Package: ramiga (via r-universe)

June 1, 2026

Title Emulating the Amiga Using the vAmiga Engine

Version 0.0.0.9007

Description This is an Amiga emulator based on the vAmiga project <https://github.com/dirkwhoffmann/vAmiga>. It provides low level access to the emulator and allows users to interact with the emulated machine through the console or scripts.

License GPL (>= 3)

Depends R (>= 4.4)

Imports R6

SystemRequirements C++20 (GCC >= 11, Clang >= 15), zlib

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

LinkingTo cpp11 (>= 0.4.2)

URL https://pepijn-devries.github.io/ramiga/, https://github.com/pepijn-devries/ramiga/

Collate 'component.R' 'documentation.R' 'controldevice.R' 'controlport.R' 'cpp11.R' 'cpu.R' 'emulator.R' 'filesystem.R' 'floppydrive.R' 'image.R' 'memory.R' 'output.R' 'ramiga-package.R' 'wav.R'

Config/pak/sysreqs zlib1g-dev

Repository https://pepijn-devries.r-universe.dev

Date/Publication 2026-06-01 22:00:31 UTC

RemoteUrl https://github.com/pepijn-devries/ramiga

RemoteRef main

RemoteSha 0a544589ae0ad07b5951b14b4549d58bb4496912

Contents

`RamigaComponent` *The RamigaComponent R6 Class*

Description

An abstract class to represent emulator components. Each of the emulator components inherit from this class. It provides consistent access to and control over component specific options.

Active bindings

`options` A named `list` of options. You can get and set all individual elements. Elements are always returned as `character`, but can be also be set as `numeric` or `logical`, depending on the specific option. Assignment values are always parsed by the vAmiga engine.

Methods

Public methods:

- `RamigaComponent$new()`
- `RamigaComponent$get_pointer()`
- `RamigaComponent$get_emulator()`
- `RamigaComponent$list_options()`
- `RamigaComponent$clone()`

Method `new()`**:** Most emulator components don't need to be initialised. It's simplest to create an emulator with `RamigaEmulator$new()`. All components can be accessed through the R6 object fields.

Usage:

```
RamigaComponent$new(emulator, component_id, ...)
```

Arguments:

`emulator` All emulator components should be part of an emulator. All components (except for the emulator itself) therefore needs the, initialised with an emulator.

component_id Integer value used to identify different types of components.

... Ignored

Method `get_pointer()`**:** Get the pointer to the emulator object in memory

Usage:

```
RamigaComponent$get_pointer(...)
```

Arguments:

... Ignored

Returns: Returns the `externalptr` pointing to the C++ class instance in memory

Method `get_emulator()`**:** Get associated emulator.

Usage:

```
RamigaComponent$get_emulator(...)
```

Arguments:

... Ignored

Returns: Returns a RamigaEmulator class object instance in memory

Method `list_options()`**:** List available options for the emulator component. It shows the name of the option and indicates their intended value(s).

Usage:

```
RamigaComponent$list_options(...)
```

Arguments:

... Ignored

Returns: Returns a named `list` with available options.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaComponent$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

`RamigaControlDevice` *The RamigaControlDevice R6 Class*

Description

A class that represents control devices, like the mouse or joystick.

Super class

`ramiga::RamigaComponent` -> `RamigaControlDevice`

Methods

Public methods:

- RamigaControlDevice$new()
- RamigaControlDevice$print()
- RamigaControlDevice$clone()

Method `new()`**:**

Usage:

```
RamigaControlDevice$new(emulator, port_number, type)
```

Arguments:

`emulator` A virtual control device only is meaningful when connected to a virtual device. Therefore, you need an emulator (RamigaEmulator) to connect the device to.

`port_number` Port on the virtual machine to which the device is connected. Should be either port 1 or 2.

`type` Type of device, should be either `"mouse"`, or `"joystick"`.

Method `print()`**:** Prints some basic information about the control device

Usage:

```
RamigaControlDevice$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaControlDevice$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RamigaControlPort` *The RamigaControlPort R6 Class*

Description

A class that represents the emulator's control ports (for mouse and joystick).

Super class

`ramiga::RamigaComponent` -> `RamigaControlPort`

Active bindings

`device` This field tells you which device (mouse, joystick or none) is connected to the current port. Should be a RamigaControlDevice class object, or `NULL` if there is no device.

Methods

Public methods:

- RamigaControlPort$new()
- RamigaControlPort$print()
- RamigaControlPort$clone()

Method `new()`**:**

Usage:

```
RamigaControlPort$new(emulator, port_number = 1L, device = NULL)
```

Arguments:

`emulator` An `RamigaEmulator` class object. The control port to be created needs to be associated with a virtual machine. The initiated object will represent the control port of the emulated machine.

`port_number` The Amiga has two control ports. This number specifies which one we are using. Should be either 1 or 2.

`device` Device connected to this port. An RamigaControlDevice class object or `NULL` if there is no device.

Method `print()`**:** Print some basic information about the control port.

Usage:

```
RamigaControlPort$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaControlPort$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RamigaCPU` *The RamigaCPU R6 Class*

Description

A class that represents the emulator's Central Processing Unit (CPU).

