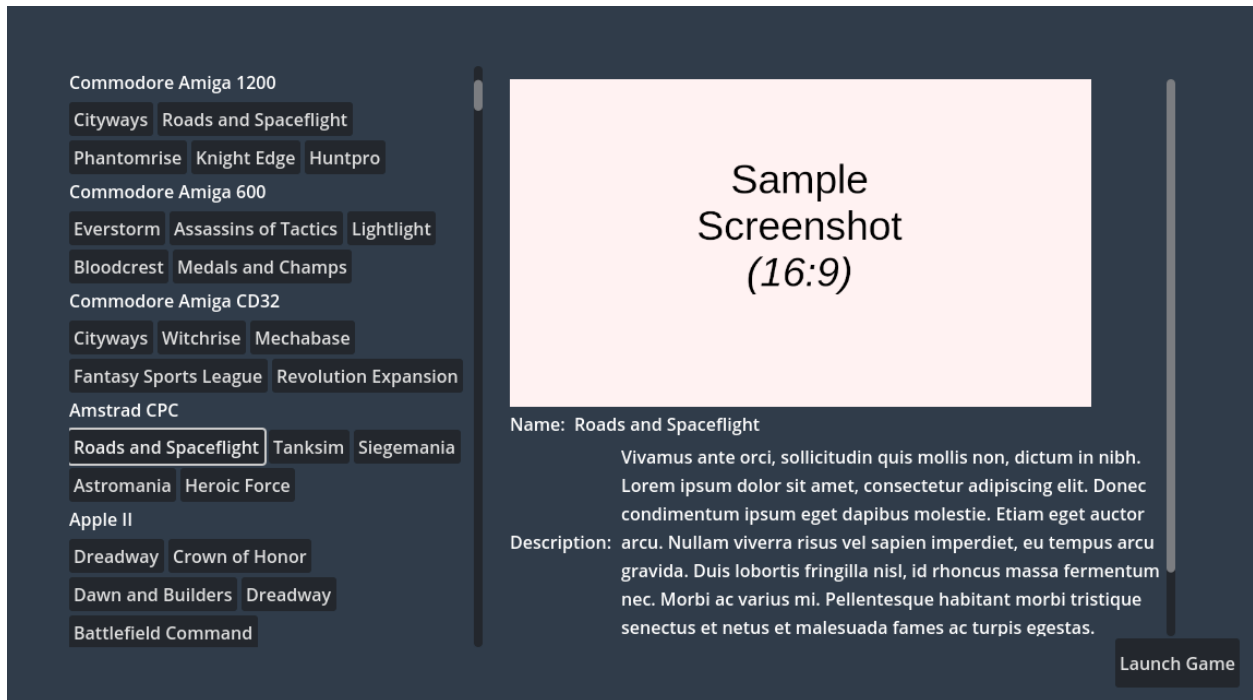
(continued from previous page)

```
if game_data.has_media:
    media_data = RetroHubMedia.retrieve_media_data(game_data)
    if media_data.screenshot:
        screenshot_texture_rect.visible = true
        screenshot_texture_rect.texture = media_data.screenshot
    else:
        screenshot_texture_rect.visible = false
else:
    screenshot_texture_rect.visible = false
```

And also hide the node by default:



Run the project now, and you'll see a sample screenshot when selecting game entries; some in 4:3, some in 16:9.



Supporting other types of media is very straightforward, as most of them are **ImageTexture** resources as well. More advanced cases such as video and box/support textures are explained in other sections if you want to explore them.

Our theme is ready now! Now, we just need to export it to be used in RetroHub, which we'll do in the next, and final, section!

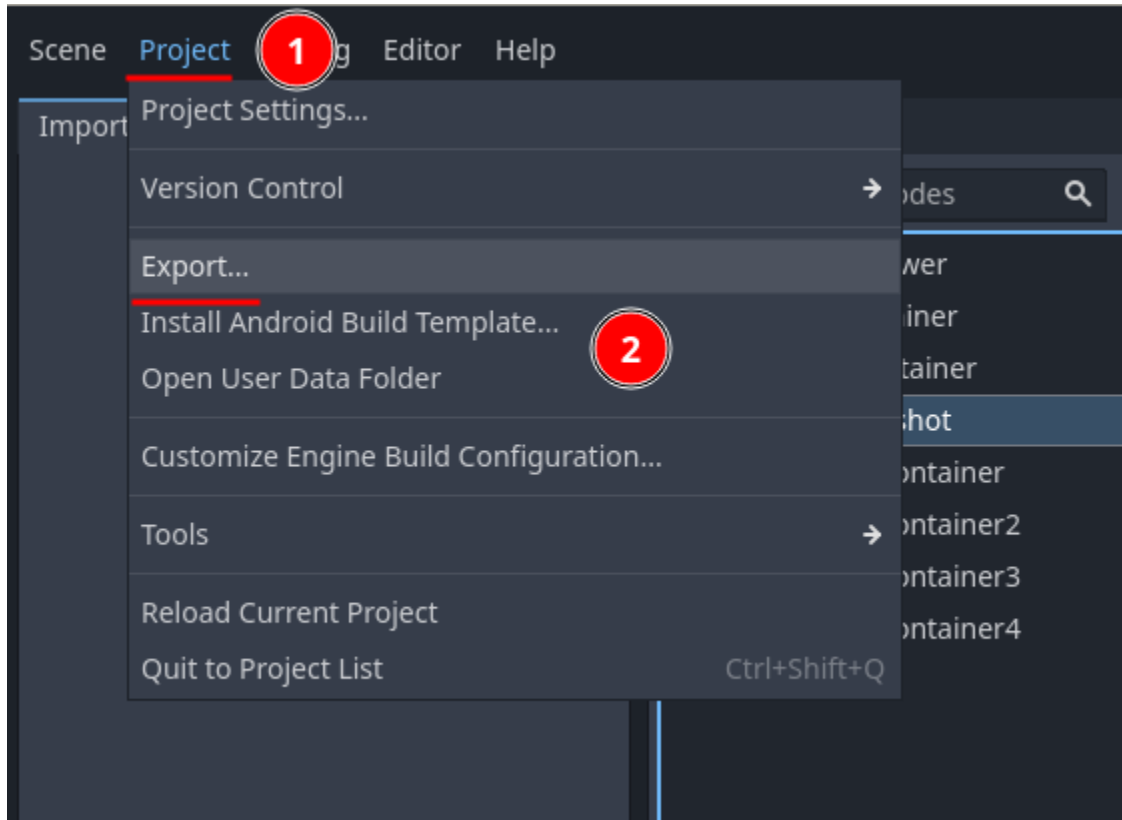
Exporting theme

With the theme fully ready, it's time to export it to be used in RetroHub. This is pretty simple and your theme should behave almost exactly the same way afterwards.

Exporting

To export a theme, you'll export it like it's a regular Godot project. The difference is you're not going to export a final executable; instead, you'll export it in a PCK format, an "archive" which Godot can load at runtime.

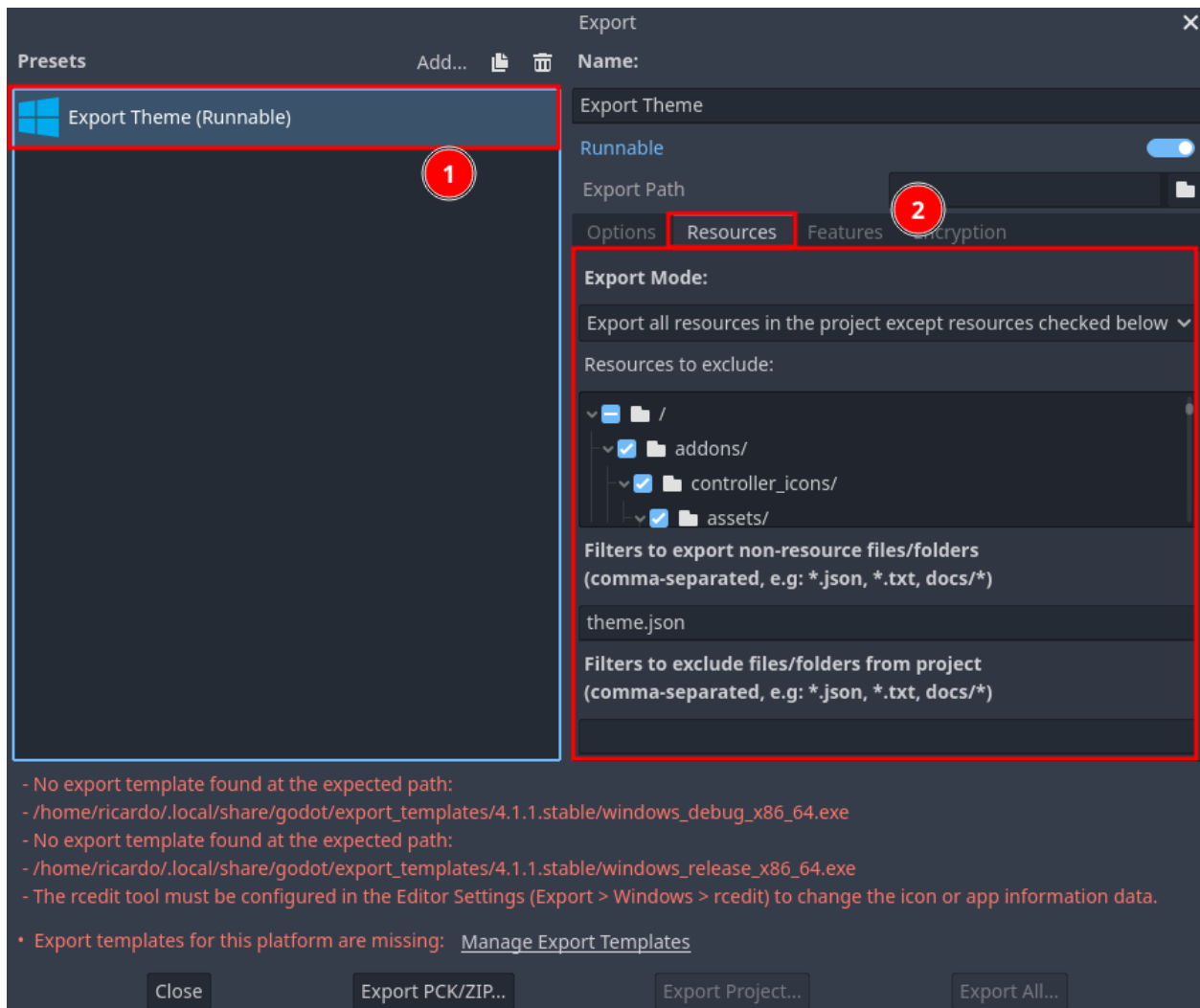
Open the export dialog by going to **Project > Export...**



Select the **Export Theme** preset already available; the selected OS is indifferent as the exported project is cross-platform and will run on any OS.

Open the **Resources** tab and ensure the settings are set properly:

- **Export Mode:** Export all resources in the project except resources checked below
- **Resources to exclude:** Tick all addons which are already bundled in RetroHub to save space, as well as the `retrohub_theme_helper` addon, which is only used for development purposes.
- **Filter to export non-resource files/folders:** Add `theme.json`. This is the file read by RetroHub with information about your theme (all the info from the **RetroHub** tab)



Finally, you can export your theme by clicking at **Export PCK/Zip...**

Note: You need to export in the PCK format, as it's the only one RetroHub currently supports.

Testing

To test your theme, drop the exported theme in RetroHub's theme folder, and *open it in the app*.

Conclusion

Congratulations on creating your first theme! To make this tutorial short and to the point, the final theme ended up being pretty barebones, but it was enough to demonstrate that all the implementation details are up to you as a theme developer, and that you'll have a lot of freedom on how to create themes.

For more in-depth guidance, check the remaining [help pages](#), and for any questions, suggestions, or just showcasing your work, join us at [Discord](#) and [Reddit](#).

2.2 General Information

2.2.1 Godot restrictions

Despite a theme being essentially a Godot project, not everything can be used as expected. The following features are restricted or outright unavailable to use:

- **Project Settings:** Any custom value that's saved as a project setting (anything set under the Project tab) is lost when exporting the project, as the settings file is neither exported nor used in RetroHub. Custom input actions, physics layer names, render/window settings and others will be lost when the theme is exported.
- **Singletons:** Since Singletons/AutoLoads are set in the project settings, this information is not saved on exporting. Furthermore, Godot loads singletons internally, so this functionality is not exposed in GDScript to modify them at runtime. To overcome this, you can preload the singleton script/scene and propagate it in scripts as necessary:

```
var MySingleton := preload("res://path/to/MySingleton.gd")

MySingleton.do_something()

foo(MySingleton)

def foo(MySingleton):
    MySingleton.do_something()
```
























- **Localization:** There's no support for localization files yet. RetroHub will offer some mechanism to support localization in themes in the future.
- **Root Viewport:** RetroHub runs themes in a dedicated viewport. This means that the root viewport is not the same as the one in the editor. Do not depend on any specific behavior of the root viewport, such as using it to detect the true screen size.
- **Viewport scaling & stretching:** RetroHub simulates the 2D stretch and `keep_height` aspect properties for [handling multiple resolutions](#). This is to ensure a UI designed for the base resolutions scales well on other resolutions, particularly for TVs. Right now, themes have no control over this behavior, but in the future it may be possible to override this.
- **Controller Input:** Controller input is finicky by default for UI elements. If you try to use the joystick for selecting UI elements, you'll find it goes all over the place and doesn't allow for proper selection of elements. This is an issue from Godot that's [yet to be solved at the time of writing](#). RetroHub has a global system to that "hijacks" controller input and solves this problem, so when your theme is exported, it should work normally. For development, avoid using the stick for navigation, and use the DPAD/keyboard instead. For more details on how to ensure controller input works in your theme, check the [Input](#) section.
- **Video Playback:** Playback of x264/x265 files (.mp4) files will not work outside of RetroHub's official builds. Godot cannot support these codecs because "H.264 and H.265 cannot be supported in core Godot, as they are both encumbered by software patents."

2.2.2 Input

Input is handled by RetroHub first, then passed onto the theme. This is to ensure input works properly if an app menu is open, as well as fix some issues regarding controller input.






Input actions

Your theme should ideally only use input actions to ensure full compatibility, which also letting users easily remap controls to their liking. The following input actions are available:

Action	Description	Default keys
rh_up	Move up	 / 
rh_down	Move down	 / 
rh_left	Move left	 / 
rh_right	Move right	 / 
rh_accept	Accept a given action	 / 
rh_back	Cancel a given action	 / 
rh_major_option	Major/frequent option	 / 
rh_minor_option	Minor/rare option	 / 
rh_theme_menu	Open the theme menu	 / 
rh_left_shoulder	Slide left	 / 
rh_right_shoulder	Slide right	 / 
rh_left_trigger	Trigger left	

continues on next page

Table 1 – continued from previous page

Action	Description	Default keys
rh_right_trigger	Trigger right	
rh_rstick_left	Move the right stick left	
rh_rstick_right	Move the right stick right	
rh_rstick_up	Move the right stick up	
rh_rstick_down	Move the right stick down	

Input Icons

RetroHub uses [Controller Icons](#) to show appropriate icons for each input method. This addon automatically switches icons between keyboard/mouse and controllers, so use it to display how to perform a given action in your theme. [Here's a quick guide to get started.](#)

Controllers and UI

Ideally your theme is fully usable with a controller, without requiring any keyboard/mouse input. If you use Godot's builtin UI nodes, you need to ensure that these nodes receive focus, and that their focus neighbors are set as well. Here are a few more tips:

- Godot will rarely set a node as focused on the first frame. Call `grab_focus` on `_ready()` to ensure there's always a focused node.
- Controls that require keyboard input, such as `LineEdit`, `TextEdit` and `SpinBox` are automatically handled by RetroHub. If you need the virtual keyboard for another type of node, you can manually call `RetroHubUI.show_virtual_keyboard()`. More information on the [UI section](#) and in the [RetroHubUI](#) class reference.

