Package: shinyphaser (via r-universe)

November 23, 2026

Title An Interface to the 'Phaser.js' Game Framework

Version 0.2.0

Description An API to build and control 2D games using the 'Phaser' 'JavaScript' engine. It enables integration with 'shiny' applications, allowing to create interactive games and simulations.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/maciekbanas/shinyphaser, https://maciekbanas.github.io/shinyphaser/

BugReports https://github.com/maciekbanas/shinyphaser/issues

Imports htmltools, jsonlite, R6, rlang, shiny

Suggests knitr, later, rmarkdown, shinyalert, testthat (>= 3.0.0), shinytest2 (>= 0.5.1), withr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Maciej Banas [aut, cre]

Maintainer Maciej Banas <banasmaciek@gmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-22 22:30:02 UTC

RemoteUrl https://github.com/cran/shinyphaser

RemoteRef HEAD

RemoteSha 6dfa2741abab8fbce72d8bf5531400121f986556

Contents

browser_actions . . . 2
Group . . . 2
Image . . . 4
PhaserGame . . . 6
Rectangle . . . 15
Sound . . . 18
Sprite . . . 19
StaticGroup . . . 25
StaticSprite . . . 26
Text . . . 28

Index **32**

`browser_actions` *Define actions that run immediately in the browser*

Description

Captures supported shinyphaser method calls without evaluating them in R. Browser actions are compiled when passed to an event registration method. Arbitrary R code belongs in the corresponding `server_action` function.

Usage

```
browser_actions(...)
```

Arguments

`...`	Supported shinyphaser browser action calls.

Value

A browser action specification.

`Group` *Group*

Description

Create and manage groups of sprites in the Phaser scene. Created with PhaserGame$add_group() method.

Methods

Public methods:

- `Group$new()`
- `Group$add_animation()`
- `Group$create()`
- `Group$show()`
- `Group$hide()`
- `Group$clone()`

Method `new()`**:** Create a dynamic group in the Phaser scene.

Usage:

```
Group$new(name, session = shiny::getDefaultReactiveDomain())
```

Arguments:

`name` Character. Unique name of the group.

`session` Shiny session object.

Method `add_animation()`**:** Add an animation that can be used by members of this group.

Usage:

```
Group$add_animation(
  suffix,
  url,
  frame_width,
  frame_height,
  frame_count,
  frame_rate
)
```

Arguments:

`suffix` Character. Animation suffix/key.

`url` Character. URL or path to spritesheet.

`frame_width` Numeric. Width of each frame.

`frame_height` Numeric. Height of each frame.

`frame_count` Numeric. Number of frames.

`frame_rate` Numeric. Frames per second.

Method `create()`**:** Create one group member at a coordinate.

Usage:

```
Group$create(x, y)
```

Arguments:

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

Method `show()`**:** Show all current members of this group.

Usage:

```
Group$show()
```

Returns: Invisible; sends a custom message to the client.

Method `hide()`**:** Hide all current members of this group.

Usage:

```
Group$hide()
```

Returns: Invisible; sends a custom message to the client.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Image` *Image*

Description

Create and manage images in the Phaser scene. Created with PhaserGame$add_image() method.

Methods

Public methods:

- `Image$new()`
- `Image$show()`
- `Image$hide()`
- `Image$follow_camera()`
- `Image$stop_camera_follow()`
- `Image$set_scroll_factor()`
- `Image$set_depth()`
- `Image$click()`
- `Image$clone()`

Method `new()`**:** Add an image object to the Phaser scene.

Usage:

```
Image$new(
  name,
  url,
  x,
  y,
  visible,
  clickable,
  session = getDefaultReactiveDomain()
)
```

Arguments:

`name` Character. Unique name of the image.

`url` Character. URL or path to image file.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`visible` Logical. Whether image is initially visible.

`clickable` Logical. Whether image emits click events.

`session` Shiny session object.

Method `show()`**:** Show a previously added image.

Usage:

```
Image$show()
```

Method `hide()`**:** Hide a previously added image.

Usage:

```
Image$hide()
```

Method `follow_camera()`**:** Make the camera follow this image as it moves through the world.

Usage:

```
Image$follow_camera(lerp_x = 1, lerp_y = 1, round_pixels = TRUE)
```

Arguments:

`lerp_x` Numeric. Horizontal interpolation factor from 0 to 1 (default: 1).

`lerp_y` Numeric. Vertical interpolation factor from 0 to 1 (default: 1).

`round_pixels` Logical. Whether to round camera pixels to avoid sub-pixel rendering (default: TRUE).

Method `stop_camera_follow()`**:** Stop the camera from following this image.

Usage:

```
Image$stop_camera_follow()
```

Method `set_scroll_factor()`**:** Set how much this image scrolls with the camera.

Usage:

```
Image$set_scroll_factor(x, y = x)
```

Arguments:

`x` Numeric. Horizontal scroll factor (0 = fixed to viewport, 1 = scrolls with world).

`y` Numeric. Vertical scroll factor. Defaults to `x`.