Super class

`ramiga::RamigaComponent` -> `RamigaCPU`

Methods

Public methods:

- RamigaCPU$get_cycles()
- RamigaCPU$get_program_counter()
- RamigaCPU$get_config()
- RamigaCPU$print()
- RamigaCPU$clone()

Method `get_cycles()`**:** Get the CPU cycle count since the emulator started

Usage:

```
RamigaCPU$get_cycles()
```

Returns: Returns the count in number of cycles

Method `get_program_counter()`**:** Get the memory address of the currently running program

Usage:

```
RamigaCPU$get_program_counter()
```

Returns: On the Amiga memory addresses are mapped as 32 bit unsigned integers. As R doesn't support this type, it is returned as a `numeric` (double) value.

Method `get_config()`**:** Get current CPU configuration details

Usage:

```
RamigaCPU$get_config()
```

Returns: Returns a named list

Method `print()`**:** Print some basic information about the CPU

Usage:

```
RamigaCPU$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaCPU$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RamigaEmulator *The RamigaEmulator R6 Class*

Description

A class representing the emulator for emulating an Amiga.

Super class

`ramiga::RamigaComponent` -> `RamigaEmulator`

Active bindings

`memory` The emulated memory, represented by RamigaMemory

`cpu` The emulated CPU, represented by RamigaCPU

`floppy_drives` The emulated floppy drives, represented by a named `list` of RamigaFloppyDrive objects.

`control_ports` Control ports to which a virtual mouse or joystick can be connected. It is represented by the RamigaControlPort class.

`output` Get object to capture emulator output. Handled by RamigaOutput.

Methods

Public methods:

- `RamigaEmulator$new()`
- `RamigaEmulator$power_on()`
- `RamigaEmulator$power_off()`
- `RamigaEmulator$soft_reset()`
- `RamigaEmulator$hard_reset()`
- `RamigaEmulator$run()`
- `RamigaEmulator$next_frame()`
- `RamigaEmulator$fast_forward()`
- `RamigaEmulator$get_info()`
- `RamigaEmulator$get_config()`
- `RamigaEmulator$set_config()`
- `RamigaEmulator$get_version()`
- `RamigaEmulator$print()`
- `RamigaEmulator$clone()`

Method `new()`**:** Initialise a new Amiga emulator

Usage:

```
RamigaEmulator$new()
```

Method `power_on()`**:** When the emulator is initialised it is powered off. Call this to power on the virtual machine.

Usage:

```
RamigaEmulator$power_on()
```

Returns: Returns the emulator object

Method `power_off()`**:** Power off emulator. Advisable if you plan to make adjustments to the emulated hardware configuration.

Usage:

```
RamigaEmulator$power_off()
```

Returns: Returns the emulator object

Method `soft_reset()`**:** A soft reset, just resets the CPU, causing the system to reboot. This is similar to simultaneously pressing <Ctrl> and both <Amiga> keys on an original machine.

Usage:

```
RamigaEmulator$soft_reset()
```

Returns: Returns the emulator object

Method `hard_reset()`**:** A hard reset. Same as powering of the machine, then powering it back on.

Usage:

```
RamigaEmulator$hard_reset()
```

Returns: Returns the emulator object

Method `run()`**:** When initialised, the emulator is paused. Call this to start running the virtual machine. It will hold up the R thread untill you interrupt (by pressing). After the interrupt the machine will be paused.

Usage:

```
RamigaEmulator$run()
```

Returns: Returns NULL invisibly.

Method `next_frame()`**:** Update the machines state to the next video frame. The machine runs until the vertical blank is reached and pauses immediately.

Usage:

```
RamigaEmulator$next_frame()
```

Returns: Returns the emulator object

Method `fast_forward()`**:** Runs the emulator at warp speed and skips the specified number of frames

Usage:

```
RamigaEmulator$fast_forward(frames)
```

Returns: Returns the emulator object

Method `get_info()`**:** Get information about the emulator state.

Usage:

```
RamigaEmulator$get_info()
```

Returns: Returns a named list.

Method `get_config()`**:** Get information about the emulator configuration.

Usage:

```
RamigaEmulator$get_config()
```

Returns: Returns a named list.

Method `set_config()`**:** Quickly set the emulator configuration to one of the pre-specified schemes.

Usage:

```
RamigaEmulator$set_config(scheme = "A500_OCS_1MB", ...)
```

Arguments:

`scheme` Should be one of the following strings: `"A1000_OCS_1MB"`, `"A500_OCS_1MB"` (default), `"A500_ECS_1MB"`, or `"A500_PLUS_1MB"`.

`...` Ignored

Returns: Returns a string

Method `get_version()`**:** Get the version number of the vAmiga core used by this package

Usage:

```
RamigaEmulator$get_version(...)
```

Arguments:

`...` Ignored

Returns: Returns a string

Method `print()`**:** Prints basic information about the emulator.

Usage:

```
RamigaEmulator$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaEmulator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RamigaFileSystem *The RamigaFileSystem R6 Class*

Description

Obtain the file system from a virtual device (if available), such that you can interact with it.

Methods

Public methods:

- RamigaFileSystem$new()
- RamigaFileSystem$get_volume_name()
- RamigaFileSystem$get_traits()
- RamigaFileSystem$print()
- RamigaFileSystem$clone()

Method `new()`**:** Attempt to access the file system on a device image if available

Usage:

```
RamigaFileSystem$new(image)
```

Arguments:

`image` The virtual device (represented by a RamigaImage class object), from which the file system should be retrieved

Method `get_volume_name()`**:** Get the name of the volume of that contains the file system

Usage:

```
RamigaFileSystem$get_volume_name()
```

Returns: Returns the name of the volume

Method `get_traits()`**:** Get some traits of the file system.

Usage:

```
RamigaFileSystem$get_traits(...)
```

Arguments:

`...` Ignored

Returns: Returns a named `list` of file system traits.

Method `print()`**:** Prints basic info about the file system

Usage:

```
RamigaFileSystem$print(...)
```

Arguments:

`...` Ignored.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaFileSystem$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RamigaFloppyDrive` *The RamigaFlopyDrive R6 Class*

Description

A class that represents the emulator's floppy drive.

Super class

`ramiga::RamigaComponent` -> `RamigaFloppyDrive`

Methods

Public methods:

- `RamigaFloppyDrive$new()`
- `RamigaFloppyDrive$insert_disk()`
- `RamigaFloppyDrive$eject_disk()`
- `RamigaFloppyDrive$get_info()`
- `RamigaFloppyDrive$print()`
- `RamigaFloppyDrive$clone()`

Method `new()`:

Usage:

```
RamigaFloppyDrive$new(emulator, drive_number = 0)
```

Arguments:

`emulator` An `RamigaEmulator` class object. The floppy drive to be created needs to be associated with a virtual machine. The initiated object will represent the floppy drive of the emulated machine.

`drive_number` The emulator has 4 floppy drives, numbered 0 to 3. Pick which drive you want to operate

Method `insert_disk()`: Insert a virtual floppy disk in the virtual floppy drive.

Usage:

```
RamigaFloppyDrive$insert_disk(disk, write_protected = TRUE)
```

Arguments:

`disk` A floppy disk represented by a RamigaImage class object.

`write_protected` `logical` value indicating whether the inserted disk needs to be write protected. Default is `TRUE`

Method `eject_disk()`**:** Eject a virtual floppy disk from the virtual drive

Usage:

```
RamigaFloppyDrive$eject_disk(delay = 0)
```

Arguments:

`delay` Delay for ejecting the disk counted in CPU cycles.

Method `get_info()`**:** Get information about the floppy drive

Usage:

```
RamigaFloppyDrive$get_info()
```

Returns: Returns a named list with information

Method `print()`**:** Prints some basic information about the floppy drive

Usage:

```
RamigaFloppyDrive$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaFloppyDrive$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RamigaImage` *The RamigaImage R6 Class*

Description

A virtual device represented by an image.

Methods

Public methods:

- `RamigaImage$new()`
- `RamigaImage$info()`
- `RamigaImage$size()`
- `RamigaImage$get_pointer()`
- `RamigaImage$is_formatted()`
- `RamigaImage$get_file_system()`
- `RamigaImage$print()`
- `RamigaImage$clone()`

Method `new()`**:** Initialise an image from a disk image file.

Usage:

```
RamigaImage$new(path)
```

Arguments:

`path` File pointing to disk image file.

Method `info()`**:** Get information about the image.

Usage:

```
RamigaImage$info()
```

Returns: Returns a named `list` with information

Method `size()`**:** Get the size of the device represented by the image.

Usage:

```
RamigaImage$size()
```

Returns: Returns the size of the image in bytes.

Method `get_pointer()`**:** Get the pointer to the image object in memory

Usage:

```
RamigaImage$get_pointer(...)
```

Arguments:

`...` Ignored.

Returns: Returns the `externalptr` pointing to the C++ class instance in memory.

Method `is_formatted()`**:** Determines if there is a file system present.

Usage:

```
RamigaImage$is_formatted()
```

Returns: Returns `logical` indicating if disk contains a file system.

Method `get_file_system()`**:** Get a file system from a virtual device if present

Usage:

```
RamigaImage$get_file_system(...)
```

Arguments:

`...` Ignored.

Returns: Returns a RamigaFileSystem class object

Method `print()`**:** Print some basic information about the device

Usage:

```
RamigaImage$print(...)
```

Arguments:

`...` Ignored.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaImage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

RamigaMemory *The RamigaMemory R6 Class*

Description

The RamigaMemory class represents the memory of the emulated machine. Use this object to interact with the machines memory.

Super class

ramiga::RamigaComponent -> RamigaEmulator

Methods

Public methods:

- RamigaMemory$load_rom()
- RamigaMemory$load_rom_extension()
- RamigaMemory$has_rom()
- RamigaMemory$get_rom_traits()
- RamigaMemory$get_config()
- RamigaMemory$print()
- RamigaMemory$clone()

Method `load_rom()`**:** Load an operating system to ROM. It can be an official Kickstart ROM image, or the rom provided for AROS Research Operating System.

Usage:

```
RamigaMemory$load_rom(path)
```

Arguments:

`path` File path pointing to the location of the ROM file.

Returns: Returns the updated RamigaMemory object.

Method `load_rom_extension()`**:** Load an extension to a ROM file.

Usage:

```
RamigaMemory$load_rom_extension(path)
```

Arguments:

`path` File path pointing to the location of the ROM extension file.

Returns: Returns the updated RamigaMemory object.

Method `has_rom()`**:** Check if a ROM image was successfully loaded

Usage:

```
RamigaMemory$has_rom()
```

Returns: Returns a `logical` value.

Method `get_rom_traits()`**:** Get ROM characteristics if known.

Usage:

```
RamigaMemory$get_rom_traits()
```

Returns: Returns `NULL` if no ROM has been loaded. It will return A named list with ROM traits otherwise.

Method `get_config()`**:** Get details on the configuration of the emulated memory

Usage:

```
RamigaMemory$get_config()
```

Returns: Returns a named `list` with memory configurations.

Method `print()`**:** Print some basic information about the emulated memory.

Usage:

```
RamigaMemory$print(...)
```

Arguments:

`...` Ignored.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaMemory$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`RamigaOutput` *The RamigaOutput R6 Class*

Description

A class to capture output from the Amiga emulator (RamigaEmulator class).

Super class

`ramiga::RamigaComponent` -> `RamigaOutput`

Methods

Public methods:

- `RamigaOutput$capture_video_frame()`
- `RamigaOutput$capture_audio_buffer()`
- `RamigaOutput$print()`
- `RamigaOutput$clone()`

Method `capture_video_frame()`**:** Captures the current pixel information as sent to the monitor of the virtual device.

Usage:

```
RamigaOutput$capture_video_frame(file, ...)
```

Arguments:

`file` A path to a png file where to store the captured frame.

`...` Ignored

Returns: If file is missing it returns a `grDevices::as.raster()` object. Otherwise, it will save it as png file to the specified path and returns nothing.

Method `capture_audio_buffer()`**:** Capture the audio that is currently on the output buffer. While the emulator is running, audio is continuously generated and stored in a ring buffer. Use this function to collect data in this buffer. It can only be read once and the buffer has a limited capacity.

Usage:

```
RamigaOutput$capture_audio_buffer(file, ...)
```

Arguments:

`file` A file path to store the audio (RIFF wav format).

`...` Ignored

Returns: If file is omitted, the audio data is returned as a `matrix` of `numeric` values. It returns 2 rows (for both stereo channels). The number of columns correspond with the number of samples available from the buffer. The waveform is scaled between -1 and +1, such that it can be played directly with `audio::play()`.

Method `print()`**:** Prints basic information about the emulator output.

Usage:

```
RamigaOutput$print(...)
```

Arguments:

`...` Ignored

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
RamigaOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`save_wav` *Save audio to WAV file*

Description

Helper function to write stereo audio to a WAV file.

Usage

```
save_wav(samples, path, rate = 44100L)
```

Arguments

samples	A matrix of double values. Values need to be between -1 and +1. The matrix needs to have 2 rows. One for each channel.
path	Path of the file where the audio needs to be stored
rate	Sample rate. Defaults to 44100L

Value

Nothing

Index