Note: To design a theme for controllers without having one available, ensure you can use your theme with only the keyboard, and that you never have to use the mouse to access some part of your theme.

2.2.3 User Interface

This section shows some existing user interfaces from RetroHub that themes can use. All of these are accessible through the `RetroHubUI` singleton.

Default Colors

There are some default colors you can use for some scenarios:

- Success - `RetroHubUI.color_success`
- Warning - `RetroHubUI.color_warning`
- Error - `RetroHubUI.color_error`
- Pending - `RetroHubUI.color_pending`
- Unavailable - `RetroHubUI.color_unavailable`

File/Directory Picker

If you need to pick a file or directory to load, you can request that through the existing interface for that. This also ensures proper controller support in this scenario.

```
# Example to load a single PNG file
RetroHubUI.filesystem_filters(["*.png ; PNG Images"])
RetroHubUI.request_file_load(FileUtils.get_home_dir())
var path : String = yield(RetroHubUI, "path_selected")
    if not path.empty():
        print("File selected: " + path)
    else:
        print("No file selected")
```

For more information check the [RetroHubUI](#) class reference.

App Icons

You can access the default icons that are shipped with RetroHub. All the icons available are in the `RetroHubUI.Icons` enum.

```
# Example to load the default icon for loading files
var icon : Texture = RetroHubUI.load_app_icon(RetroHubUI.Icons.LOAD)
```

Virtual Keyboard

RetroHub will show a virtual keyboard automatically if a `LineEdit` or `TextEdit` node is focused from a controller and/or mouse input. If you need to, you can also manually trigger the virtual keyboard to show up with `RetroHubUI.show_virtual_keyboard()`.

You need to have the required node be currently focused and accept raw key events for this to properly work. For more information check the [RetroHubUI](#) class reference.

Interface Sounds

RetroHub ships with some interface sounds which can also be used by themes with very minor modifications. This system works by detecting focused [Control](#) nodes and playing appropriate sounds.

To prevent this system from playing sounds automatically to a particular node, add it to the `rh_no_sound` group.

To manually play sounds yourself, use the *[play_sound](#)* function.

2.2.4 Theme Configuration

Since RetroHub gives near complete freedom to theme developers on how games are presented and selected, it's very likely that you'll have behaviors that users will want to customize.

Use *[RetroHubConfig](#)*'s `theme_config` methods and signals to access and modify these settings:

```
var dark_mode := false

func _ready():
    RetroHubConfig.theme_config_ready.connect(_on_theme_config_ready)
    RetroHubConfig.theme_config_updated.connect(_on_theme_config_updated)

func _on_theme_config_ready():
    dark_mode = RetroHubConfig.get_theme_config("dark_mode", false)

func _on_theme_config_updated(key, old_value, new_value):
    if key == "dark_mode":
        dark_mode = new_value
        print("Dark mode changed from %s to %s" % [old_value, new_value])
```

Theme configurations are stored internally in a JSON file, and work as key/value pairs. Theoretically you can use any Godot's [Variant](#), but it's recommended to stick to regular variable types, such as `bool`, `int`, `float`, `String`, `Vector2`, `Color`, etc...



When reading settings, you need to specify a default value in case that setting doesn't exist yet. To update settings, you only need to call `set_theme_config`, and RetroHub will internally save it.

Showing configuration

You have two main options for this:

Note: These options are not mutually exclusive. You can use both if you want.

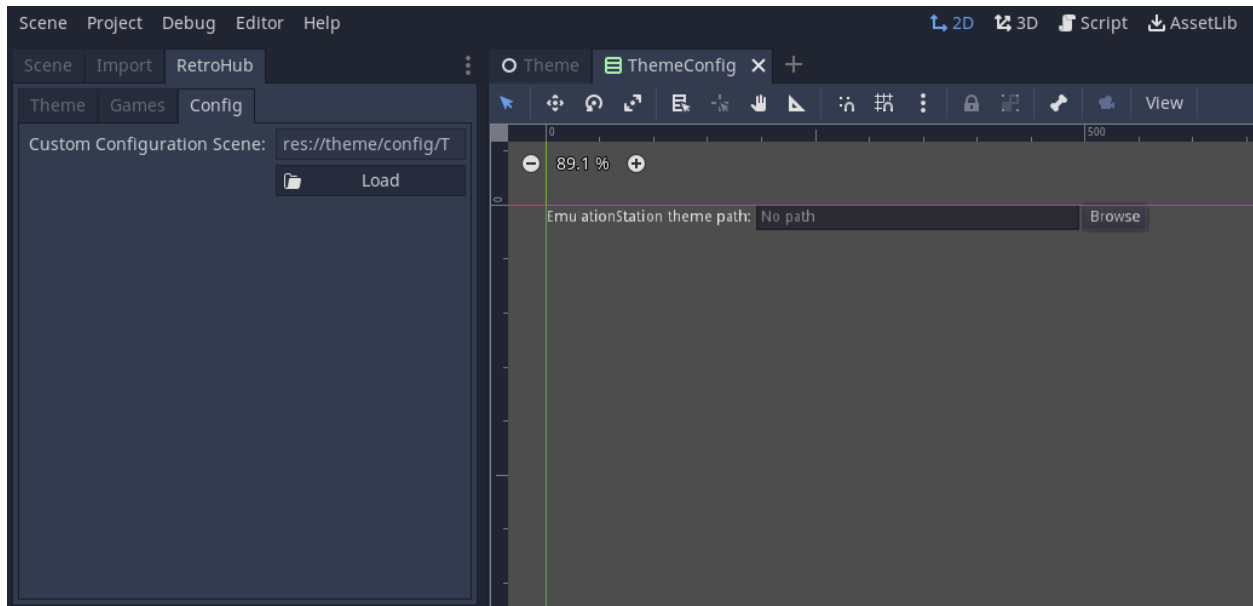
In-theme menu

Themes have a special input action available: **rh_theme_menu** ( /  by default). You can use this to implement a configuration UI from within your theme.

Warning: You will have to manually save this configuration by calling `save_theme_config`.

In-app menu

If you want to integrate the configuration UI better in the app, you can also specify a scene for RetroHub to load and add to the configuration menu. This will be displayed under the “Theme” section. To do this, set a Custom Configuration Scene in the theme helper addon’s configuration.



Note: When using this approach, RetroHub will automatically save theme configuration when the theme is unloaded, so you don’t need to do it manually.

2.3 Specific Information

2.3.1 Handling multiple aspect ratios

Screens have different sizes, and consequently different aspect ratios. While 16:9 has become the standard for most modern screens, there are plenty of other screens out there:

- Small variations, such as 16:10 (e.g. Steam Deck)
- Ultra-wide formats, such as 21:9
- Old “square” TVs, such as 4:3

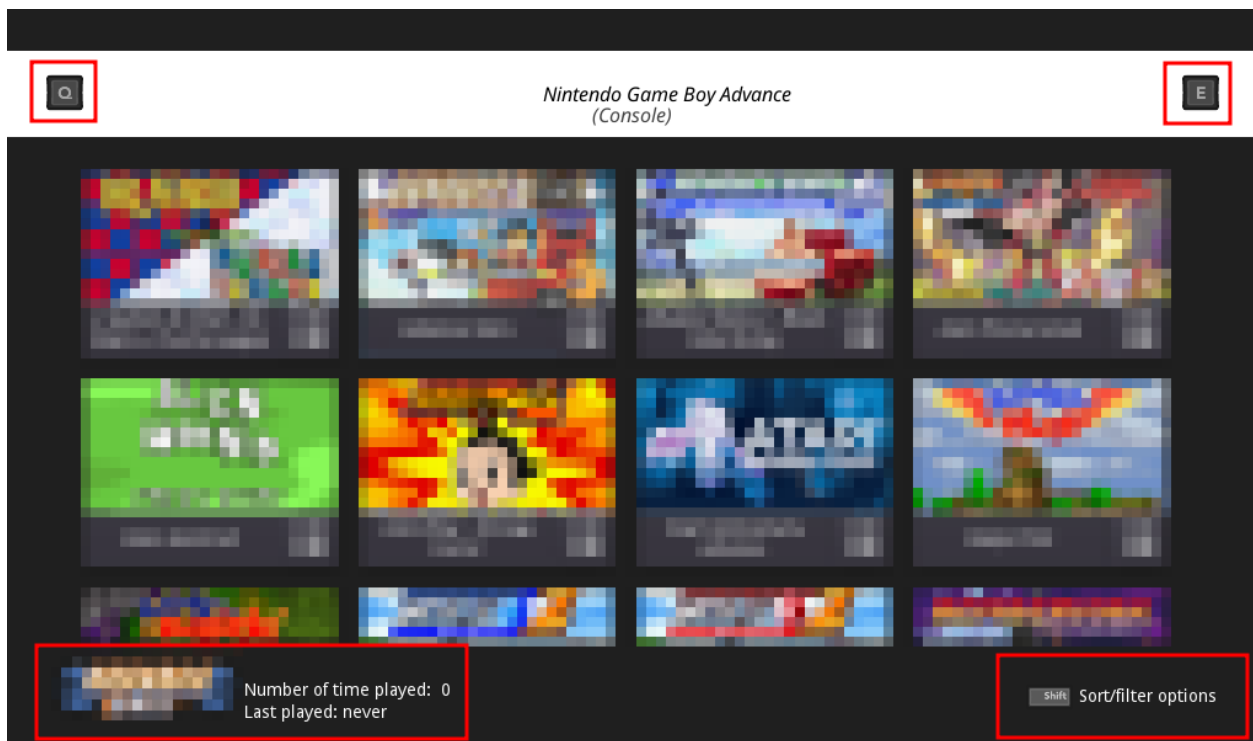
RetroHub is likely to run on such scenarios, but it's infeasible to develop for every specific aspect ratio, and no doubt even more will come in the future. Instead, themes should be designed according to Godot's guidelines, allowing themes to dynamically change depending on the user's resolution.

Even when designing for 16:9, you should keep these tips in mind when designing your theme, and hopefully supporting more aspect ratios will be pretty easy. RetroHub automatically scales the theme's resolution to keep the same height in all cases. Therefore, only the width will change.

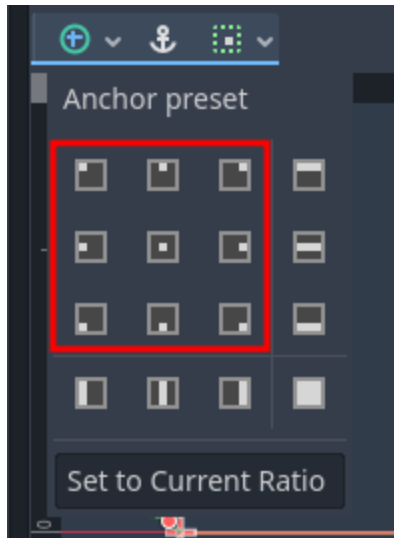
Note: For more information on this topic, checkout [this Godot tutorial](#).

Identify corner elements

Corner elements are portions of your UI that should “stick” to a corner, and not grow in size if the aspect ratio is different.

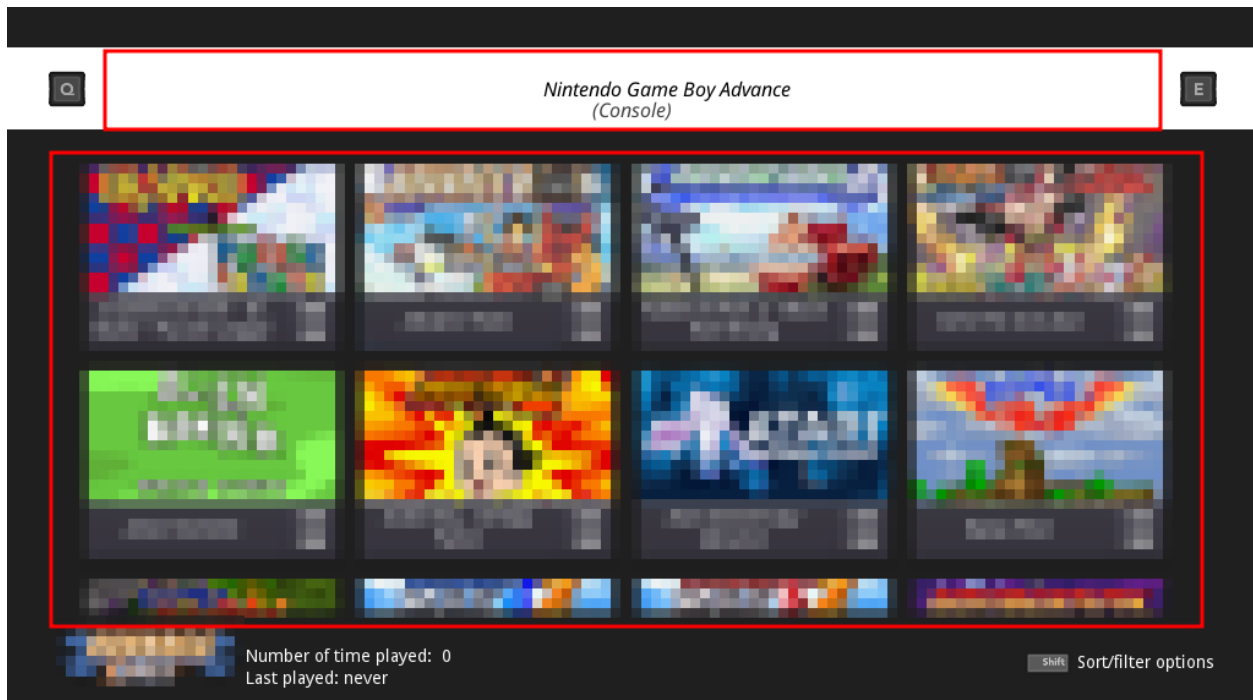


To have them automatically “stick” to corners, you can set the Layout for a specific case.

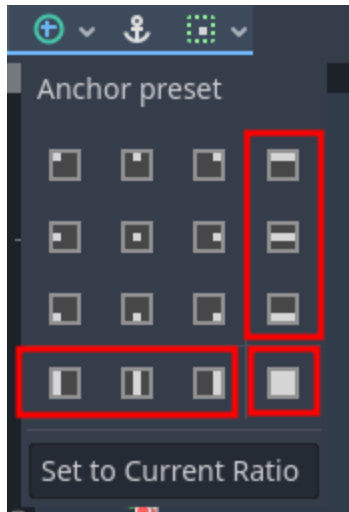


Identify filler elements

Filler elements are any portions of your UI you want to take all the available space.



To have them automatically take all the available space, you can set filler layouts for a specific case.



If those nodes have children, you'll need to set their size flags accordingly to use all the space available, such as **Fill** and **Expand**.