Method `set_depth()`**:** Set the image's rendering depth. Objects with a larger depth are rendered in front of objects with a smaller depth.

Usage:

```
Image$set_depth(depth)
```

Arguments:

`depth` Numeric. Phaser rendering depth.

Returns: This image object, invisibly, to support method chaining.

Method `click()`**:** Add a click event listener to the image that triggers an R function when clicked.

Usage:

```
Image$click(event_fun, input)
```

Arguments:

`event_fun` A function.

`input` Shiny input object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`PhaserGame` *PhaserGame*

Description

R6 class to create and manage a Phaser game within a Shiny application. Provides methods for adding sprites, animations, images, sounds, backgrounds, controls, and collision handling.

Public fields

`id` Character. ID of the Game container. Used as the HTML element ID where the game canvas will be rendered.

Methods

Public methods:

- `PhaserGame$new()`
- `PhaserGame$set_shiny_session()`
- `PhaserGame$save_game()`
- `PhaserGame$list_saved_games()`
- `PhaserGame$load_game()`
- `PhaserGame$use_phaser()`
- `PhaserGame$add_text()`
- `PhaserGame$add_rectangle()`
- `PhaserGame$add_image()`
- `PhaserGame$add_sound()`
- `PhaserGame$add_state()`

- PhaserGame$add_cooldown()
- PhaserGame$add_map()
- PhaserGame$activate_map()
- PhaserGame$add_proximity_trigger()
- PhaserGame$set_world_bounds()
- PhaserGame$enable_terrain_collision()
- PhaserGame$add_sprite()
- PhaserGame$add_collision_rectangle()
- PhaserGame$add_group()
- PhaserGame$add_static_sprite()
- PhaserGame$add_static_group()
- PhaserGame$add_collider()
- PhaserGame$add_overlap()
- PhaserGame$are_overlap()
- PhaserGame$add_overlap_end()
- PhaserGame$add_control()
- PhaserGame$clone()

Method `new()`**:** Create a PhaserGame object with the given configuration.

Usage:

```
PhaserGame$new(
  id = "phaser_game",
  width = 800,
  height = 600,
  gravity_x = 0,
  gravity_y = 0
)
```

Arguments:

`id` Character. ID of the Game container (defaults to "phaser_game").

`width` Numeric. Width of the Phaser canvas in pixels (defaults to 800).

`height` Numeric. Height of the Phaser canvas in pixels (defaults to 600).

`gravity_x` Numeric. Horizontal Arcade Physics world gravity (defaults to 0).

`gravity_y` Numeric. Vertical Arcade Physics world gravity (defaults to 0).

Returns: A new PhaserGame object.

Examples:

```
game <- PhaserGame$new(id = "my_game", width = 1024, height = 768)
```

Method `set_shiny_session()`**:** Set the Shiny session used to send Phaser custom messages.

Usage:

```
PhaserGame$set_shiny_session(session = shiny::getDefaultReactiveDomain())
```

Arguments:

`session` Shiny session object (default: shiny::getDefaultReactiveDomain()).

Method `save_game()`**:** Save game data and Phaser object state to a JSON file on the server. Positions are read directly from Phaser immediately before the save request, avoiding stale Shiny coordinates.

Usage:

```
PhaserGame$save_game(
  name,
  state = list(),
  objects = NULL,
  snapshot = NULL,
  directory = NULL
)
```

Arguments:

`name` Character. Human-readable name of the save.

`state` Named list. Additional JSON-serializable application state.

`objects` Character vector. Named Phaser scene objects to capture. By default all named scene objects are captured.

`snapshot` Named list. Optional Phaser object snapshot already captured in the browser. Supplying it writes the save synchronously.

`directory` Character. Server-side directory. Defaults to a game-specific folder below `tempdir()`.

Returns: Invisible request identifier. The disk write completes asynchronously after Phaser returns its snapshot.

Method `list_saved_games()`**:** List saves stored on the server for this game.

Usage:

```
PhaserGame$list_saved_games(directory = NULL)
```

Arguments:

`directory` Character. Server-side save directory.

Returns: A list of saved game records, newest first.

Method `load_game()`**:** Load a saved game from server disk.

Usage:

```
PhaserGame$load_game(name, restore = TRUE, directory = NULL)
```

Arguments:

`name` Character. Save name.

`restore` Logical. Restore captured Phaser object properties.

`directory` Character. Server-side save directory.

Returns: The additional application state stored with the save, invisibly.

Method `use_phaser()`**:** Load dependencies and initialize the Phaser game in the UI.

Usage:

```
PhaserGame$use_phaser()
```

Returns: HTML tag list containing dependencies and initialization script.

Examples:

```
game$use_phaser()
```

Method `add_text()`**:** Add a text object to the Phaser scene.

Usage:

```
PhaserGame$add_text(
  text,
  id,
  x,
  y,
  style = list(font_size = "22px"),
  visible = TRUE
)
```

Arguments:

`text` Character. Text value to display.

`id` Character. Unique ID for the text object.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`style` Named list. Styling options passed to Phaser text rendering.

`visible` Logical. Whether text is initially visible.

