Package: shinyphaser (via r-universe)

June 8, 2026

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

Version 0.1.0

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.0

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

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

Imports htmltools, R6, rlang, shiny

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

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

Date/Publication 2026-06-07 20:41:33 UTC

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

RemoteRef HEAD

RemoteSha 29cc98ca37cb7fe39746478f185b7ce20acc6406

Contents

Group 2
Image 3
PhaserGame 4

Rectangle 10
run_sample_app 11
Sprite 12
StaticGroup 15
StaticSprite 16
Text 16

Index **18**

`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$create()`
- `Group$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 `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$hide()`
- `Image$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 `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, 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$use_phaser()
- PhaserGame$add_text()
- PhaserGame$add_rectangle()
- PhaserGame$add_image()
- PhaserGame$add_map()
- PhaserGame$enable_terrain_collision()
- PhaserGame$add_sprite()
- PhaserGame$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)
```

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).

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 `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"))
```

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.

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_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 `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_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,
  callback_fun = NULL,
  input
)
```

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.

`callback_fun` A function to be run when collision occurs.

`input` Shiny input list.

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

Usage:

```
PhaserGame$add_overlap(
  object_one,
  object_two = NULL,
  group = NULL,
  callback_fun,
  input
)
```

Arguments:

`object_one` Character. Name of the first object.

`object_two` Character. Name of the second object.

`group` Character. Name of the group.

`callback_fun` A function to be run when overlap occurs.

`input` Shiny input list.

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,
  callback_fun,
  input,
  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.

`callback_fun` Function. Callback executed when overlap ends.

`input` Shiny input list.

`session` Shiny session object.

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

Usage:

```
PhaserGame$add_control(key, action, input)
```

Arguments:

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

`action` A function to be run after key is pressed.

`input` Shiny input list.

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$hide()
- Rectangle$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 `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.

`run_sample_app` *Run the packaged shinyphaser sample app*

Description

Launches the sample Shiny application bundled with the package. This is a quick way to see a working `shinyphaser` game setup.

Usage

```
run_sample_app()
```

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$play_animation()
- Sprite$add_player_controls()
- Sprite$set_velocity_x()
- Sprite$set_velocity_y()
- Sprite$set_gravity()
- Sprite$set_bounce()
- Sprite$destroy()
- Sprite$set_in_motion()
- 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 `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_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 `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$disable()
- StaticGroup$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 `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$new()`
- `StaticSprite$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 `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$clone()

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

Usage:

```
Text$new(text, id, x, y, style, 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.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

Index

Group, 2

Image, 3

PhaserGame, 4

Rectangle, 10
run_sample_app, 11

Sprite, 12
StaticGroup, 15
StaticSprite, 16

Text, 16