If you have a variable amount of elements (e.g. game entries), you can use [FlowContainer](#) nodes to automatically fill and overflow elements.

2.3.2 Accessibility - Supporting screen readers

A [screen reader](#) is a dedicated software that reads aloud content from the user's screen. It is used by people with partial or full visual impairments.

RetroHub supports screen readers in its interface by bundling [godot-accessibility](#) for out-of-the-box support for Godot's UI nodes.

This addon was heavily modified for RetroHub, and this page describes its functionality and how your theme can be adapted for screen reader usage.

Fundamentals

If you've never developed for visually impaired users, these are the most important design principles to abide by:

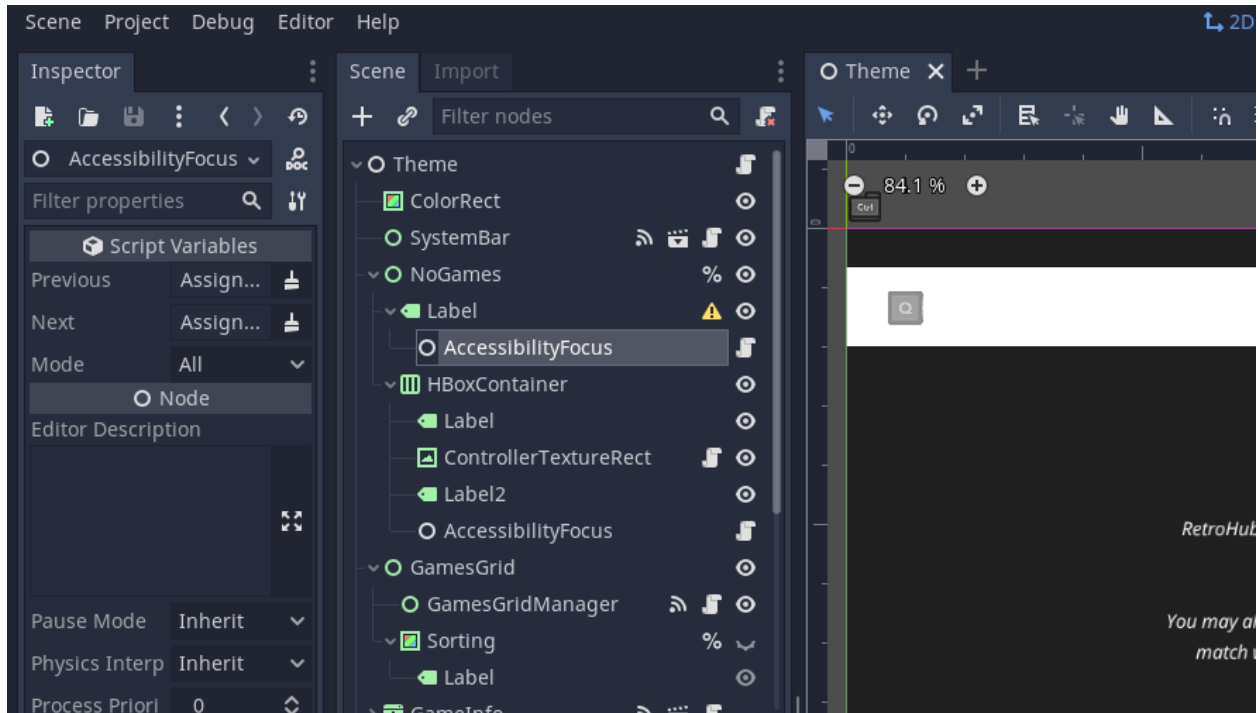
- **Mouse input must not be necessary:** Most visually impaired users do not use a mouse, and instead use keyboards/controllers to fully navigate the UI.
- **UI navigation is 1D, instead of 2D:** The interface must be fully navigable by keyboard/controller and only with two keys, typically the Up and Down arrow keys on keyboard.
- **Avoid audio interference:** Be careful with background sounds playing. While some software can automatically lower the application's sound when TTS is active, you should take extra care to ensure any TTS speech is not being interfered with by other elements of your theme.

While these additional restrictions are likely to affect your theme development significantly, the way these addons interact, as well as modifications done to them, should make the process of adapting your theme very straightforward, and as a complement to the existing design.

Making necessary nodes focusable

The addon works by detecting changes to the focused `Control`, and inspecting its content to read it aloud. However, Godot only makes interactible nodes focusable by default, leaving some essential nodes out (most notably `Label`).

For the screen reader to work, any `Control` node that should be transmitted to the user must be focusable. However, this would affect your existing designs for non-impaired users. To fix this, there's a special node to handle this: `AccessibilityFocus`. This node detects when a screen reader is enabled, and modifies its parent node's focus properties accordingly, while keeping the original configuration in case the user disables screen readers.



Likewise, if you use `grab_focus` to set an initial focus when your theme is loaded, you should adapt your theme to focus the appropriate node depending on the screen reader being enabled or not. You can check this setting through the `accessibility_screen_reader_enabled` property:

```
func _ready():
    if RetroHubConfig.config.accessibility_screen_reader_enabled:
        $LabelNode.grab_focus()
    else:
        $ButtonNode.grab_focus()
```

Ensuring one-dimensional navigation

To support one-dimensional movement and selection on the UI, RetroHub changes the `ui_up/ui_down` mappings to `ui_focus_next/ui_focus_prev` respectively. Godot's automatic mechanism to detect next and previous elements takes into consideration both axis, and thus it can focus elements more reliably.

If you need more control on what neighbor nodes get focused, add a `AccessibilityFocus` node to the parent node, and set the `previous` and `next` properties to the appropriate nodes. Likewise, if there's an element that doesn't make sense to be focusable for 1D UI, you can set the `focus_mode` property accordingly.

It is also extremely important to make the UI navigation deterministic. Test your theme to ensure that when you cycle through the UI backwards, the same elements are being focused in the reverse order. You can still have the concept of

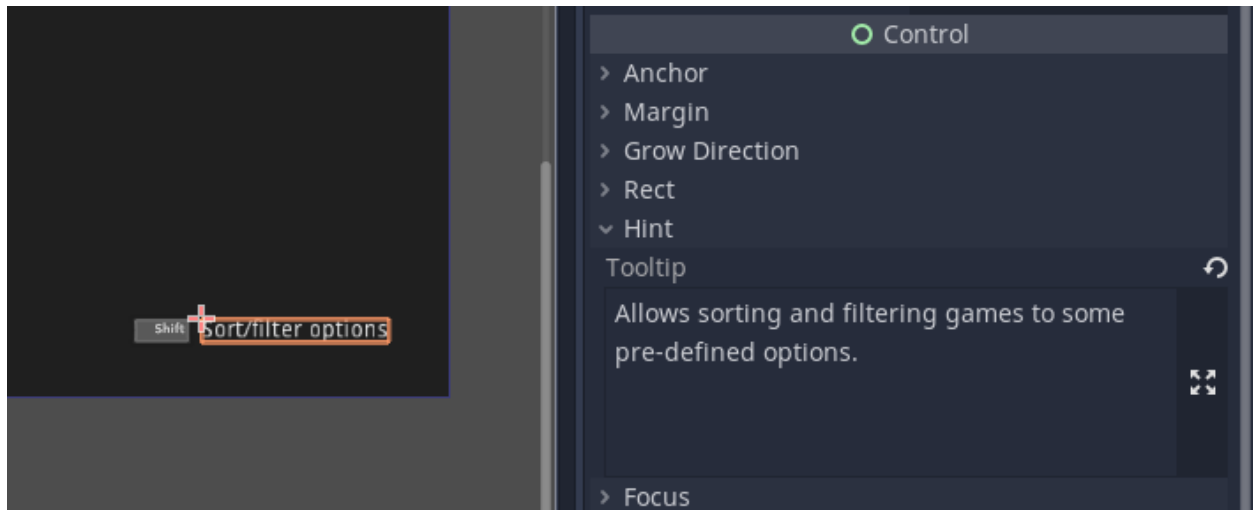
entering (`rh_accept`) and exiting (`rh_back`) from certain elements for more specialized navigation.

The remaining UI interactions, such as sliding (`rh_left_trigger/rh_right_trigger`), do not need any special handling, other than making the user aware that these actions are available.

Controlling TTS output

By default, this addon will output text according to the node's type. However, you can add more information or even control the TTS output you give to the user.

To better explain a given object's functionality, you can set the tooltip text for any node. The addon will append this text after the regular speech.



If you want to control the TTS output yourself, you can do that through a `tts_text` method. The addon will visit this node and parents recursively to check if this method exists, and if yes, use that as the output instead. This way, you can define your own text for any node, and have the flexibility of either defining this method in children node's scripts, or consolidating everything to a parent node.

```
func tts_text(focused: Control) -> String:
    if focused == special_node:
        return "This is a special node!"
    else:
        # Empty strings make the addon keep visiting further nodes or eventually
        ↪ default to it's own output
        return ""
```

There are also more specialized methods for different node types, behaving similarly but giving more info. For more information on other available functions, check the `TTS` class.

Lastly, if you want to immediately speak a string of text, you can call `TTS.speak` directly:

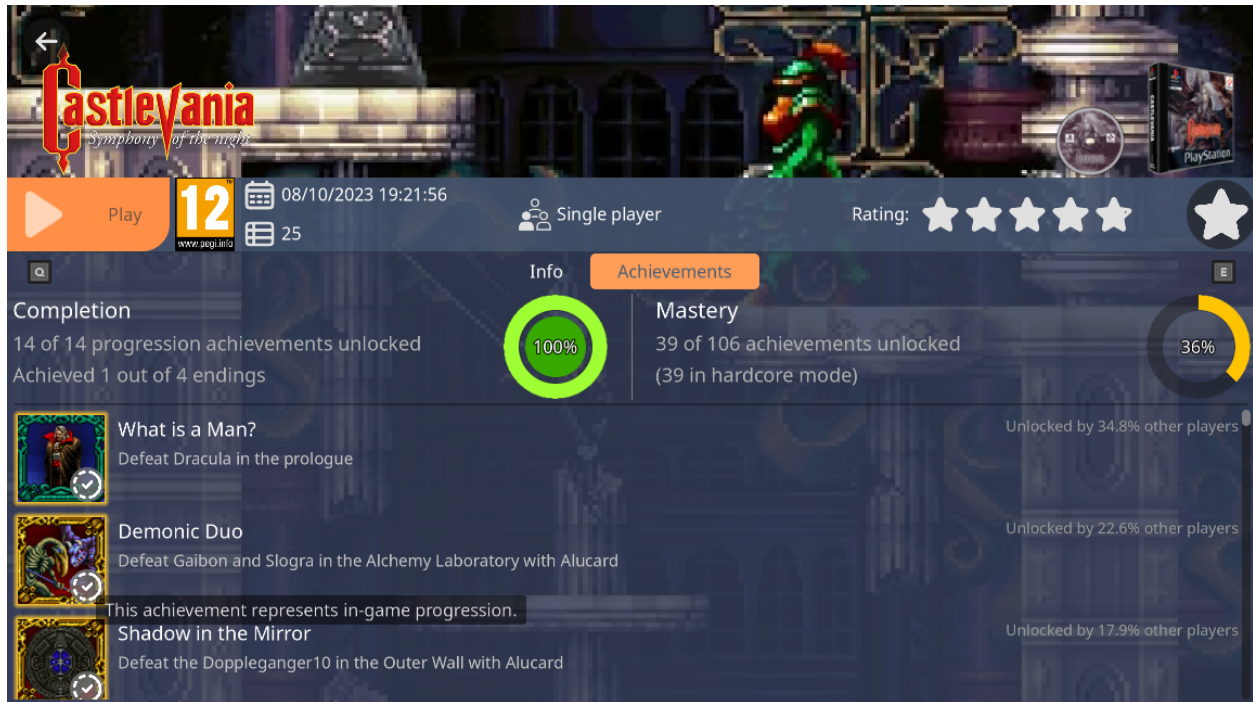
```
# Interrupts any current text
TTS.speak("Hello")
# Only speaks after any pending text is finished
TTS.speak("World!", false)
```


2.4 Integrations

2.4.1 RetroAchievements

RetroAchievements is a service that introduces achievements on classic systems, such as the SNES, Genesis, PlayStation, and many others. It also tracks user progression, statistics, and allows them to compete with other players.

RetroHub can fetch the user's personal achievement information and progression.



Setup

As with all integrations, you need to query if they're available before you can create them:

```
if not RetroAchievements.is_available():
    # Not available; warn the user of this
    ...
    return
```

To setup RetroAchievements, create an instance of *RetroAchievements*, and add it to the scene tree:

```
var retroachievements = RetroAchievements.new()
add_child(retroachievements)
```

Warning: Make sure to create only one instance in your whole theme. Multiple instances won't share cache, which will increase memory usage in your theme.

Obtaining achievement information

To avoid dealing with raw API calls and parsing information by hand, this integration provides a high-level API, available at [RetroAchievements](#), to fetch information about games and achievements directly. While for most scenarios this is sufficient, you can still access the raw API for other use cases, which is detailed on the [Raw API access](#) section.

To query for game and achievement information, you need to call [RetroAchievements.get_game_info](#) for a specific [RetroHubGameData](#):

```
var game_info = await retroachievements.get_game_info(game_data)
```

This call may take some time to complete, since it may connect to RetroAchievement's servers to fetch the information. Therefore, you'll need to use the `await` keyword to wait for the result without blocking your theme.

This call will always return a [RetroAchievements.GameInfo](#), but that doesn't necessarily mean it has been successful. You should only proceed if there is no error:

```
if game_info.err != RetroAchievements.GameInfo.Error.OK:
    # Handle error
    ...
    return
```

Once that's checked, you now have access to the game's information and achievements. Check the [RetroAchievements.GameInfo](#) and [RetroAchievements.Achievement](#) class references for more information on what's available:

```
print(game_info.game.player_count) # 7924
for achievement in game_info.achievements:
    print(achievement.title) # "First Blood"
    print(achievement.description) # "Kill an enemy for the first time"
    print(achievement.type) # Achievement.Type.NORMAL
    print(achievement.unlocked) # true
    print(achievement.unlocked_hard_mode) # false
    print(achievement.unlocked_count) # 6823
    print(achievement.unlocked_hard_mode_count) # 4712
```

Loading achievement icons

Every achievement has an icon associated with it. To reduce load in both RetroHub and RetroAchievements' servers, these icons need to be loaded on demand. To do so, you can call [Achievement.load_icon](#):

```
var texture = await achievement.load_icon()
# Now assign the texture to some UI element
...
```

This function will have some delay if it needs to download the icon. However, once downloaded, the icon is cached on the user's storage, so any subsequent calls will be much faster.

Warning: Keep this function call to the bare minimum necessary. Downloads are queued and cannot be canceled, so any other API calls that need to download information will need to wait for the current queue to finish.

Handling error scenarios

In the event that there is an error when fetching information, you should attempt to handle this scenario by informing the user on what's wrong, and any tips for fixing it.