Method `add_rectangle()`**:** Add a rectangle object to the Phaser scene.

Usage:

```
PhaserGame$add_rectangle(
  name,
  x,
  y,
  width,
  height,
  color,
  visible = TRUE,
  clickable = FALSE
)
```

Arguments:

`name` Character. Unique name for the rectangle.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`width` Numeric. Rectangle width in pixels.

`height` Numeric. Rectangle height in pixels.

`color` Character. Fill color in Phaser-compatible format.

`visible` Logical. Whether rectangle is initially visible.

`clickable` Logical. Whether rectangle emits click events.

Method `add_image()`**:** Adds a static image to the Phaser scene.

Usage:

```
PhaserGame$add_image(name, url, x, y, visible = TRUE, clickable = FALSE)
```

Arguments:

`name` Character. Unique key to reference this image.

`url` Character. URL or path to the image file.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`visible` Logical. Whether the image is initially visible (default: TRUE).

`clickable` Logical. Whether the image should emit click events (default: FALSE).

Method `add_sound()`**:** Adds a sound to the Phaser scene.

Usage:

```
PhaserGame$add_sound(name, url, volume = 1, loop = FALSE)
```

Arguments:

`name` Character. Unique key to reference this sound.

`url` Character. URL or path to the audio file.

`volume` Numeric. Initial playback volume from 0 to 1 (default: 1).

`loop` Logical. Whether the sound should loop by default (default: FALSE).

Method `add_state()`**:** Create browser-resident state for use in browser actions.

Usage:

```
PhaserGame$add_state(key, initial = NULL)
```

Arguments:

`key` Character. Unique state key.

`initial` Initial JSON-serializable value.

Method `add_cooldown()`**:** Create a browser-resident cooldown for browser-action conditions.

Usage:

```
PhaserGame$add_cooldown(key, duration)
```

Arguments:

`key` Character. Unique cooldown key.

`duration` Numeric. Cooldown duration in milliseconds.

Method `add_map()`**:** Add a background (tilemap) layer from Tiled JSON + tileset image(s).

Usage:

```
PhaserGame$add_map(map_key, map_url, tileset_urls, tileset_names, layer_name)
```

Arguments:

`map_key` Character. Key for the tilemap JSON.

`map_url` Character. URL of the Tiled JSON file (relative to www/assets/).

`tileset_urls` Character vector. URLs of tileset image files.

`tileset_names` Character vector. Names of tilesets as defined in Tiled.

`layer_name` Character. Name of the layer to render from Tiled.

Returns: Invisible; sends a custom message to the client.

Method `activate_map()`**:** Activate a tilemap previously loaded with `add_map()`.

Usage:

```
PhaserGame$activate_map(
  map_key,
  player_name = NULL,
  x = NULL,
  y = NULL,
  visible_objects = character(),
  hidden_objects = character()
)
```

Arguments:

`map_key` Character. Key of the tilemap to activate.

`player_name` Character. Optional player sprite to reposition.

`x` Numeric. Optional player x-coordinate.

`y` Numeric. Optional player y-coordinate.

`visible_objects` Character vector. Scene objects to show and enable.

`hidden_objects` Character vector. Scene objects to hide and disable.

Returns: Invisible; sends a custom message to the client.

Method `add_proximity_trigger()`**:** Add a trigger that monitors when a scene object is near a point.

Usage:

```
PhaserGame$add_proximity_trigger(
  id,
  object_name,
  x,
  y,
  radius = 180,
  element_id = NULL,
  context = NULL,
  input_id = NULL
)
```

Arguments:

`id` Character. Unique trigger identifier.

`object_name` Character. Scene object whose position is monitored.

`x` Numeric. Target x-coordinate.

`y` Numeric. Target y-coordinate.

`radius` Numeric. Maximum distance at which the trigger is active.

`element_id` Character or `NULL`. ID of an HTML element to show while the trigger is active.

`context` Character or `NULL`. Optional map key that must be active for the trigger to be evaluated.

`input_id` Character or `NULL`. Optional Shiny input that receives a payload whenever the trigger is entered or exited.

Returns: Invisible; sends a custom message to the client.

Method `set_world_bounds()`**:** Set the Phaser physics world and camera bounds.

Usage:

```
PhaserGame$set_world_bounds(width, height)
```

Arguments:

`width` Numeric. World width in pixels.

`height` Numeric. World height in pixels.

Returns: Invisible; sends a custom message to the client.

Method `enable_terrain_collision()`**:** Enable terrain collision for a player sprite.

Usage:

```
PhaserGame$enable_terrain_collision(name)
```

Arguments:

`name` Character. Name of the player sprite (as added via add_player_sprite).

Method `add_sprite()`**:** Load a base spritesheet and create an "idle" animation.

Usage:

```
PhaserGame$add_sprite(
  name,
  url,
  x,
  y,
  frame_width,
  frame_height,
  frame_count = 1,
  frame_rate = 1
)
```

Arguments:

`name` Character. Unique key for the sprite and its idle animation.

`url` Character. URL or path to the spritesheet image.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`frame_width` Numeric. Width of each frame.