Check the *GameInfo.Error* enumeration for a general description of each error. Since most of these errors are temporary, we recommend the following behaviors for tackling them:

- *RetroAchievements.is_available* returns **false**: The integration is disabled in the user settings, and themes cannot use it. You should inform the user of this, and allow them to enable it in the settings. You can open the settings panel directly with *RetroHubUI.open_app_config*. To get notified of such changes, connect to the *RetroHubConfig.config_updated* signal and listen for updates in the *integration_rcheevos_enabled* key.
- **ERR_INVALID_CRED**: The user's credentials are invalid; you should instruct them to fix this by opening settings, and ensuring the information is correct. You can also open the panel directly with *RetroHubUI.open_app_config*. The integration reloads the credentials automatically on the next API call when this error occurs, so you don't need to do anything else.
- **ERR_CONSOLE_NOT_SUPPORTED**: The requested game's system is not supported by RetroAchievements. You should inform the user of this, and not attempt to fetch any information.
- **ERR_GAME_NOT_FOUND**: The requested game was not found. RetroAchievements only accepts some specific game hashes per game, so it's possible that the requested game is actually supported, but the file hash does not match any of the supported ones. Because the hashing process is not trivial, you should display the game's hash to help users troubleshoot the issue. You can obtain this value from *RetroAchievements.get_game_hash*.
- **ERR_NETWORK**: There was an error connecting to RetroAchievements' servers. This could be a temporary network issue on the user, or RetroAchievements' servers may be down. As it's usually temporary and fixable, you should allow users to retry a given request.
- **ERR_INTERNAL**: There was an internal error in RetroHub and/or Godot. This should never happen, so it's important to track and report this scenario. In any case, do allow the user to retry, as it may be a temporary error.

Interpreting RetroAchievements' data

This serves as a primer on how RetroAchievements work, and how to interpret the data returned. For more information, you should check RetroAchievements' [documentation](#).

Hardmode

Hardmode refers to a special mode offered by emulators where some features are restricted, such as savestates, rewind, and others. RetroAchievements allows achievements to be unlocked in both normal and hardmode, and tracks them separately.

The *RetroAchievements.Achievement* class has separate properties to track progression on both modes:

- *unlocked* and *unlocked_count* track the normal mode.
- *unlocked_hard_mode* and *unlocked_hard_mode_count* track the hardmode.

If an achievement was unlocked in hardmode, it's important to show that to the user, especially if some achievements are unlocked in normal mode only.



Progression

Some achievements have a special type that denotes some in-game progression. The *Achievement.Type* enumeration lists such cases.

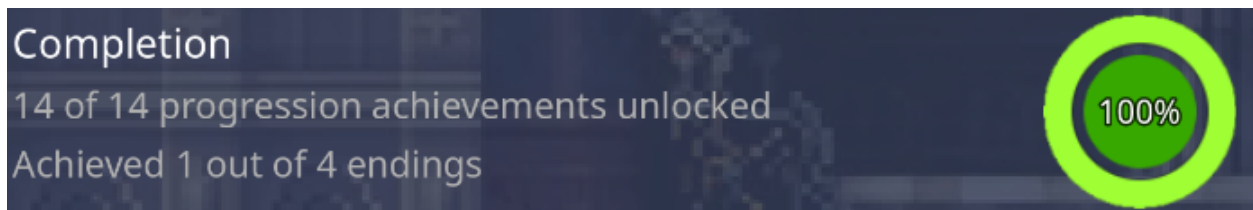
A game is considered completed if, and only if:

- All achievements marked as PROGRESSION are unlocked.

and

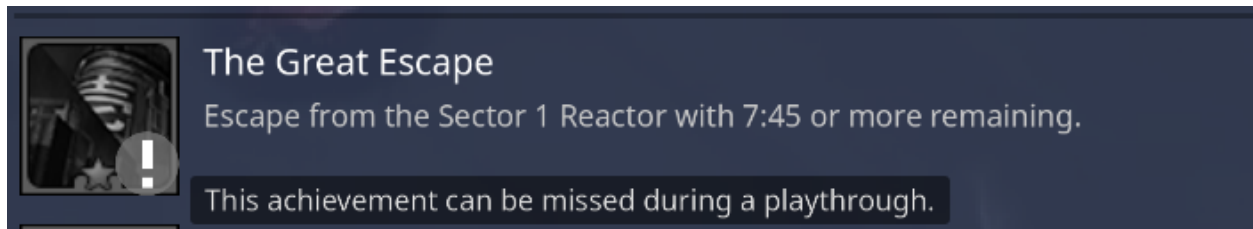
- Any achievement marked as WIN is unlocked.

You can use this information to show the user's progression in the game, and if they have completed it.



Missable achievements

Achievements marked as MISSABLE can be missed during a normal playthrough, so you may display them differently to inform this to the user.



Raw API access

If you want to access the RetroAchievements' raw API, you can do so by directly calling the exposed endpoints at *RetroAchievements.Raw*.

You will receive raw JSON responses that you'll need to parse however you see fit. Check RetroAchievements' [documentation](#) for more information.

For almost all of these endpoints, you will need to supply an authentication dictionary with the user's credentials. You can obtain one from *RetroAchievements.build_auth*:

```
var auth = RetroAchievements.build_auth()
var response = await RetroAchievements.Raw.get_achievement_of_the_week(auth)
```

All these endpoints return a *RetroAchievements.Raw.Response* that contains error codes and the response JSON body already parsed into a *Array* or *Dictionary*.

Warning: If the user's credentials are invalid (e.g. 401 HTTP errors), you will need to rebuild the authentication dictionary to refetch the credentials. You can do so by calling *RetroAchievements.build_auth* with the optional *force_refetch* argument to true.

APP DEVELOPMENT

This section includes some guides and information if you're planning to develop on RetroHub itself.

3.1 Contributing to RetroHub

3.1.1 Setting up the development environment

If you intend to develop for RetroHub, you must first setup the environment and all required dependencies.

Warning: The development version of RetroHub stores configuration on a separate directory from the normal version. This is to prevent modification and corruption to any existing setup. This directory is located at:

- **Windows:** C:\Users\<username>\RetroHub-Dev
- **macOS:** /Users/<username>/.**retrohub-dev**
- **Linux:** /home/<username>/.**retrohub-dev**

Compiling Godot

RetroHub requires some functionality not present on the Godot engine yet. While changes are kept to a minimum and are mostly pending code contributions and custom bugfixes, if you try to develop this project with a regular Godot version, you'll encounter errors due to missing functionality.

RetroHub [maintains a custom Godot fork](#) with custom changes being implemented on the `retrohub_patches_4x`. To compile it, simply follow the official Godot [compilation instructions](#).

Obtaining third-party libraries

Due to some existing third-party dependencies with different licenses, the RetroHub source code only includes dependencies with the same license as ours. For any missing dependencies, you'll need to manually obtain them in order to setup the development environment.

Video playback uses [FFmpeg](#) for decoding, which is under GPLv3. You will need to compile these dependencies yourself. Check out [retrohub-org/godot-videodecoder](#) for instructions on compiling for your system. Once you're done, move the compiled binaries to the appropriate system folder under `addons/godot-videodecoder`.

Note: You only need to compile binaries for your current system, unless you intend to export for other platforms.

Obtaining default themes

The default themes are not included in the source code, and are rather downloaded and packaged on our CI. Therefore, you'll need to obtain at least one theme in order to display any games. You can download official themes directly from their project page:

- [Default Theme](#)
- [EmulationStation Theme Wrapper](#)

Then place them at your RetroHub-Dev configuration directory.

3.1.2 Adding systems and emulators

RetroHub is a community project, and we welcome contributions to systems and emulators! If your favorite system or emulator is not yet supported by default, you can help us by adding it to the project.

The [Emulation General Wiki](#) is a fantastic resource for finding information about systems and emulators, and we recommend it as a reference in order to help your research when contributing information.

Warning: Be wary of the license of any assets you use: only assets under free/open licenses will be accepted.

Contributing a new system

A system is a particular piece of hardware that can run a specific range of games released for it. Currently, RetroHub distinguishes systems as computers, game engines, and consoles. Here are a few design choices that can help you understand what we're looking for in RetroHub:

- RetroHub is a “retrogaming library frontend”, and therefore there's no point in adding artificial system or categories that fall outside of this goal, like audio and video software.
- A system should be something that's usually not straightforward for the user to just run, meaning they need to open an emulator or some other software in order to play. Games that can be directly launched break RetroHub's philosophy. This implicitly means PC games and storefront like Steam, Epic, GOG and etc. are not accepted.
- A system **needs** to have some emulator associated with it. Systems without or with very poor emulation are not accepted.

Systems are defined in `base_config/systems.json`. It consists of an array of dictionaries, each one defining a new system:

```
{
  "name": "amstradcpc",
  "fullname": "Amstrad CPC",
  "category": "computer",
  "extension": [
    ".cdt",
    ".cpr",
    ".cpc",
    ".dsk",
    ".kcr",
    ".m3u",
    ".sna",
    ".tap",
```

(continues on next page)

(continued from previous page)

```

        ".voc",
        ".zip"
    ],
    "platform": "amstradcpc",
    "emulator": [
        {
            "retroarch": [
                "caprice32",
                "crocods",
                "mame"
            ]
        },
        "caprice32",
        "mame"
    ]
}

```

The specification for systems is the same as the end-user configuration, so check the [rh_systems.json specification](#) in order to understand each required field.

A new system will require two assets: a picture of the system (preferably a drawing of it, and not a photo), and its commercial logo. Both assets should be placed in `assets/systems/`, and be named `<system_name>_photo.png` and `<system_name>_logo.png` respectively.

Contributing a new emulator

An emulator is some software that RetroHub will launch in order to run games of a given system. The emulator must provide some CLI interface in order for RetroHub to launch it directly. Additionally, here's some guidelines for adding emulators:

- It's preferable to choose emulators that are cross-platform and/or open-source.
- The emulator must have a reasonable degree of compatibility with the given system.
- The emulator must be able to run games directly, without requiring further user input.
- The emulator must be relatively popular and active, and preferably marked as recommended on the [Emulation General Wiki](#).

Emulators are defined in `base_config/emulators.json`. It consists of an array of dictionaries, each one defining a new emulator:

```

{
    "name": "cemu",
    "fullname": "Cemu",
    "binpath": [
        {
            "windows": [
                "Emulators/cemu/Cemu.exe"
            ]
        },
        {
            "macos": [
                "/Applications/Cemu.app/Contents/MacOS/Cemu"
            ]
        }
    ]
}

```

(continues on next page)

(continued from previous page)

```

        ],
        {
            "linux": [
                "/bin/cemu",
                "/usr/bin/cemu"
            ]
        }
    ],
    "command": "{binpath} -g \"{rompath}\" -f"
},

```

The specification for emulators is the same as the end-user configuration, so check the *rh_emulators.json specification* in order to understand each required field.

A new emulator will require an icon to represent it. Place this under `assets/emulators/`, and name it `<emulator_name>.png`.

3.2 Using RetroHub as a base for your own project

3.2.1 Licensing Concerns

RetroHub is freely available under the [MIT License](#). In a nutshell, you can use it for any purpose, including commercial usage. There are still some considerations you'll need to have in mind if you intend to use the RetroHub codebase.

You can find detailed licensing information on RetroHub's **About** panel.

Bundled libraries

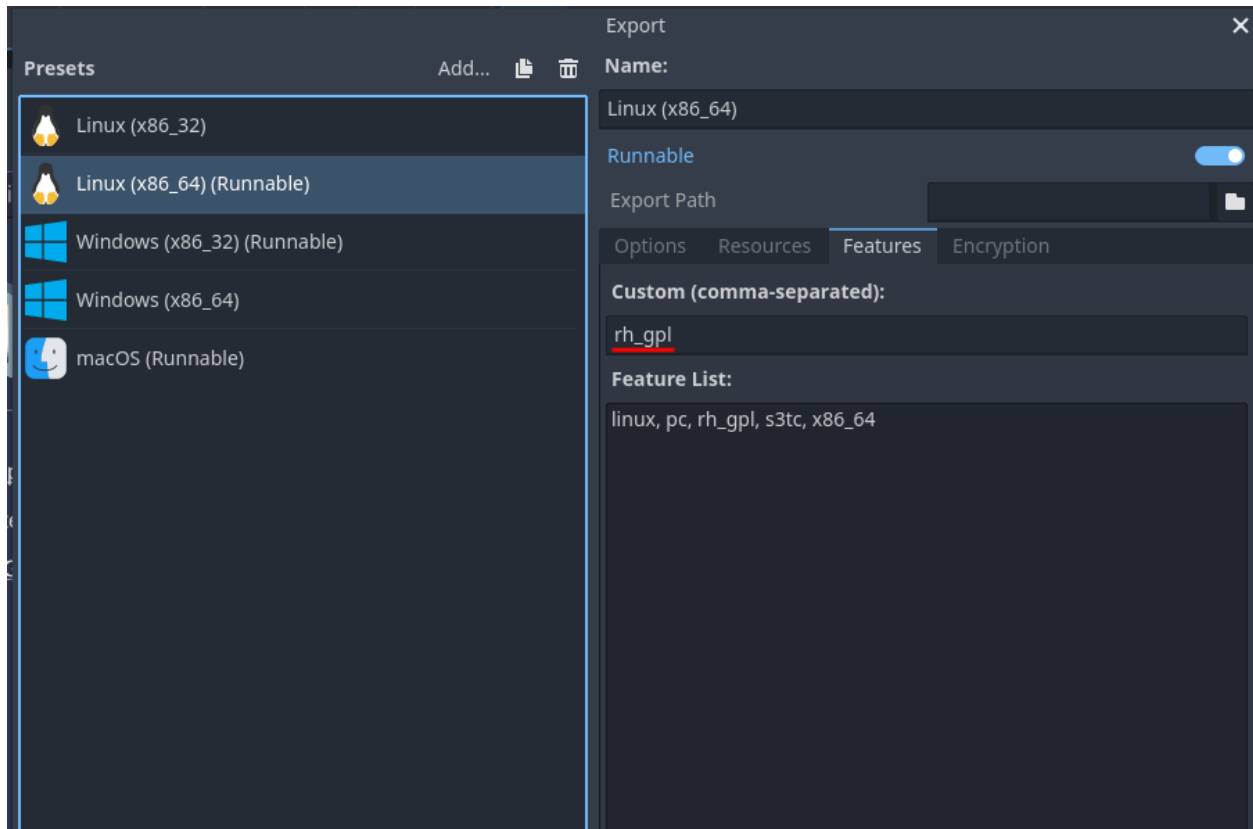
The released version of RetroHub makes usage of some [GPLv3](#) third-party libraries. This retroactively forces the distributed binary to be under GPLv3 as well. If you intend to use the following libraries, **your project will have to be under GPLv3**:

- [FFmpeg](#) (used for video playback)

If you cannot use one or more of these libraries, here are some instructions in order to remove them:

- **FFmpeg**: This library is built as a GDExtension from [retrohub-org/godot-videodecoder](#) and bundled in our CI. Therefore, you don't need to do anything in order to remove it. Make sure to modify the existing code to prevent RetroHub from trying to load unsupported formats like `.mp4`.

If your project is released with a license other than GPLv3, modify the export settings in order to remove the `rh_gpl` flag.



Scraper services

API keys that have been provided for RetroHub are exclusive and cannot be used for your own project. If you intend to support such scraper services, you'll have to get in contact with each service in order to request permission for your project:

- [ScreenScraper](#)

Used assets

RetroHub's logo is licensed under [CC-BY-4.0](#), being originally designed by [Lynn Pepin](#). While this license permits you to use this logo for your own projects (with attribution), please be reasonable with your usage. If your project is intended to stay connected to the original project then it's fine, but otherwise we ask you to design your own branding.

RetroHub ships with system assets that were manually obtained from ScreenScraper. Permission was given for the usage of these assets, but this permission is exclusive to RetroHub. If you intend to use these assets for your own projects, you'll have to get in contact with ScreenScraper to request permission as well. Regardless of that, these system assets, along with emulators icons, **cannot** be used in commercial projects, so if that's the case, they **must be** entirely removed from `assets/systems` and `assets/emulators`.

For interface sounds, although most of them are [CC0](#) (public domain), some of them are not, and you'll have to consult each one of them individually.

CLASS API

This page documents the public API of RetroHub.

4.1 Data classes

4.1.1 RetroHubGameData

Represents a game entry from the user's library. Holds all informations available about a game. It is generated by *RetroHub* when the app is launched, and passed to themes through the *game_received* signal.

The received object is unique and shared among the app. An instance can also be used to uniquely identify this game, which is done for other parts of the API.

Enumerations

enum **BoxTextureRegions**:

- **BACK** - Back cover.
- **SPINE** - Spine portion.
- **FRONT** - Front cover.

Properties

Type	Name	Default
bool	<i>has_metadata</i>	false
bool	<i>has_media</i>	false
<i>RetroHubSystem-Data</i>	<i>system</i>	<i>other</i>
String	<i>system_path</i>	""
String	<i>name</i>	<i>other</i>
String	<i>path</i>	<i>other</i>
String	<i>description</i>	""
float	<i>rating</i>	0.0
String	<i>release_date</i>	""
String	<i>developer</i>	""
String	<i>publisher</i>	""
Array	<i>genres</i>	[]
String	<i>num_players</i>	""
String	<i>age_rating</i>	""
bool	<i>favorite</i>	false
int	<i>play_count</i>	0
String	<i>last_played</i>	""
String	<i>emulator</i>	""
Dictionary	<i>box_texture_regions</i>	{}

bool **has_metadata** = false

Whether there's metadata available. If false, only the *name* and *path* properties are guaranteed to be valid. In that case, *name* will be set to the game's file name.

bool **has_media** = false

Whether there's media available for this game. If false, any calls to retrieve media will fail.

Note: This only denotes the existence of any media, and doesn't guarantee that all media types are available.

RetroHubSystemData **system** = <valid system instance>

The *RetroHubSystemData* this game belongs to.

String **system_path** = ""

The physical system folder this game comes from. Because of system remaps for different regions, the *system* may be pointing to a different system from the game file's location. This is used internally by RetroHub to determine the correct system for the game when launching the game.

String **name** = "..."

The game's name. This field is always set even when *has_metadata* is `false`, in which case it will be set to the game's file name.

`String path = "..."`

The game's file absolute path. This field is always set even when *has_metadata* is `false`.

`String description = ""`

The game's description.

`float rating = 0.0`

The game's rating, which ranges between `0.0` and `1.0`.

`String release_date = ""`

The game's release date, in ISO8601 format. Use *RegionUtils.localize_date()* to convert it to the user's local date format.

`String developer = ""`

The game's developer.

`String publisher = ""`

The game's publisher.

`Array genres = []`

The game's genres. This is an array of strings.

`String num_players = ""`

The game's number of players, in the format "{min_players}-{max_players}". This is used to represent both singleplayer ("1-1") and multiplayer ("2-4") games.

`String age_rating = ""`

The game's age rating, in the format "{esrb}/{pegi}/{cero}". Values range from 0 to 5, where 0 means unknown, and 1 through 5 increasing levels of age rating. Use *RegionUtils.localize_age_rating* to get the appropriate rating as a *Texture*.

`bool favorite = false`

Whether the game is marked as a favorite.

```
int play_count = 0
```

The number of times the game has been played.

```
String last_played = ""
```

The date the game was last played, in ISO8601 format. Use [RegionUtils.localize_date\(\)](#) to convert it to the user's local date format.

Warning: If never played before, the string will be set to "null".

```
String emulator = ""
```

Used internally by RetroHub to specify a specific emulator to be used when launching the game.

```
Dictionary box_texture_regions = {}
```

Defines regions of the game's box art texture. This allows you to extract specific regions of the box art, such as the front portion. See [RetroHubMedia.get_box_texture_region\(\)](#) for more information.

Methods

Type	Name
bool	<i>sort</i>
void	<i>copy_from</i>

```
bool sort (a: RetroHubGameData, b: RetroHubGameData)
```

Custom sorting implementation for *RetroHubGameData* instances. This is used internally by RetroHub to sort games by name, in ascending order. You can also use this if you need to sort game datas, by using [Array.sort_custom\(\)](#).

```
var game_datas = [...]  
  
game_datas.sort_custom(RetroHubGameData, "sort")  
  
# game_datas will now be sorted by names
```

```
void copy_from (other: RetroHubGameData)
```

Copies all information from existing game data, creating a duplicate.

4.1.2 RetroHubSystemData

Represents a system from the user's library. Holds all informations available about a system. It is generated by *RetroHub* when the app is launched, and passed to themes through the *system_received* signal.

The received object is unique and shared among the app. An instance can also be used to uniquely identify this system, which is done for other parts of the API.

Enumerations

enum **Category**:

- **Console** - A retro console.
- **Arcade** - An arcade machine.
- **Computer** - A retro computer.
- **GameEngine** - A game engine.
- **ModernConsole** - A modern console (7th generation upwards).

Properties

Type	Name	Default
String	<i>name</i>	<i>other</i>
String	<i>fullname</i>	""
String	<i>platform</i>	""
Category	<i>category</i>	0
int	<i>num_games</i>	""

String **name** = ". . ."

The system's "short" name. This field is always set, and uniquely identifies this system.

String **fullname** = ""

The system's full name.

String **platform** = ""

The system's platform. While usually the same as the *name* field, it can be different for systems that are very similar to a base system, such that RetroHub uses the same configuration among them. For example, Nintendo 64 DD on Nintendo 64 and Sega Genesis CD on Sega Genesis.

Category **category** = 0

The system's category.

```
int num_games = 0
```

The number of games detected for this system. In practice, this will always be larger than 0, as RetroHub doesn't send system data if there are no games for it. This can be used to known in advance how many game datas will arrive for that specific system.

Methods

Type	Name
<i>Category</i>	<i>category_to_idx</i>
String	<i>idx_to_category</i>

```
Category category_to_idx ( String category )
```

Converts a category name to the enum value. Used internally.

```
String idx_to_category ( Category category )
```

Converts a category enum value to a string name. Used internally.

4.1.3 RetroHubGameMediaData

Holds media information for a given *RetroHubGameData*. Retrieved through calls to *RetroHubMedia*.

Warning: Of all the data classes, this one is the heaviest to have in memory, as all game assets are loaded into VRAM. Therefore, you must be careful to ensure this is deleted when you don't need it anymore.

Properties

Type	Name	Default
ImageTexture	<i>logo</i>	null
ImageTexture	<i>screenshot</i>	null
ImageTexture	<i>title_screen</i>	null
VideoStream	<i>video</i>	null
ImageTexture	<i>box_render</i>	null
ImageTexture	<i>box_texture</i>	null
ImageTexture	<i>support_render</i>	null
ImageTexture	<i>support_texture</i>	null
<undefined>	<i>manual</i>	null

```
ImageTexture logo = null
```

The game's logo, usually the title art.

`ImageTexture screenshot = null`

A screenshot of the game in action.

`ImageTexture title_screen = null`

A screenshot of the game's title screen.

`VideoStream video = null`

A video of the game in action.

`ImageTexture box_render = null`

A pre-rendered image of the game's box art.

`ImageTexture box_texture = null`

A raw texture of the game's box art. This can be used to create a 3D model.

Note: Automatic generation of box art models is planned for a future release.

`ImageTexture support_render = null`

A pre-rendered image of the game's physical support (*cartridge, CD, etc.*)

`ImageTexture support_texture = null`

A raw texture of the game's physical support (*cartridge, CD, etc.*). This can be used to create a 3D model.

Note: Automatic generation of physical support models is planned for a future release.

`manual = null`

The game's manual. Currently unimplemented, will always be `null`.

4.1.4 ConfigData

This class holds app configuration information from RetroHub. This is not directly accessible from themes, and instead passed on through the *config_ready* and *config_updated* signals.

Constants

Type	Name
String	<i>KEY_CONFIG_VERSION</i>
String	<i>KEY_IS_FIRST_TIME</i>
String	<i>KEY_GAMES_DIR</i>
String	<i>KEY_CURRENT_THEME</i>
String	<i>KEY_LANG</i>
String	<i>KEY_FULLSCREEN</i>
String	<i>KEY_VSYNC</i>
String	<i>KEY_RENDER_RESOLUTION</i>
String	<i>KEY_REGION</i>
String	<i>KEY_RATING_SYSTEM</i>
String	<i>KEY_DATE_FORMAT</i>
String	<i>KEY_SYSTEM_NAMES</i>
String	<i>KEY_SCRAPER_HASH_FILE_SIZE</i>
String	<i>KEY_SCRAPER_SS_USE_CUSTOM_ACCOUNT</i>
String	<i>KEY_SCRAPER_SS_MAX_THREADS</i>
String	<i>KEY_CUSTOM_INPUT_REMAP</i>
String	<i>KEY_INPUT_KEY_MAP</i>
String	<i>KEY_INPUT_CONTROLLER_MAP</i>
String	<i>KEY_INPUT_CONTROLLER_MAIN_AXIS</i>
String	<i>KEY_INPUT_CONTROLLER_SECONDARY_AXIS</i>
String	<i>KEY_INPUT_CONTROLLER_ICON_TYPE</i>
String	<i>KEY_INPUT_CONTROLLER_ECHO_PRE_DELAY</i>
String	<i>KEY_INPUT_CONTROLLER_ECHO_DELAY</i>
String	<i>KEY_VIRTUAL_KEYBOARD_LAYOUT</i>
String	<i>KEY_VIRTUAL_KEYBOARD_TYPE</i>
String	<i>KEY_VIRTUAL_KEYBOARD_SHOW_ON_CONTROLLER</i>
String	<i>KEY_VIRTUAL_KEYBOARD_SHOW_ON_MOUSE</i>
bool	<i>KEY_ACCESSIBILITY_SCREEN_READER_ENABLED</i>
String	<i>KEY_CUSTOM_GAMEMEDIA_DIR</i>
String	<i>KEY_UI_VOLUME</i>
String	<i>KEY_INTEGRATION_RCHEEVOS_ENABLED</i>

String **KEY_CONFIG_VERSION** = "config_version"

Key for the *config_version* property.

String **KEY_IS_FIRST_TIME** = "is_first_time"

Key for the *is_first_time* property.

String **KEY_GAMES_DIR** = "games_dir"

Key for the *games_dir* property.

String **KEY_CURRENT_THEME** = "current_theme"

Key for the *current_theme* property.

`String KEY_LANG = "lang"`

Key for the *lang* property.

`String KEY_FULLSCREEN = "fullscreen"`

Key for the *fullscreen* property.

`String KEY_VSYNC = "vsync"`

Key for the *vsync* property.

`String KEY_RENDER_RESOLUTION = "render_resolution"`

Key for the *render_resolution* property.

`String KEY_REGION = "region"`

Key for the *region* property.

`String KEY_RATING_SYSTEM = "rating_system"`

Key for the *rating_system* property.

`String KEY_DATE_FORMAT = "date_format"`

Key for the *date_format* property.

`String KEY_SYSTEM_NAMES = "system_names"`

Key for the *system_names* property.

`String KEY_SCRAPER_HASH_FILE_SIZE = "scraper_hash_file_size"`

Key for the *scraper_hash_file_size* property.

`String KEY_SCRAPER_SS_USE_CUSTOM_ACCOUNT = "scraper_ss_use_custom_account"`

Key for the *scraper_ss_use_custom_account* property.

`String KEY_SCRAPER_SS_MAX_THREADS = "scraper_ss_max_threads"`

Key for the *scraper_ss_max_threads* property.

String **KEY_CUSTOM_INPUT_REMAP** = "custom_input_remap"

Key for the *custom_input_remap* property.

String **KEY_INPUT_KEY_MAP** = "input_key_map"

Key for the *input_key_map* property.

String **KEY_INPUT_CONTROLLER_MAP** = "input_controller_map"

Key for the *input_controller_map* property.

String **KEY_INPUT_CONTROLLER_MAIN_AXIS** = "input_controller_main_axis"

Key for the *input_controller_main_axis* property.

String **KEY_INPUT_CONTROLLER_SECONDARY_AXIS** = "input_controller_secondary_axis"

Key for the *input_controller_secondary_axis* property.

String **KEY_INPUT_CONTROLLER_ICON_TYPE** = "input_controller_icon_type"

Key for the *input_controller_icon_type* property.

String **KEY_INPUT_CONTROLLER_ECHO_PRE_DELAY** = "input_controller_echo_pre_delay"

Key for the *input_controller_echo_pre_delay* property.

String **KEY_INPUT_CONTROLLER_ECHO_DELAY** = "input_controller_echo_delay"

Key for the *input_controller_echo_delay* property.

String **KEY_VIRTUAL_KEYBOARD_LAYOUT** = "virtual_keyboard_layout"

Key for the *virtual_keyboard_layout* property.

String **KEY_VIRTUAL_KEYBOARD_TYPE** = "virtual_keyboard_type"

Key for the *virtual_keyboard_type* property.

String **KEY_VIRTUAL_KEYBOARD_SHOW_ON_CONTROLLER** =
"virtual_keyboard_show_on_controller"

Key for the *virtual_keyboard_show_on_controller* property.

String **KEY_VIRTUAL_KEYBOARD_SHOW_ON_MOUSE** = "virtual_keyboard_show_on_mouse"

Key for the *virtual_keyboard_show_on_mouse* property.

```
String KEY_ACCESSIBILITY_SCREEN_READER_ENABLED =
    "accessibility_screen_reader_enabled"
```

Key for the *accessibility_screen_reader_enabled* property.

```
String KEY_CUSTOM_GAMEMEDIA_DIR = "custom_gamemedia_dir"
```

Key for the *custom_gamemedia_dir* property.

```
String KEY_UI_VOLUME = "ui_volume"
```

Key for the *ui_volume* property.

```
String KEY_INTEGRATION_RCHEEVOS_ENABLED = "integration_rcheevos_enabled"
```

Key for the *integration_rcheevos_enabled* property.