`frame_height` Numeric. Height of each frame.

`frame_count` Numeric. Number of frames in the spritesheet.

`frame_rate` Numeric. Frames per second for the idle animation.

Method `add_collision_rectangle()`**:** Add an invisible static collision rectangle to the world.

Usage:

```
PhaserGame$add_collision_rectangle(name, x, y, width, height)
```

Arguments:

`name` Character. Unique name of the collision rectangle.

`x` Numeric. X-coordinate of its center.

`y` Numeric. Y-coordinate of its center.

`width` Numeric. Collision width in pixels.

`height` Numeric. Collision height in pixels.

Method `add_group()`**:** Adds a dynamic group from a spritesheet.

Usage:

```
PhaserGame$add_group(name)
```

Arguments:

`name` Character. Unique name of the group.

Method `add_static_sprite()`**:** Adds a static sprite to the scene (non-animated).

Usage:

```
PhaserGame$add_static_sprite(name, url, x, y)
```

Arguments:

`name` Character. Unique name of the sprite.
`url` Character. URL or path to the image file.
`x` Numeric. X-coordinate in pixels.
`y` Numeric. Y-coordinate in pixels.

Method `add_static_group()`**:** Adds a static group to the scene (non-animated).

Usage:

```
PhaserGame$add_static_group(name, url)
```

Arguments:

`name` Character. Unique name of the group.
`url` Character. URL or path to the image file.

Method `add_collider()`**:** Adds a collider between two game objects.

Usage:

```
PhaserGame$add_collider(
  object_one,
  object_two = NULL,
  group = NULL,
  browser_action = browser_actions(),
  input = NULL,
  server_action = NULL
)
```

Arguments:

`object_one` Character. Name of the first object.
`object_two` Character. Name of the second object.
`group` Character. Name of the group to compare against.
`browser_action` Actions created with `browser_actions()` that run immediately in the browser.
`input` Shiny input list.
`server_action` Function called in R with the collision event.

Method `add_overlap()`**:** Adds a collider between two game objects.

Usage:

```
PhaserGame$add_overlap(
  object_one,
  object_two = NULL,
  group = NULL,
  browser_action = browser_actions(),
  input = NULL,
  server_action = NULL,
  mode = c("enter", "stay"),
  interval = 0
)
```

Arguments:

`object_one` Character. Name of the first object.

`object_two` Character. Name of the second object.

`group` Character. Name of the group.

`browser_action` Actions created with `browser_actions()` that run immediately in the browser.

`input` Shiny input list.

`server_action` Function called in R with the overlap event.

`mode` Character. `"enter"` (default) runs once per contact; `"stay"` repeats while overlapping.

`interval` Numeric. Minimum milliseconds between `"stay"` actions.

Method `are_overlap()`**:** Create a reactive expression for overlap state between two objects.

Usage:

```
PhaserGame$are_overlap(object_one, object_two, input)
```

Arguments:

`object_one` Character. Name of the first object.

`object_two` Character. Name of the second object.

`input` Shiny input list.

Method `add_overlap_end()`**:** Register a callback fired when overlap between objects ends.

Usage:

```
PhaserGame$add_overlap_end(
  object_one,
  object_two = NULL,
  group = NULL,
  browser_action = browser_actions(),
  input = NULL,
  server_action = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`object_one` Character. Name of the first object.

`object_two` Character. Name of the second object.

`group` Character. Name of the group to compare against.

`browser_action` Actions created with `browser_actions()` that run immediately in the browser.

`input` Shiny input list.

`server_action` Function called in R with the overlap-end event.

`session` Shiny session object.

Method `add_control()`**:** Register a callback fired when a specific key is pressed.

Usage:

```
PhaserGame$add_control(
  key,
  browser_action = browser_actions(),
  input = NULL,
  server_action = NULL
)
```

Arguments:

`key` A character, accepts Javascript key events (they need to align with event.code).

`browser_action` Actions created with `browser_actions()` that run immediately in the browser.

`input` Shiny input list.

`server_action` Function called in R with the keyboard event.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `PhaserGame$new`
## ------------------------------------------------

game <- PhaserGame$new(id = "my_game", width = 1024, height = 768)

## ------------------------------------------------
## Method `PhaserGame$use_phaser`
## ------------------------------------------------

 game$use_phaser()
```

`Rectangle` *Rectangle*

Description

Create and manage rectangles in the Phaser scene. Created with PhaserGame$add_rectangle() method.

Methods

Public methods:

- Rectangle$new()
- Rectangle$show()
- Rectangle$hide()
- Rectangle$follow_camera()
- Rectangle$stop_camera_follow()
- Rectangle$set_scroll_factor()
- Rectangle$set_depth()
- Rectangle$click()
- Rectangle$clone()

Method `new()`**:** Add a rectangle object to the Phaser scene.

Usage:

```
Rectangle$new(
  name,
  x,
  y,
  width,
  height,
  color,
  visible,
  clickable,
  session = getDefaultReactiveDomain()
)
```

Arguments:

`name` Character. Unique name of the rectangle.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`width` Numeric. Rectangle width in pixels.