Properties

Type	Name	Default
int	<i>config_version</i>	1
bool	<i>is_first_time</i>	true
String	<i>games_dir</i>	<i>FileUtils.get_home_dir()</i> + "/ROMS"
String	<i>current_theme</i>	"default"
String	<i>lang</i>	"en"
bool	<i>fullscreen</i>	true
bool	<i>vsync</i>	true
int	<i>render_resolution</i>	100
String	<i>region</i>	"usa"
String	<i>rating_system</i>	"esrb"
String	<i>date_format</i>	"mm/dd/yyyy"
Dictionary	<i>system_names</i>	Dictionary
int	<i>scraper_hash_file_size</i>	64
bool	<i>scraper_ss_use_custom_account</i>	false
int	<i>scraper_ss_max_threads</i>	6
String	<i>custom_input_remap</i>	""
Dictionary	<i>input_key_map</i>	Dictionary
Dictionary	<i>input_controller_map</i>	Dictionary
int	<i>input_controller_main_axis</i>	JOY_ANALOG_LX
int	<i>input_controller_secondary_axis</i>	JOY_ANALOG_RX
int	<i>input_controller_icon_type</i>	0
float	<i>input_controller_echo_pre_delay</i>	0.75
float	<i>input_controller_echo_delay</i>	0.15
String	<i>virtual_keyboard_layout</i>	"qwerty"

continues on next page

Table 2 – continued from previous page

Type	Name	Default
String	<i>virtual_keyboard_type</i>	"builtin"
bool	<i>virtual_keyboard_show_on_controller</i>	true
bool	<i>virtual_keyboard_show_on_mouse</i>	false
bool	<i>accessibility_screen_reader_enabled</i>	false
String	<i>custom_gamemedia_dir</i>	""
int	<i>ui_volume</i>	100
bool	<i>integration_rcheevos_enabled</i>	false

```
int config_version = 1
```

Specifies the latest internal configuration version. This is used internally to determine if the configuration needs to be updated.

```
bool is_first_time = true
```

If `true`, this is the first time the user is running RetroHub, therefore the first-time wizard will be shown. Themes are not loaded until the first-time wizard is completed, so in practice this property is always `false` when queried by the theme.

```
String games_dir = FileUtils.get_home_dir() + "/ROMS"
```

The user's game library. This is the directory where RetroHub will look for games.

```
String current_theme = "default"
```

The currently loaded theme. If lacking any extension, it will load a default theme at `res://default_themes/{current_theme}.pck`. Otherwise, the theme is loaded from `<retrohub_config_dir>/themes/{current_theme}`.

```
String lang = "en"
```

Language. Used throughout the UI and for themes that support localization as well.

Valid values:

- "en": English
-

```
bool fullscreen = true
```

If `true`, RetroHub will run in fullscreen mode, otherwise it will run in windowed mode.

```
bool vsync = true
```

If `true`, RetroHub will enable V-Sync.

```
int render_resolution = 100
```

Theme render resolution, in percentage. Doesn't affect RetroHub's UI. Can range from 50 to 100.

`String region = "usa"`

User's preferred region.

Valid values:

- "usa": USA
 - "eur": Europe
 - "jpn": Japan
-

`String rating_system = "esrb"`

User's preferred age rating system.

Valid values:

- "esrb": ESRB
 - "pegi": PEGI
 - "cero": CERO
-

`String date_format = "mm/dd/yyyy"`

User's preferred date formatting.

Valid values:

- "mm/dd/yyyy": month / day / year
 - "dd/mm/yyyy": day / month / year
 - "yyyy/mm/dd": year / month / day
-

`Dictionary system_names = Dictionary`

Chosen system remaps per region. Each key determines the displayed name for the system.

`int scraper_hash_file_size = 64`

Maximum allowed size, in MB, of game files to be hashed for scraping. This is used to avoid scraping large files. Set to 0 to disable this limit.

`bool scraper_ss_use_custom_account = false`

If true, uses a user's account details when using ScreenScraper. Used to bypass global API quota limits.

`int scraper_ss_max_threads = 6`

Maximum amount of threads to use when scraping with ScreenScraper with a custom account. Set this to limit thread usage to a specific number.

Note: RetroHub will always check how many threads your account has available. If this value is bigger than your allowed thread count, RetroHub will respect your account's thread limit.

String **custom_input_remap** = ""

Custom controller layout string. This follows the SDL format. See [Controller Layouts](#) for more information.

Dictionary **input_key_map** = Dictionary

Input map for keyboard actions. Keys are RetroHub's input actions, available on [Input actions](#). Values are KEY_* constants available in [@GlobalScope](#).

Dictionary **input_controller_map** = Dictionary

Input map for controller actions. Keys are RetroHub's input actions, available on [Input actions](#). Values are JOY_* constants available in [@GlobalScope](#).

int **input_controller_main_axis** = JOY_ANALOG_LX

Main joystick axis for controller input. This is used for navigation.

Valid values:

- JOY_ANALOG_LX: Left stick
 - JOY_ANALOG_RX: Right stick
-

int **input_controller_secondary_axis** = JOY_ANALOG_RX

Secondary joystick axis for controller input. This is used for scrolling.

Valid values:

- JOY_ANALOG_LX: Left stick
 - JOY_ANALOG_RX: Right stick
-

int **input_controller_icon_type** = 0

Type of controller icons.

Valid values:

- 0: Detect controller automatically
 - 1: Xbox 360
 - 2: Xbox One
 - 3: Xbox Series
 - 4: PS3
-

- 5: PS4
- 6: PS5
- 7: Switch (Controller)
- 8: Switch (Joy-Con)
- 9: Steam Controller
- 10: Steam Deck
- 11: Amazon Luna
- 12: Google Stadia

float `input_controller_echo_pre_delay` = 0.75

Pre-delay in seconds for controller echo events. This is the amount of time a user has to keep a button pressed/axis moved before it starts repeating.

float `input_controller_echo_delay` = 0.15

Delay in seconds between controller echo events. This dictates the frequency of controller echo events.

String `virtual_keyboard_layout` = "qwerty"

Preferred virtual keyboard layout.

Valid values:

- "qwerty": QWERTY
- "qwertz": QWERTZ
- "azerty": AZERTY


String `virtual_keyboard_type` = "builtin"

What virtual keyboard to use.

Valid values:

- "builtin": App built-in virtual keyboard
- "steam": Steam/Steam Deck virtual keyboard

bool `virtual_keyboard_show_on_controller` = true

If true, shows the virtual keyboard when a controller is connected and a given valid node is selected with .

bool `virtual_keyboard_show_on_mouse` = false

If true, shows the virtual keyboard when a mouse click or touch event is done on a valid node.

`bool accessibility_screen_reader_enabled = false`

If `true`, enables screen reader support.

`String custom_gamemedia_dir = ""`

If specified, stores downloaded media files separately from the main configuration directory, on a user specified path.

`int ui_volume = 100`

UI volume, in percentage. Can range from 0 to 100.

`bool integration_rcheevos_enabled = false`

If `true`, enables RetroHub's integration with the [RetroAchievements](#) service.

4.2 Main singletons

4.2.1 RetroHub

This is the main class to interact with RetroHub. It is also the source of all information regarding the user's library.

Constants

Type	Name
<code>int</code>	<code>version_major</code>
<code>int</code>	<code>version_minor</code>
<code>int</code>	<code>version_patch</code>
<code>String</code>	<code>version_extra</code>
<code>String</code>	<code>version_str</code>

`int version_major`

Major version of RetroHub. A change in the major version number indicates a breaking change to RetroHub or Godot. Themes designed for a different major version are highly unlikely to work without significant modifications regarding RetroHub's API or even Godot changes.

`int version_minor`

Minor version of RetroHub. A change in the minor version number indicates minor changes to RetroHub's API. Themes designed for a different minor version are highly unlikely to work without some modifications regarding RetroHub's API.

`int version_patch`

Patch version of RetroHub. A change in the patch version number indicates mostly bug fixes or new features implemented in a backwards-compatible way. Themes designed for a different patch version are expected to work without issues.

String version_extra

Extra information about the version, such as `alpha` or `beta`. On release versions, this is empty.

String version_str

Fully assembled version string, for convenience, in the format `{major}.{minor}.{patch}{extra}`.

Signals

app_initializing ()

Called only one time, when the app is initializing. Themes aren't able to catch this signal.

app_closing ()

Called only one time, when the app is closing.

app_received_focus ()

Called when the app receives focus.

app_lost_focus ()

Called when the app loses focus.

app_returning (system_data: *RetroHubSystemData*, game_data: *RetroHubGameData*)

Called when the app is returning from a launched game. This signal is emitted after a full theme load cycle, notably after all the *system_received* and *game_received* signals have been emitted.

system_receive_start ()

Called when system data is about to be sent to the theme. This signal is emitted only if there's system data available.

system_received (system_data: *RetroHubSystemData*)

Information about a *RetroHubSystemData*.

system_receive_end ()

Called when all system data has been sent. Game data will follow after this signal.

game_receive_start ()

Called when game data is about to be sent to the theme. This signal is emitted after all system data has been sent.

game_received (game_data: *RetroHubGameData*)

Information about a *RetroHubGameData*.

game_receive_end ()

Called when all game data has been sent. By this point, all information about the user's gaming library has been sent.

Methods

Type	Name
void	<i>set_curr_game_data</i>
bool	<i>is_main_app</i>
bool	<i>is_input_echo</i>
void	<i>quit</i>
void	<i>launch_game</i>
void	<i>request_theme_reload</i>

void **set_curr_game_data** (game_data: *RetroHubGameData*)

Sets the currently selected game data. This is needed to launch games, but also so users can edit and/or scrape game information from RetroHub.

bool **is_main_app** ()

Returns whether the theme is running on the main app, or under the theme helper addon.

bool **is_input_echo** ()

Returns whether the current input event was generated by RetroHub. This is used to simulate echo events for controller inputs. This can be undesirable for some scenarios, so in that case, use this to ignore such events.

```
func _input(event):  
    if RetroHub.is_input_echo():  
        return  
  
    ...
```

void **quit** ()

Request the app to quit. Theme configuration will be saved.

void **launch_game** ()

Launches the currently selected game. Must be set prior with *set_curr_game_data*. After this call, the theme will be unloaded, so this should be treated as an exit point where no more code from your theme will be executed.

void **request_theme_reload** ()

Requests RetroHub to fully reload your theme. This is useful if some major configuration was changed from the app, and it's easier to reload the theme than to try to update it in place.

4.2.2 RetroHubMedia

This class handles media loading for *RetroHubGameData* instances.

Enumerations

enum **Type**:

- **LOGO = 1** - Game logo.
- **SCREENSHOT = 2** - Game screenshot.
- **TITLE_SCREEN = 4** - Game title screen.
- **VIDEO = 8** - Game video.
- **BOX_RENDER = 16** - Game box render.
- **BOX_TEXTURE = 32** - Game box texture.
- **SUPPORT_RENDER = 64** - Game physical support render.
- **SUPPORT_TEXTURE = 128** - Game physical support texture.
- **MANUAL = 256** - Game manual.
- **ALL = 511** - All media types.

Signals

media_loaded (media_data: *RetroHubGameMediaData*, game_data: *RetroHubGameData*, types: *Type*)

Emitted when an asynchronous request for media has finished loading. Returns the original game data and requested types when doing the request, which allows to uniquely identify this request among others.

Methods

Type	Name
void	<i>clear_media_cache</i>
void	<i>clear_all_media_cache</i>
Array	<i>convert_type_bitmask_to_list</i>
String	<i>convert_type_to_media_path</i>
<i>RetroHubGameMediaData</i>	<i>retrieve_media_data</i>
<i>RetroHubGameMediaData</i>	<i>retrieve_media_blurhash</i>
void	<i>retrieve_media_data_async</i>
<i>RetroHubGameMediaData</i>	<i>retrieve_media_data_and_blurhash_async</i>
void	<i>cancel_media_data_async</i>
Texture2D	<i>get_box_texture_region</i>

void **clear_media_cache** (data: *RetroHubGameMediaData*)

Clears the internal media cache for the given data. The media will thus be freed if the theme doesn't reference it anymore.

void **clear_all_media_cache** ()

Clears the entire internal media cache. Unreference data will be freed.

Array **convert_type_bitmask_to_list** (bitmask: *Type*)

Converts a bitmask of media types to a list with each type separated, to make it easily iterable.

```
# Returns [Type.LOGO, Type.VIDEO]
convert_type_bitmask_to_list(Type.LOGO | Type.VIDEO)
```

String **convert_type_to_media_path** (type: *Type*)

Converts a media type to its corresponding path in the RetroHub media directory. Used internally when saving downloaded media. If an invalid/unknown type is passed, returns the string "unknown".

```
# Returns "support_render"
convert_type_to_media_path(Type.SUPPORT_TEXTURE)
```

RetroHubGameMediaData **retrieve_media_data** (game_data: *RetroHubGameData*, types: *Type* = *Type*.ALL)

Retrieves media for the requested game data. If *types* is not specified, all media types will be retrieved.

Returns a *RetroHubGameMediaData* instance with the requested media data. If game data doesn't have *has_media* set to *true*, returns *null* instead.

This function blocks while media is loaded, so if you want to load plenty of media files without freezing your theme, use *retrieve_media_data_async* instead.


```
# Returns a RetroHubMediaData instance with all media types
var media_data = retrieve_media_data(game_data)

# Returns a RetroHubMediaData instance with only the logo and video
var media_data = retrieve_media_data(game_data, Type.LOGO | Type.VIDEO)
```

Note: The *Type* enum is a bitmask, so you can combine any types you need using the bitwise OR operator (`|`).

Warning: Never assume a given media type is available. Always check that the requested media file is non null before using it.

RetroHubGameData **retrieve_media_blurhash** (game_data: *RetroHubGameData*, types: *Type* = *Type.ALL*)

Fetches BlurHash data for existing game media, returning media data with blurred previews of the requested media types. Although the function is blocking, fetching a BlurHash is very quick, so this can be repeatedly called without freezing your theme.

Note: The *Type* enum is a bitmask, so you can combine any types you need using the bitwise OR operator (`|`).

Warning: Never assume a given media type is available. Always check that the requested media file is non null before using it.

void **retrieve_media_data_async** (game_data: *RetroHubGameData*, types: *Type* = *Type.ALL*, priority: bool = false)

Retrieves media for the requested game data asynchronously. If *types* is not specified, all media types will be retrieved. If *priority* is true, the request will be given priority over pending async requests.

This function returns immediately, and the *media_loaded* signal will be emitted when the media has finished loading. You should keep a reference to the supplied game data and types, to later be able to identify this request among others.

If game data doesn't have *has_media* set to true, the request is never made, and so the signal will never be emitted.

```
connect("media_loaded", self, "on_media_loaded")

# Loads all media types asynchronously
retrieve_media_data_async(game_data, Type.ALL)

func on_media_loaded(media_data, game_data, types):
    if self.game_data == game_data and types == Type.ALL:
        # This was the request we asked. Use media
        ...
```

Note: The *Type* enum is a bitmask, so you can combine any types you need using the bitwise OR operator (`|`).

Warning: Never assume a given media type is available. Always check that the requested media file is not null before using it.

RetroHubGameMediaData **retrieve_media_data_and_blurhash_async** (game_data: *RetroHubGameData*, types: *Type* = *Type*.ALL, priority: *bool* = false)

Calls *retrieve_media_data_async* and *retrieve_media_blurhash* at the same time. This provides you with a quick media preview while the media request is handled asynchronously.

Note: The *Type* enum is a bitmask, so you can combine any types you need using the bitwise OR operator (*|*).

Warning: Never assume a given media type is available. Always check that the requested media file is not null before using it.

void **cancel_media_data_async** (game_data: *RetroHubGameData*)

Cancels all pending asynchronous request done with *retrieve_media_data_async* under that game data. This is useful if you want to cancel a request that you don't need anymore, to avoid wasting resources.

Note: This will cancel all requests for a given game_data, even if multiple requests were made with different types.

Note: A request may still finish and be sent over the *media_loaded* signal even after calling this function.

Texture2D **get_box_texture_region** (game_data: *RetroHubGameData*, media_data: *RetroHubGameMediaData*, region: *RetroHubGameData.BoxTextureRegions*, rotate: *bool* = true)

Extracts the specified box region from a game box art. This allows you to only use relevant portions of the box art, like the front cover. If *rotate* is true, it will also rotate the region to ensure text is displayed left-to-right.

If the media data doesn't have any box art, or the game data *box_texture_regions* doesn't contain the specified region, this function returns null.

```
# Get the front cover of the box texture.
get_box_texture_region(game_data, media_data, RetroHubGameData.BoxTextureRegions.FRONT)

# Get the spine portion of the box texture, without rotating it (since they're usually
↳vertical).
get_box_texture_region(game_data, media_data, RetroHubGameData.BoxTextureRegions.SPINE,
↳false)
```

4.2.3 RetroHubConfig

This class handles all configuration aspects of RetroHub.

Signals

config_ready (config_data: *ConfigData*)

Emitted when RetroHub has finished loading app configs. This signal is emitted after theme initialization.

config_updated (key: *String*, old_value: *Variant*, new_value: *Variant*)

Emitted when a config value of the app has been updated. *key* refers to one of the key constants defined in *ConfigData*, with *old_value* and *new_value* representing the old and new values of that config.

theme_config_ready ()

Emitted when RetroHub has finished loading theme configs. This signal is emitted after theme initialization, and configurations can be retrieved with *get_theme_config*.

theme_config_updated (key: *String*, old_value: *Variant*, new_value: *Variant*)

Emitted when a config value of the theme has been updated. *key* is the same value used on *get_theme_config/set_theme_config*, and therefore defined by your theme. *old_value* and *new_value* represent the old and new values of that config.

game_data_updated (game_data: *RetroHubGameData*)

Emitted when a game data has been updated by the user. The reference doesn't change from the earlier data received from *game_received*, so you can use it to test whether you need to update displayed information in your theme.

```
connect("game_data_updated", self, "_on_game_data_updated")

func _on_game_data_updated(game_data):
    if self.game_data == game_data:
        # Update displayed information with new info
        ...
```

Methods

Type	Name
<i>Variant</i>	<i>get_theme_config</i>
<i>void</i>	<i>set_theme_config</i>
<i>void</i>	<i>save_theme_config</i>

Variant **get_theme_config** (key: *String*, default_value: *Variant*)

Get a theme config under the given `key`. If the key doesn't exist, `default_value` is returned.

Warning: If the key doesn't exist, `default_value` is not implicitly set in the configuration.

void **set_theme_config** (key: `String`, value: `Variant`)

Sets a theme config under the given `key` to `value`. If the key doesn't exist, it is created.

Warning: RetroHub only emits the `theme_config_updated` signal when the settings are saved with `save_theme_config`.

void **save_theme_config** ()

Saves the theme config to disk. For every `key` that has been changed, the `theme_config_updated` signal is emitted.

Note: If you're using a dedicated theme config scene, RetroHub will automatically call this function by you when the theme is unloaded, and therefore you only need to use `set_theme_config`.

4.2.4 RetroHubUI

This class exposes some UI actions and data for themes.

Constants

Type	Name
Color	<code>color_theme_accent</code>
Color	<code>color_success</code>
Color	<code>color_warning</code>
Color	<code>color_error</code>
Color	<code>color_pending</code>
Color	<code>color_unavailable</code>
int	<code>max_popupmenu_height</code>

Color **color_theme_accent**

Color used for “accent” elements on RetroHub’s interface.

Color **color_success**

Color used for success messages.

Color **color_warning**

Color used for warning messages.

Color `color_error`

Color used for error messages.

Color `color_pending`

Color used for pending operations.

Color `color_unavailable`

Color used for unavailable objects.

`int max_popupmenu_height`

Maximum size, in pixels, for `PopupMenu` elements. Use this value in your themes to have consistent sizes with RetroHub's defaults.

Enumerations

enum **AudioKeys**:

- **ACTIVATED** - Button activated.
 - **CHECK_BUTTON_OFF** - Check button toggled off.
 - **CHECK_BUTTON_ON** - Check button toggled on.
 - **KEYBOARD_TYPE** - Virtual keyboard inputs.
 - **MENU_ENTER** - Menu opened.
 - **MENU_IN** - Popup opened.
 - **MENU_OUT** - Popup/menu closed.
 - **NAVIGATION** - UI navigation.
 - **SLIDE** - Sliding tabs.
 - **SLIDER_TICK** - Value change ticks.
-

enum **Icons**:

- **DOWNLOADING** - Downloading icon.
- **ERROR** - Error icon.
- **FAILURE** - Failure icon.
- **IMAGE_DOWNLOADING** - Image downloading icon.
- **LOAD** - Load file/directory icon.
- **LOADING** - Loading icon.
- **SUCCESS** - Success icon.

- **VISIBILITY_HIDDEN** - Hidden visibility icon.
 - **VISIBILITY_VISIBLE** - Visible visibility icon.
 - **WARNING** - Warning icon.
-

enum **ConfigTabs**:

- **QUIT** - Quit tab.
- **GAME** - Game metadata editor tab.
- **SCRAPER** - Scraper tab.
- **THEME** - Theme settings tab.
- **SETTINGS_GENERAL** - General settings tab.
- **SETTINGS_INPUT** - Input settings tab.
- **SETTINGS_REGION** - Region settings tab.
- **SETTINGS_SYSTEMS** - Systems settings tab.
- **SETTINGS_EMULATORS** - Emulators settings tab.
- **SETTINGS_ABOUT** - About tab.

Signals

path_selected (file: [String](#))

Emitted when a path is selected in the file/directory dialog. Used for [request_file_load/request_folder_load](#). `file` is the path selected, or "" if nothing was selected.

Methods

Type	Name
void	filesystem_filters
void	request_file_load
void	request_folder_load
Texture	load_app_icon
void	show_virtual_keyboard
bool	is_virtual_keyboard_visible
bool	is_event_from_virtual_keyboard
void	hide_virtual_keyboard
void	open_app_config
void	show_warning
Window	get_focused_window
Control	get_true_focused_control
void	play_sound

void **filesystem_filters** (filters: [Array](#))

Set file filters for the file/directory dialog. Works exactly the same as Godot's [FileDialog.filters](#).

```
# Only show supported image files
filesystem_filters(["*.png, *.jpg, *.jpeg ; Supported Images"])]
```

void **request_file_load** (base_path: String)

Request a file load. The file dialog will be opened and the user will be able to select a file. `base_path` indicates what folder path to start on initially. Call `filesystem_filters` before to set desired file filters.

When a file is selected, the `path_selected` signal will be emitted with the path to the file. If the user cancels the dialog, the signal will be emitted with an empty string.

```
# Example to load a single PNG file
RetroHubUI.filesystem_filters(["*.png ; PNG Images"])
RetroHubUI.request_file_load(FileUtils.get_home_dir())
var path : String = yield(RetroHubUI, "path_selected")
    if not path.empty():
        print("File selected: " + path)
    else:
        print("No file selected")
```

void **request_folder_load** (base_path: String)

Request a folder load. The file dialog will be opened and the user will be able to select a folder. `base_path` indicates what folder path to start on initially.

When a folder is selected the `path_selected` signal will be emitted with the path to the folder. If the user cancels the dialog, the signal will be emitted with an empty string.

```
# Example to load a folder
RetroHubUI.request_folder_load(FileUtils.get_home_dir())
var path : String = yield(RetroHubUI, "path_selected")
    if not path.empty():
        print("Folder selected: " + path)
    else:
        print("No folder selected")
```

Texture **load_app_icon** (icon: Icons)

Load an app-default icon.

void **show_virtual_keyboard** ()

Asks RetroHub to show the virtual keyboard. The node you want to receive key events must be focused (`Control.grab_focus()`) before calling this method. The virtual keyboard will then send raw `InputEventKey` events to the focused node.

Note: RetroHub automatically opens the virtual keyboard for `LineEdit` and `TextEdit` nodes. This method is only needed if you want to show the virtual keyboard for other nodes.

bool `is_virtual_keyboard_visible ()`

Returns `true` if the virtual keyboard is currently visible.

bool `is_event_from_virtual_keyboard ()`

Returns `true` if the current input event is coming from the virtual keyboard.

```
func _input(event):  
    if RetroHubUI.is_event_from_virtual_keyboard():  
        # From virtual keyboard  
        ...  
    else:  
        # From actual user input  
        ...
```

void `hide_virtual_keyboard ()`

Hides the virtual keyboard.

void `open_app_config (tab : ConfigTabs = -1)`

Opens RetroHub's app configuration UI. If `-1` is specified, the app will stay on the currently selected tab.

Warning: Opening the app configuration UI will steal focus from the theme until the user closes it.
--

void `show_warning (text: String)`

Shows a warning to the user. Use this for warnings that are at the application level, and not specific to your theme.

Window `get_focused_window ()`

Returns the currently focused **Window** across the entire application.

Control `get_true_focused_control ()`

Returns the currently focused **Control** by also taking into account the currently focused **Window**.

void `play_sound (key: AudioKeys, override: bool = true)`

Plays a UI sound from one of the existing sound keys. If `override` is `true`, the sound will replace any current sound; otherwise, it may be played in parallel to other sounds.

4.3 Utils

4.3.1 RegionUtils

This is a helper class to handle challenges with presenting information in the user's region.

Methods

Type	Name
String	<i>localize_date</i>
String	<i>globalize_date_str</i>
String	<i>globalize_date_dict</i>
Control	<i>localize_age_rating</i>
int	<i>localize_age_rating_idx</i>
String	<i>localize_system_name</i>

String **localize_date** (date_raw: String)

Localizes a raw date string which is in ISO8601 format.

```
# User prefers DD/MM/YYYY
RegionUtils.localize_date("19870102T152741") # Returns "02/01/1987 15:27:41"
```

String **globalize_date_str** (date_raw: String, source_format: String = "")

Globalizes a previously localized date with *localize_date* back to a global ISO8601 format. *source_format* can be specified to tell in which format the string is (one of *date_format*), otherwise leave it empty to use the current user's region.

```
# User prefers DD/MM/YYYY
RegionUtils.globalize_date_str("02/01/1987 15:27:41") # Returns "19870102T152741"
```

String **globalize_date_dict** (date_dict: Dictionary, source_format: String = "")

Globalizes a date dictionary coming from *Godot* back to a global ISO8601 format. *source_format* can be specified to tell in which format the string is (one of *date_format*), otherwise leave it empty to use the current user's region.

```
# User prefers DD/MM/YYYY
RegionUtils.globalize_date_dict({
    "year": 1987, "month": 1,
    "day": 2, "hour": 15,
    "minute": 27, "second": 41
}) # Returns "19870102T152741"
```

Control **localize_age_rating** (age_rating_raw: String)

Localizes an age rating string to the user's preferred age rating system. This returns a `Control` depicting that information which you should add in your theme's scene tree.

```
# User prefers PEGI
RegionUtils.localize_age_rating("2/3/4") # Returns a Control with the PEGI 12 rating icon
```

`int localize_age_rating_idx ()`

Returns the internal index for age rating strings (e.g. "0/3/2"). This is useful if you want to use the age rating string in your own code.

```
# User prefers PEGI
RegionUtils.localize_age_rating_idx() # Returns idx 1
```

`String localize_system_name (system_name: String)`

Localizes a short system name to the user's preferred system name.

```
# User prefers "Sega Megadrive"
RegionUtils.localize_system_name("genesis") # Returns "megadrive"
```

4.3.2 FileUtils

This is a helper class to handle file operations in a platform agnostic way.

Enumerations

enum **OS_ID**:

- **WINDOWS** - Windows or UWP
- **MACOS** - macOS
- **LINUX** - Unix systems (Linux, BSD, etc.)
- **UNSUPPORTED** - Any other system currently not supported

Methods

Type	Name
String	<i>test_for_valid_path</i>
void	<i>ensure_path</i>
String	<i>expand_path</i>
int	<i>get_space_left</i>
String	<i>get_folder_size</i>
int	<i>get_file_count</i>
int	<i>get_home_dir</i>
<i>OS_ID</i>	<i>get_os_id</i>
String	<i>get_os_string</i>
bool	<i>is_steam_deck</i>

String test_for_valid_path (paths: String | Array)

Tests whether the supplied path(s) exist in the user's system. If **paths** is a **String**, tests if it exists. Otherwise, if it is a **Array**, tests for all strings inside and returns the first valid one. Returns "" if no valid path is found.

void ensure_path (path: String)

Recursively creates the directory structure of the specified path, ensuring all subfolders exist

```
# Creates the directory structure "very/long/folder" if necessary
FileUtils.ensure_path("very/long/folder/my_file.txt")
var file = File.open("very/long/folder/my_file.txt", File.WRITE)
...
```

String expand_path (path: String)

Expand paths with embedded variables. Supported variables are:

- ~ - User's home directory

```
# Returns C:/Users/<username>/Documents on Windows
FileUtils.expand_path("~/Documents")
```

int get_space_left ()

Returns the amount of free space, in bytes, of the disk that contains RetroHub's config directory.

String get_folder_size (path: String, filter_folders: Array = [])

Returns the total size of the specified folder, in bytes. If **filter_folders** is specified, it will not count the size of any folder that matches any of the strings in the array.

int get_file_count (path: String, filter_folders: Array = [])

Returns the total number of files in the specified folder. If **filter_folders** is specified, it will not count any files inside a folder that matches any of the strings in the array.

String get_home_dir ()

Returns the user's home directory.

OS_ID get_os_id ()

Returns the current operating system as an **OS_ID**.

String get_os_string ()

Returns the current operating system as a string. Possible values are "windows", "macos", "linux", and "null".

bool is_steam_deck ()

Returns **true** if the current system is a Steam Deck.

4.3.3 JSONUtils

This is a helper class to facilitate usage of JSON files.

Methods

Type	Name
Dictionary / Array	<i>load_json_file</i>
Dictionary / Array	<i>load_json_buffer</i>
void	<i>save_json_file</i>
void	<i>make_system_specific</i>
Dictionary	<i>map_array_by_key</i>
String	<i>format_string_with_substitutes</i>

Dictionary / Array **load_json_file** (filepath: String)

Loads a JSON file and returns it as a Dictionary or as an Array. If the file does not exist or is invalid, an empty dictionary is returned.