`height` Numeric. Rectangle height in pixels.

`color` Character. Fill color in Phaser-compatible format.

`visible` Logical. Whether rectangle is initially visible.

`clickable` Logical. Whether rectangle emits click events.

`session` Shiny session object.

Method `show()`**:** Show a previously added rectangle.

Usage:

```
Rectangle$show()
```

Method `hide()`**:** Hide a previously added rectangle.

Usage:

```
Rectangle$hide()
```

Method `follow_camera()`**:** Make the camera follow this rectangle as it moves through the world.

Usage:

```
Rectangle$follow_camera(lerp_x = 1, lerp_y = 1, round_pixels = TRUE)
```

Arguments:

`lerp_x` Numeric. Horizontal interpolation factor from 0 to 1 (default: 1).

`lerp_y` Numeric. Vertical interpolation factor from 0 to 1 (default: 1).

`round_pixels` Logical. Whether to round camera pixels to avoid sub-pixel rendering (default: TRUE).

Method `stop_camera_follow()`**:** Stop the camera from following this rectangle.

Usage:

```
Rectangle$stop_camera_follow()
```

Method `set_scroll_factor()`**:** Set how much this rectangle scrolls with the camera.

Usage:

```
Rectangle$set_scroll_factor(x, y = x)
```

Arguments:

`x` Numeric. Horizontal scroll factor (0 = fixed to viewport, 1 = scrolls with world).

`y` Numeric. Vertical scroll factor. Defaults to `x`.

Method `set_depth()`**:** Set the rectangle's rendering depth. Objects with a larger depth are rendered in front of objects with a smaller depth.

Usage:

```
Rectangle$set_depth(depth)
```

Arguments:

`depth` Numeric. Phaser rendering depth.

Returns: This rectangle object, invisibly, to support method chaining.

Method `click()`**:** Add a click event listener to the rectangle that triggers an R function when clicked.

Usage:

```
Rectangle$click(event_fun, input)
```

Arguments:

`event_fun` A function.

`input` Shiny input object.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Sound *Sound*

Description

Create and manage audio in the Phaser scene. Created with PhaserGame$add_sound() method.

Methods

Public methods:

- `Sound$new()`
- `Sound$play()`
- `Sound$pause()`
- `Sound$resume()`
- `Sound$stop()`
- `Sound$set_volume()`
- `Sound$set_loop()`
- `Sound$clone()`

Method `new()`**:** Load a sound file and register it with the Phaser scene.

Usage:

```
Sound$new(
  name,
  url,
  volume = 1,
  loop = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`name` Character. Unique key for the sound.

`url` Character. URL or path to the audio file.

`volume` Numeric. Initial playback volume from 0 to 1 (default: 1).

`loop` Logical. Whether the sound should loop by default (default: FALSE).

`session` Shiny session object.

Method `play()`**:** Play the sound.

Usage:

```
Sound$play(volume = NULL, loop = NULL)
```

Arguments:

`volume` Numeric. Optional playback volume from 0 to 1. Defaults to the sound's current/default volume.

`loop` Logical. Optional loop setting for this playback. Defaults to the sound's current/default loop setting.

Method `pause()`**:** Pause the sound.

Usage:

```
Sound$pause()
```

Method `resume()`**:** Resume a paused sound.

Usage:

```
Sound$resume()
```

Method `stop()`**:** Stop the sound.

Usage:

```
Sound$stop()
```

Method `set_volume()`**:** Set the sound's volume.

Usage:

```
Sound$set_volume(volume)
```

Arguments:

`volume` Numeric. Volume from 0 to 1.

Method `set_loop()`**:** Set whether the sound loops by default.

Usage:

```
Sound$set_loop(loop)
```

Arguments:

`loop` Logical. Whether playback should loop.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Sprite` *Sprite*

Description

Create and manage animated sprites in the Phaser scene. Created with PhaserGame$add_sprite() method.

Methods

Public methods:

- Sprite$new()
- Sprite$add_animation()
- Sprite$play_animation()
- Sprite$stop_motion()
- Sprite$show()
- Sprite$hide()
- Sprite$add_player_controls()
- Sprite$set_player_animation_prefix()
- Sprite$follow_camera()
- Sprite$stop_camera_follow()
- Sprite$set_scroll_factor()
- Sprite$set_depth()
- Sprite$set_body_size()
- Sprite$set_depth_by_y()
- Sprite$set_velocity_x()
- Sprite$set_velocity_y()
- Sprite$set_gravity()
- Sprite$set_bounce()
- Sprite$destroy()
- Sprite$set_in_motion()
- Sprite$start_approach_on_sight()
- Sprite$clone()

Method `new()`:

Usage:

```
Sprite$new(
  name,
  url,
  x,
  y,
  frame_width,
  frame_height,
  frame_count = NULL,
  frame_rate,
  session = getDefaultReactiveDomain()
)
```

Arguments:

`name` Character. Unique key for the sprite and its idle animation.

`url` Character. URL or path to the spritesheet image.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`frame_width` Numeric. Width of each frame.

`frame_height` Numeric. Height of each frame.

`frame_count` Numeric. Number of frames in the spritesheet. If NULL, auto-detect from spritesheet dimensions.

`frame_rate` Numeric. Frames per second for the idle animation.

`session` Shiny session object.

Method `add_animation()`**:** Load a custom animation for any sprite previously added.

Usage:

```
Sprite$add_animation(
  suffix,
  url,
  frame_width,
  frame_height,
  frame_count = NULL,
  frame_rate
)
```

Arguments:

`suffix` Character. Identifier for this animation (e.g. "move_left").

`url` Character. URL or path to the spritesheet.

`frame_width` Numeric. Width of each frame.

`frame_height` Numeric. Height of each frame.

`frame_count` Numeric. Number of frames in the spritesheet. If NULL, auto-detect from spritesheet dimensions.

`frame_rate` Numeric. Frames per second for playback.

Returns: Invisible; sends a custom message to the client.

Method `play_animation()`**:** Play a loaded animation for the sprite.

Usage:

```
Sprite$play_animation(anim_name, duration = Inf)
```

Arguments:

`anim_name` Character. Identifier for the animation to play (e.g. " move_left").

`duration` Numeric. Optional duration in milliseconds to play the animation before reverting to idle (defaults to Inf, which loops indefinitely until another animation is played).

Method `stop_motion()`**:** Stop this sprite's current scripted movement.

Usage:

```
Sprite$stop_motion()
```

Returns: Invisible; sends a custom message to the client.

Method `show()`**:** Show this sprite.

Usage:

```
Sprite$show()
```

Returns: Invisible; sends a custom message to the client.

Method `hide()`**:** Hide this sprite.

Usage:

```
Sprite$hide()
```

Returns: Invisible; sends a custom message to the client.

Method `add_player_controls()`**:** Enable movement controls (arrow keys) for a player sprite.

Usage:

```
Sprite$add_player_controls(
  directions = c("left", "right", "down", "up"),
  speed = 200
)
```

Arguments:

`directions` Character vector. Directions to enable (defaults to c("left","right","down","up")).

`speed` Numeric. Movement speed in pixels/second (default: 200).

Method `set_player_animation_prefix()`**:** Choose the animation prefix used by player controls.

Usage:

```
Sprite$set_player_animation_prefix(prefix)
```

Arguments:

`prefix` Character. Prefix for idle and directional movement animation keys.

Returns: Invisible; sends a custom message to the client.

Method `follow_camera()`**:** Make the camera follow this sprite as it moves through the world.

Usage:

```
Sprite$follow_camera(lerp_x = 1, lerp_y = 1, round_pixels = TRUE)
```

Arguments:

`lerp_x` Numeric. Horizontal interpolation factor from 0 to 1 (default: 1).

`lerp_y` Numeric. Vertical interpolation factor from 0 to 1 (default: 1).

`round_pixels` Logical. Whether to round camera pixels to avoid sub-pixel rendering (default: TRUE).

Method `stop_camera_follow()`**:** Stop the camera from following this sprite.

Usage:

```
Sprite$stop_camera_follow()
```

Method `set_scroll_factor()`**:** Set how much this sprite scrolls with the camera.

Usage:

```
Sprite$set_scroll_factor(x, y = x)
```

Arguments:

`x` Numeric. Horizontal scroll factor (0 = fixed to viewport, 1 = scrolls with world).

`y` Numeric. Vertical scroll factor. Defaults to `x`.

Method `set_depth()`**:** Set the sprite's rendering depth. Objects with a larger depth are rendered in front of objects with a smaller depth.

Usage:

```
Sprite$set_depth(depth)
```

Arguments:

`depth` Numeric. Phaser rendering depth.

Returns: This sprite object, invisibly, to support method chaining.

Method `set_body_size()`**:** Set a collision body size and its offset within the unscaled sprite frame. Calls made before the sprite loads are queued.

Usage:

```
Sprite$set_body_size(width, height, offset_x = 0, offset_y = 0)
```

Arguments:

`width, height` Positive numeric. Body size in frame pixels.

`offset_x, offset_y` Numeric. Offset from the frame's top-left corner.

Returns: This sprite, invisibly.

Method `set_depth_by_y()`**:** Continuously set drawing depth to the sprite's y-coordinate plus an offset, in the browser. Disable to retain the current depth; set_depth() also disables automatic depth sorting.

Usage:

```
Sprite$set_depth_by_y(offset = 0, enabled = TRUE)
```

Arguments:

`offset` Numeric. Vertical distance from the sprite origin to its ground point, in world pixels (40 for the example's centered Orc).

`enabled` Logical. Whether to update depth each frame.

Returns: This sprite, invisibly.

Method `set_velocity_x()`**:** Set the sprite's velocity in the x direction.

Usage:

```
Sprite$set_velocity_x(x = 100)
```

Arguments:

`x` Numeric. Velocity in pixels/second (positive = right, negative = left).

Method `set_velocity_y()`**:** Set the sprite's velocity in the y direction.

Usage:

```
Sprite$set_velocity_y(x = 100)
```

Arguments:

`x` Numeric. Velocity in pixels/second (positive = down, negative = up).

Method `set_gravity()`**:** Set the sprite's velocity in both x and y directions.

Usage:

```
Sprite$set_gravity(x = 100, y = 100)
```

Arguments:

`x` Numeric. Velocity in pixels/second (positive = right, negative = left).

`y` Numeric. Velocity in pixels/second (positive = down, negative = up).

Method `set_bounce()`**:** Set the sprite's bounce factor.

Usage:

```
Sprite$set_bounce(x)
```

Arguments:

`x` Numeric. Bounce factor.

Method `destroy()`**:** Remove sprite from the scene.

Usage:

```
Sprite$destroy()
```

Method `set_in_motion()`**:** Move sprite along a vector for a set distance.

Usage:

```
Sprite$set_in_motion(dir_x, dir_y, speed, distance, lag = distance/speed)
```

Arguments:

`dir_x` Numeric. Horizontal direction (-1 = left, +1 = right, 0 = none).

`dir_y` Numeric. Vertical direction (-1 = up, +1 = down, 0 = none).

`speed` Numeric. Speed in pixels/second.

`distance` Numeric. Distance in pixels to travel before stopping.

`lag` Numeric. Optional delay before sending the command (defaults to distance/speed).

Method `start_approach_on_sight()`**:** Start client-side sight checks that make this sprite alert, wander while the target is out of range, and approach the target when it is in range. This is a reusable browser-side behaviour rather than an RPG-specific rule.

Usage:

```
Sprite$start_approach_on_sight(
  target_name,
  sight_range,
  speed,
  distance,
  check_interval = 250,
  alert_duration = 1200,
  wander_interval = 1500
)
```

Arguments:

`target_name` Character. Name of the target sprite to approach.

`sight_range` Numeric. Maximum distance in pixels at which the target is noticed.

`speed` Numeric. Approach speed in pixels/second.

`distance` Numeric. Distance in pixels to travel for each approach step.

`check_interval` Numeric. Milliseconds between sight checks.

`alert_duration` Numeric. Milliseconds to show the alert while approaching.

`wander_interval` Numeric. Milliseconds between random movements while the target is out of sight.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`StaticGroup` *Static Group*

Description

Create and manage groups of static sprites in the Phaser scene. Created with PhaserGame$add_static_group() method.

Methods

Public methods:

- `StaticGroup$new()`
- `StaticGroup$create()`
- `StaticGroup$show()`
- `StaticGroup$hide()`
- `StaticGroup$disable()`
- `StaticGroup$clone()`

Method `new()`**:** Create a static group from a base image.

Usage:

```
StaticGroup$new(name, url, session = shiny::getDefaultReactiveDomain())
```

Arguments:

`name` Character. Unique name of the group.

`url` Character. URL or path to image file.

`session` Shiny session object.

Method `create()`**:** Create one static group member at a coordinate.

Usage:

```
StaticGroup$create(x, y)
```

Arguments:

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

Method `show()`**:** Show all current members of this static group.

Usage:

```
StaticGroup$show()
```

Returns: Invisible; sends a custom message to the client.

Method `hide()`**:** Hide all current members of this static group.

Usage:

```
StaticGroup$hide()
```

Returns: Invisible; sends a custom message to the client.

Method `disable()`**:** Disable a static group member body based on overlap event payload.

Usage:

```
StaticGroup$disable(evt)
```

Arguments:

`evt` List-like event payload containing x2 and y2 values.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`StaticSprite` *Static Sprite*

Description

Create and manage non-animated sprites in the Phaser scene. Created with PhaserGame$add_static_sprite() method.

Methods

Public methods:

- `StaticSprite$new()`
- `StaticSprite$destroy()`
- `StaticSprite$show()`
- `StaticSprite$hide()`
- `StaticSprite$follow_camera()`
- `StaticSprite$stop_camera_follow()`
- `StaticSprite$set_scroll_factor()`
- `StaticSprite$set_depth()`
- `StaticSprite$clone()`

Method `new()`**:** Add a non-animated static sprite to the scene.

Usage:

```
StaticSprite$new(name, url, x, y, session = getDefaultReactiveDomain())
```

Arguments:

`name` Character. Unique name of the sprite.

`url` Character. URL or path to image file.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`session` Shiny session object.

Method `destroy()`**:** Remove static sprite from the scene.

Usage:

```
StaticSprite$destroy()
```

Method `show()`**:** Show this static sprite.

Usage:

```
StaticSprite$show()
```

Returns: Invisible; sends a custom message to the client.

Method `hide()`**:** Hide this static sprite.

Usage:

```
StaticSprite$hide()
```

Returns: Invisible; sends a custom message to the client.

Method `follow_camera()`**:** Make the camera follow this static sprite as it moves through the world.

Usage:

```
StaticSprite$follow_camera(lerp_x = 1, lerp_y = 1, round_pixels = TRUE)
```

Arguments:

`lerp_x` Numeric. Horizontal interpolation factor from 0 to 1 (default: 1).

`lerp_y` Numeric. Vertical interpolation factor from 0 to 1 (default: 1).

`round_pixels` Logical. Whether to round camera pixels to avoid sub-pixel rendering (default: TRUE).

Method `stop_camera_follow()`**:** Stop the camera from following this static sprite.

Usage:

```
StaticSprite$stop_camera_follow()
```

Method `set_scroll_factor()`**:** Set how much this static sprite scrolls with the camera.

Usage:

```
StaticSprite$set_scroll_factor(x, y = x)
```

Arguments:

`x` Numeric. Horizontal scroll factor (0 = fixed to viewport, 1 = scrolls with world).

`y` Numeric. Vertical scroll factor. Defaults to `x`.

Method `set_depth()`**:** Set the static sprite's rendering depth. Objects with a larger depth are rendered in front of objects with a smaller depth.

Usage:

```
StaticSprite$set_depth(depth)
```

Arguments:

`depth` Numeric. Phaser rendering depth.

Returns: This static sprite object, invisibly, to support method chaining.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`Text` *Text*

Description

R6 class to represent a text object in the Phaser scene, allowing dynamic updates to its content. Created with PhaserGame$add_text() method.

Methods

Public methods:

- `Text$new()`
- `Text$set()`
- `Text$show()`
- `Text$hide()`
- `Text$follow_camera()`
- `Text$stop_camera_follow()`
- `Text$set_scroll_factor()`
- `Text$set_depth()`
- `Text$clone()`

Method `new()`**:** Create a text object in the Phaser scene.

Usage:

```
Text$new(
  text,
  id,
  x,
  y,
  style,
  visible = TRUE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments:

`text` Character. Text value to display.

`id` Character. Unique ID for the text object.

`x` Numeric. X-coordinate in pixels.

`y` Numeric. Y-coordinate in pixels.

`style` Named list. Styling options passed to Phaser text rendering.

`visible` Logical. Whether text is initially visible.

`session` Shiny session object.

Method `set()`**:** Update the text content of this object.

Usage:

```
Text$set(text)
```

Arguments:

`text` Character. New text value to display.

Method `show()`**:** Show a previously added text object.

Usage:

```
Text$show()
```

Method `hide()`**:** Hide a previously added text object.

Usage:

```
Text$hide()
```

Method `follow_camera()`**:** Make the camera follow this text object as it moves through the world.

Usage:

```
Text$follow_camera(lerp_x = 1, lerp_y = 1, round_pixels = TRUE)
```

Arguments:

`lerp_x` Numeric. Horizontal interpolation factor from 0 to 1 (default: 1).

`lerp_y` Numeric. Vertical interpolation factor from 0 to 1 (default: 1).

`round_pixels` Logical. Whether to round camera pixels to avoid sub-pixel rendering (default: TRUE).

Method `stop_camera_follow()`**:** Stop the camera from following this text object.

Usage:

```
Text$stop_camera_follow()
```

Method `set_scroll_factor()`**:** Set how much this text object scrolls with the camera.

Usage:

```
Text$set_scroll_factor(x, y = x)
```

Arguments:

`x` Numeric. Horizontal scroll factor (0 = fixed to viewport, 1 = scrolls with world).

`y` Numeric. Vertical scroll factor. Defaults to `x`.

Method `set_depth()`**:** Set the text object's rendering depth. Objects with a larger depth are rendered in front of objects with a smaller depth.

Usage:

```
Text$set_depth(depth)
```

Arguments:

`depth` Numeric. Phaser rendering depth.

Details: Every individually renderable object wrapper supports `set_depth()`. Larger values draw in front of smaller values. The method invisibly returns its object, so it can be chained with other object methods.

Returns: This text object, invisibly, to support method chaining.

Examples:

```
\dontrun{
background <- game$add_image(
  "background", "assets/background.png", x = 400, y = 300
)
background$set_depth(-10)

control <- game$add_rectangle(
  "jump_control", x = 720, y = 540, width = 120, height = 48,
  color = "0x223344"
)
control$set_depth(50)$set_scroll_factor(0)

label <- game$add_text("Jump", "jump_label", x = 690, y = 528)
label$set_depth(51)$set_scroll_factor(0)
}
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## ------------------------------------------------
## Method `Text$set_depth`
## ------------------------------------------------

## Not run:
background <- game$add_image(
  "background", "assets/background.png", x = 400, y = 300
)
background$set_depth(-10)

control <- game$add_rectangle(
  "jump_control", x = 720, y = 540, width = 120, height = 48,
  color = "0x223344"
)
control$set_depth(50)$set_scroll_factor(0)

label <- game$add_text("Jump", "jump_label", x = 690, y = 528)
label$set_depth(51)$set_scroll_factor(0)

## End(Not run)
```

Index

browser_actions, 2
browser_actions(), *13–15*

Group, 2

Image, 4

PhaserGame, 6

Rectangle, 15

Sound, 18
Sprite, 19
StaticGroup, 25
StaticSprite, 26

tempdir(), *8*
Text, 28