Dictionary / Array **load_json_buffer** (data: String)

Loads a JSON file from data in memory, and returns it as a Dictionary or as an Array. If the data is invalid, an empty dictionary is returned.

void **save_json_file** (json: Dictionary, file_path: String)

Saves a dictionary as a JSON file to file_path.

void **make_system_specific** (json: Dictionary, curr_system: String)

Edits a dictionary in-place to remove variable *Paths*, taking into account the current system. Used internally to load configurations dependent on the current system.

```
var data = {
  "foo": "bar",
  "path": {
    "windows": [
      "C:\\foo\\bar",
      "C:\\foo\\bar2"
    ],
    "macos": "/Applications/foo/bar",
    "linux": "/bin/foo/bar"
  }
}

JSONUtils.make_system_specific(data, "windows")
# Returns {"foo": "bar", "path": ["C:\\foo\\bar", "C:\\foo\\bar2"]}
```

Dictionary **map_array_by_key** (input: Array, key: String)

Given an array of dictionaries with a given key, returns a dictionary mapped by the key's value.

```
var data = [
  {"id": "foo", "name": "Foo"},
  {"id": "bar", "name": "Bar"}
]

JSONUtils.map_array_by_key(data, "id")
# Returns {"foo": {"id": "foo", "name": "Foo"}, "bar": {"id": "bar", "name": "Bar"}}
```

String **format_string_with_substitutes** (raw_str: String, substitutes: Dictionary, tk_start: String = "{", tk_end: String = "}", remove_empty: bool = false)

Takes a string with embedded tokens and replaces them with values from a dictionary. Tokens are defined by `tk_start` and `tk_end`, and are `{}` by default. If `remove_empty` is `true`, tokens that are not found in the dictionary are removed from the string.

```
var raw_str = "Hello {name}! You are {age} years old."

JSONUtils.format_string_with_substitutes(raw_str, {"name": "John", "age": 30})
# Returns "Hello John! You are 30 years old."

JSONUtils.format_string_with_substitutes(raw_str, {"name": "John"})
# Returns "Hello John! You are {age} years old."

JSONUtils.format_string_with_substitutes(raw_str, {"name": "John"}, "{", "}", remove_
↪empty = true)
# Returns "Hello John! You are years old."
```

4.3.4 TTS

This is a helper class to handle text-to-speech functionality.

Virtual Methods

Note: These methods should be implemented in other scripts to control TTS output. When a node gets focused, the TTS system will traverse the nodes and it's parents recursively in search of one of these methods, and if existing, use it's text output instead.

Type	Name
String	<i>tts_text</i>
String	<i>tts_tree_item_text</i>
String	<i>tts_range_value_text</i>
String	<i>tts_popup_menu_item_text</i>

String **tts_text** (focused: **Control**)

Generic TTS text method. This is called when a node gets focused. The `focused` parameter is the node that got focused, and is a child of either the node where this script is, or of one of this children.

This method should return the text to be spoken, which overrides the default implementation. If it returns an empty string, the TTS system will continue traversing the parent methods, and use the default implementation if no further node overrides it.

```
func tts_text(focused):
    if focused is $MyLabel:
        # Override output in this particular case
        return "Hey, my label got focused!"
    # We're not interested in this node, so return an empty string
    return ""
```

String **tts_tree_item_text** (item: **TreeItem**, tree: **Tree**)

Specific TTS text for a focused cell on a **Tree** node.

This method should return the text to be spoken, which overrides the default implementation. If it returns an empty string, the TTS system will continue traversing the parent methods, and use the default implementation if no further node overrides it.

String **tts_range_value_text** (value: **float**, range: **Range**)

Specific TTS text for a focused **Range** node. It supplies the current node's value to customize how it is presented to the user.

This method should return the text to be spoken, which will be appended to the default implementation. If it returns an empty string, the TTS system will continue traversing the parent methods, and use the default implementation if no further node overrides it.

Warning: Always use the supplied value, not the range's value. The TTS system might use different values, such as minimum/maximum ranges.

```
func tts_range_value_text(value, range):
    if range is $FilesizeRange:
        # This range works with file sizes, so add a suffix
        return "%d megabytes" % value
    # We're not interested in this node, so return an empty string
    return ""
```

String **tts_popup_menu_item_text** (idx: **int**, menu: **PopupMenu**)

Specific TTS text for a focused **PopupMenu** node. It supplies the currently selected menu item.

This method should return the text to be spoken, which will be appended to the default implementation. If it returns an empty string, the TTS system will continue traversing the parent methods, and use the default implementation if no further node overrides it.

Methods

Type	Name
void	<i>speak</i>
void	<i>stop</i>
bool	<i>is_speaking</i>
String	<i>singular_or_plural</i>

void **speak** (text: `String`, interrupt: `bool` = `true`)

Speaks the given text. If `interrupt` is `true`, the current speech is interrupted, ttherwise, the new text is queued.

```
TTS.speak("Hello")
TTS.speak("World!", false)
# Will speak "Hello World!"
```

void **stop** ()

Stops any current speech immediately.

bool **is_speaking** ()

Returns whether the TTS system is currently speaking any sentence.

String **singular_or_plural** (count: `int`, singular: `String`, plural: `String`)

Returns the singular or plural form of a word depending on the given count. Useful if using variables and the final text needs to change due to pluralization.

```
var count = 5
TTS.speak("You have %d %s" % [
    count,
    TTS.singular_or_plural(count, "apple", "apples")
])
```

4.4 Integrations

4.4.1 RetroAchievements

This is the main class to interact with RetroAchievements.

Note: For a proper guide on how to use this integration, please refer to the *dedicated integration guide*.

Methods

Type	Name
bool	<i>is_available</i>
String	<i>get_cheevos_dir</i>
String	<i>get_cheevos_hash_cache_path</i>
Dictionary	<i>build_auth</i>
<i>RetroAchievements.GameInfo</i>	<i>get_game_info</i>
String	<i>get_game_hash</i>

static bool **is_available** ()

Indicates whether the RetroAchievements integration is enabled by the user. If this returns `false`, any usage of this API results in undefined behavior.

Warning: It is only safe to instantiate this class if this function returns `true`. Otherwise, the instance will remove itself and become `null`, which may cause crashes.

Note: This only refers to RetroHub’s configuration. The API itself may be unavailable for other reasons, such as the user not being logged in, or network issues.

static String **get_cheevos_dir** ()

Returns the path to the directory where RetroAchievements stores its data. Used internally by RetroHub.

static String **get_cheevos_hash_cache_path** ()

Returns the path to the file where RetroAchievements stores the known ROM hashes. Used internally by RetroHub.

Dictionary **build_auth** (bool reload_credentials = false)

Builds an authentication [Dictionary](#), needed for any raw API calls. If `reload_credentials` is `true`, the credentials will be re-fetched from the user’s configuration.

Warning: You will only need to `reload_credentials` if using the [RetroAchievements.Raw](#) endpoints; if you’re only using the high-level API calls on this page, this is not needed.

Additionally, only `reload_credentials` when you know the credentials are invalid (e.g. receiving 401 HTTP response codes), as every force reload will cause file I/O to load user credentials.

RetroAchievements.GameInfo **get_game_info** (*RetroHubGameData* data)

Returns a *RetroAchievements.GameInfo* object for the given game. This object contains all achievement information for themes to use.

Note: This function is asynchronous; use `await` to wait for it's completion.

`String get_game_hash (RetroHubGameData data)`

Returns the hash of the given game. This is used for identifying game files in RetroAchievements.

Note: RetroAchievements has it's own hashing algorithms dependant on different game files. Check their [documentation](#) for more information.

4.4.2 RetroAchievements.GameInfo

This class contains all information about a game's achievements.

Note: For a proper guide on how to use this integration, please refer to the [dedicated integration guide](#).

Enumerations

enum **Error**:

- **OK** - Success / no error.
- **ERR_INVALID_CRED** - The user's credentials are invalid.
- **ERR_CONSOLE_NOT_SUPPORTED** - The requested game's system is not supported by RetroAchievements.
- **ERR_GAME_NOT_FOUND** - The requested game hash was not found on RetroAchievements.
- **ERR_NETWORK** - A network error occurred.
- **ERR_INTERNAL** - An internal error occurred.

Properties

Type	Name	Default
<i>GameInfo.Error</i>	<i>err</i>	OK
int	<i>id</i>	0
Array <i>RetroAchievements.Achievement</i>	[<i>achievements</i>]	[]
int	<i>player_count</i>	0

GameInfo.Error **err** = OK

The error code of the function call that generated/returned this game info. If not OK, the remaining properties will have no meaning, and this object should be discarded.

int id = 0

The game's ID on RetroAchievements.

Array [*RetroAchievements.Achievement*] **achievements** = []

An *Array* of *RetroAchievements.Achievement* objects, containing all the achievements available for this game.

int player_count = 0

The total number of players that have played this game on RetroAchievements.

Methods

Type	Name
void	<i>parse_raw</i>

void **parse_raw** (*Dictionary* data)

Parses a raw JSON response from RetroAchievements's API into this object. Used internally by RetroHub.

4.4.3 RetroAchievements.Achievement

This class contains information of a specific achievement from a *RetroAchievements.GameInfo* object.

Note: For a proper guide on how to use this integration, please refer to the *dedicated integration guide*.

Enumerations

enum **Type**:

- **NORMAL** - A normal achievement.
- **PROGRESSION** - An achievement that denotes some in-game progression.
- **WIN** - An achievement obtainable by reaching (one of) the game's ending.
- **MISSABLE** - An achievement that can be missed during a playthrough, requiring the player to start a new game to obtain it.

Properties

Type	Name	Default
int	<i>id</i>	0
String	<i>title</i>	""
String	<i>description</i>	""
<i>Achievement.Type</i>	<i>type</i>	NORMAL
bool	<i>unlocked</i>	false
bool	<i>unlocked_hard_mode</i>	false
int	<i>unlocked_count</i>	0
int	<i>unlocked_hard_mode_count</i>	0

`int id = 0`

The achievement's ID on RetroAchievements.

`String title = ""`

The achievement's title.

`String description = ""`

The achievement's description.

`Achievement.Type type = NORMAL`

The achievement's type.

`bool unlocked = false`

Whether the achievement has been unlocked.

`bool unlocked_hard_mode = false`

Whether the achievement has been unlocked, in hard mode.

`int unlocked_count = 0`

How many players unlocked this achievement.

`int unlocked_hard_mode_count = 0`

How many players unlocked this achievement, in hard mode.

Methods

Type	Name
Texture	<i>load_icon</i>
void	<i>parse_raw</i>

Texture `load_icon ()`

Loads the achievement's icon. If the icon doesn't exist yet, it will download it from RetroAchievements' servers and store it automatically.

Warning: This function will queue downloads, which cannot be canceled. A large queue will create delays on other API calls that require any network access. Therefore, you should only call this function only when immediately needed, and take care to not call it for a lot of achievements at once.

Note: This function is asynchronous; use an `await` to wait for it's completion.

void `parse_raw (Dictionary data)`

Parses a raw JSON response from RetroAchievements's API into this object. Used internally by RetroHub.

4.4.4 RetroAchievements.Raw

This class exposes the RetroAchievements' raw API for any advanced use cases.

Note: This API wraps the [RetroAchievements' official API endpoints](#). You should refer to it for details on each API call and arguments.

Note: Almost all requests require you to supply an authentication [Dictionary](#), which can be obtained through [RetroAchievements.build_auth](#).

Methods

Type	Name
<i>Raw.Response</i>	<i>get_achievement_of_the_week</i>
<i>Raw.Response</i>	<i>get_claims</i>
<i>Raw.Response</i>	<i>get_active_claims</i>
<i>Raw.Response</i>	<i>get_top_ten_users</i>
<i>Raw.Response</i>	<i>get_user_recent_achievements</i>
<i>Raw.Response</i>	<i>get_achievements_earned_between</i>
<i>Raw.Response</i>	<i>get_achievements_earned_on_day</i>
<i>Raw.Response</i>	<i>get_game_info_and_user_progress</i>
<i>Raw.Response</i>	<i>get_user_awards</i>
<i>Raw.Response</i>	<i>get_user_claims</i>
<i>Raw.Response</i>	<i>get_user_completed_games</i>
<i>Raw.Response</i>	<i>get_user_game_rank_and_score</i>
<i>Raw.Response</i>	<i>get_user_points</i>
<i>Raw.Response</i>	<i>get_user_progress</i>
<i>Raw.Response</i>	<i>get_user_recently_played_games</i>
<i>Raw.Response</i>	<i>get_user_summary</i>
<i>Raw.Response</i>	<i>get_achievement_count</i>
<i>Raw.Response</i>	<i>get_achievement_distribution</i>
<i>Raw.Response</i>	<i>get_game</i>
<i>Raw.Response</i>	<i>get_game_extended</i>
<i>Raw.Response</i>	<i>get_game_rank_and_score</i>
<i>Raw.Response</i>	<i>get_console_ids</i>
<i>Raw.Response</i>	<i>get_game_list</i>
<i>Raw.Response</i>	<i>get_achievement_unlocks</i>

Raw.Response **get_achievement_of_the_week** (Dictionary auth)

API_GetAchievementOfTheWeek.php

Raw.Response **get_claims** (Dictionary auth, String kind)

API_GetClaims.php

Raw.Response **get_active_claims** (Dictionary auth)

API_GetActiveClaims.php

Raw.Response **get_top_ten_users** (Dictionary auth)

API_GetTopTenUsers.php

Raw.Response **get_user_recent_achievements** (Dictionary auth, String username, int recent_minutes = 60)

API_GetUserRecentAchievements.php

Raw.Response **get_achievements_earned_between** (Dictionary auth, String username, int from_date, int to_date)

API_GetAchievementsEarnedBetween.php

Raw.Response **get_achievements_earned_on_day** (Dictionary auth, String username, String on_date)

API_GetAchievementsEarnedOnDay.php

Raw.Response **get_game_info_and_user_progress** (Dictionary auth, String username, int game_id)

API_GetGameInfoAndUserProgress.php

Raw.Response **get_user_awards** (Dictionary auth, String username)

API_GetUserAwards.php

Raw.Response **get_user_claims** (Dictionary auth, String username)

API_GetUserClaims.php

Raw.Response **get_user_completed_games** (Dictionary auth, String username)

API_GetUserCompletedGames.php

Raw.Response **get_user_game_rank_and_score** (Dictionary auth, String username, int game_id)

API_GetUserGameRankAndScore.php

Raw.Response **get_user_points** (Dictionary auth, String username)

API_GetUserPoints.php

Raw.Response **get_user_progress** (Dictionary auth, String username, Array [int] game_ids)

API_GetUserProgress.php

Raw.Response **get_user_recently_played_games** (Dictionary auth, String username, int count = 10, int offset = 0)

API_GetUserRecentlyPlayedGames.php

Raw.Response **get_user_summary** (Dictionary auth, String username, int recent_games_count = 0, int recent_achievements_count = 5)

API_GetUserSummary.php

Raw.Response **get_achievement_count** (Dictionary auth, int game_id)

API_GetAchievementCount.php

Raw.Response **get_achievement_distribution** (Dictionary auth, int game_id)

API_GetAchievementDistribution.php

Raw.Response **get_game** (Dictionary auth, int game_id)

API_GetGame.php

Raw.Response **get_game_extended** (Dictionary auth, int game_id)

API_GetGameExtended.php

Raw.Response **get_game_rank_and_score** (Dictionary auth, int game_id, String type)

API_GetGameRankAndScore.php

Raw.Response **get_console_ids** (Dictionary auth)

API_GetConsoleIDs.php

Raw.Response **get_game_list** (Dictionary auth, int console_id, bool should_only_retrieve_games_with_achievements = false, bool should_retrieve_game_hashes = false)

API_GetGameList.php

Raw.Response **get_achievement_unlocks** (Dictionary auth, int achievement_id, int count = 50, int offset = 0)

API_GetAchievementUnlocks.php

4.4.5 RetroAchievements.Raw.Response

Class holding API responses from the *RetroAchievements.Raw* class.

Properties

Type	Name
int	<i>godot_error</i>
int	<i>response_code</i>
Dictionary / Array	<i>body</i>

int godot_error

Godot error code. If not OK, there was some internal error when making the HTTP request.

int response_code

HTML response code. If not 200, the request failed on the server side.

Dictionary / Array body

The response JSON body, already parsed into a Dictionary or Array.