El Polo Loco Js Doc

Inhalt

Global

`InitiateGame`
`checkForGameEnd`
`fullscreen`
`info`
`reset`
`toggleMusic`

Classes

`BackgroundObject`
`Character`
`Chicken`
`Cloude`
`CollectableObjects`
`EndScreen`
`Endboss`
`Keyboard`
`Level`
`SmallChicken`
`StatusBar`
`ThrowableObject`
`World`

Global Methods

`InitiateGame()`

Starts the game when the button is clicked. - Hides the start screen. - Initializes a new World object. - Starts checking for game end condition. - Hides the help bar.

Source:

game.js, line 14

`checkForGameEnd()`

Starts an interval to check if the game has ended. - If the world's `gameEnde` property is true (game ended), - shows the restart button. - clears the game's background sound interval. - pauses the game's background sound.

Source:

game.js, line 32

`fullscreen()`

Toggles fullscreen mode for the game container. - If not in fullscreen, requests fullscreen mode for the game container. - Hides the info and help bar elements. - If already in fullscreen, exits fullscreen mode. - Shows the info element.

Source:

game.js, line 85

`info()`

Toggles the visibility of the help bar.

Source:

game.js, line 100

`reset()`

Resets the game state when the button is clicked. - Clears references to world and level objects. - Hides the restart button. - Hides the canvas element. - Clears the game end check interval. - Shows the start screen.

Source:

game.js, line 51

`toggleMusic()`

Toggles music on/off based on the current music state. - Updates the music button image based on the music state. - Sets the `music` variable (presumably controls music playback).

Source:

game.js, line 66

Class: BackgroundObject

BackgroundObject`(imagePath, x)`

`new BackgroundObject(imagePath, x)`

Creates a new background object instance.

Parameters:

Name	Type	Description
`imagePath`	string	The path to the image for the background object.
`x`	number	The initial x-coordinate of the object on the canvas.

Source:

background-objeckt.class.js, line 11

Class: Character

Character()

`new Character()`

Loads the default idle image for the character.

Source:

character.class.js, line 75

Members

`animation :number`

Starts an interval that checks the character's state and plays the corresponding animation (dead, hurt, jumping, running, idle). The interval ID for the main animation loop (private property).

Type:

- number

Source:

character.class.js, line 134

`enableMovment :number`

Starts an interval that handles character movement based on keyboard input and jumping. The interval ID for movement and jump handling (private property)

Type:

- number

Source:

character.class.js, line 186

`idelAnimation :number`

Starts an interval that plays the long idle animation when certain conditions are met (not jumping, on ground, not moving). The interval ID for the long idle animation (private property)

Type:

- number

Source:

character.class.js, line 111

Methods

`deadAnimation()`

Plays the death animation and sets the game to over state.

Source:

character.class.js, line 150

`enableWalking()`

Handles character movement to the left or right based on keyboard input and keeps the camera centered.

Source:

character.class.js, line 206

`hurtAnimation()`

Plays the hurt animation and resets the idle timer.

Source:

character.class.js, line 158

`idleEnd()`

Resets the idle start time.

Source:

character.class.js, line 125

`idleStart()`

Records the start time for the idle animation.

Source:

character.class.js, line 91

`idleTime() → {boolean}`

Calculates if the character has been idle for more than 3 seconds. True if the character has been idle for more than 3 seconds, false otherwise.

Source:

character.class.js, line 100

Returns:

Type

boolean

`jump()`

Sets the character's vertical speed for jumping.

Source:

character.class.js, line 223

`jumpAnimation()`

Plays the jump animation and stops the running sound.

Source:

character.class.js, line 166

`jumpingSound()`

Plays the jump sound effect with adjusted volume.

Source:

character.class.js, line 198

`runningAnimation()`

Plays the running animation and starts the running sound.

Source:

character.class.js, line 175

Class: Chicken

Chicken()

`new Chicken()`

Creates a new chicken instance.

Source:

chicken.class.js, line 16

Members

`enableMovment :number`

Starts an interval that continuously moves the chicken to the left at its set speed. The interval ID for movement (private property)

Type:

- number

Source:

chicken.class.js, line 30

Methods

`animate()`

Starts an animation loop that plays the chicken's walking animation or dead state.

Source:

chicken.class.js, line 37

`enableAnimation()`

Starts an interval that plays the appropriate animation (walking or dead) based on the chicken's life points. The interval ID for animation (private property)

Source:

chicken.class.js, line 46

Class: Cloude

Cloude `(img)`

`new Cloude(img)`

Creates a new cloud instance.

Parameters:

Name	Type	Description
`img`	string	The path to the image for the cloud.

Source:

cloud.class.js, line 11

Methods

`animate()`

Starts an animation loop that continuously moves the cloud to the left.

Source:

cloud.class.js, line 20

Class: CollectableObjects

CollectableObjects (y, img)

`new CollectableObjects(y, img)`

Creates a new collectable object instance.

Parameters:

Name	Type	Description
`y`	number	The initial y-coordinate of the object.
`img`	string	The path to the image for the object.

Source:

collectable-objects.class.js, line 13

Class: EndScreen

EndScreen (img)

`new EndScreen(img)`

Creates a new end screen instance. - Loads the provided image for the end screen.

Parameters:

Name	Type	Description
`img`	string	The image path for the end screen.

Source:

end-screean.class.js, line 13

Class: Endboss

Endboss ()

`new Endboss()`

Creates a new endboss enemy. - Loads the default walking animation image. - Sets the starting position on the x-axis. - Loads walking, dead, hurt, and alerted animation images. - Starts the movement and animation loops.

Source:

endboss.class.js, line 42

Members

`animate`

Internal function that continuously plays the appropriate animation. - Checks if the endboss is dead, hurt, alerted, or in its normal walking state. - Plays the dead animation, stops movement, and updates the last image if dead. - Plays the hurt animation and a sound effect if hurt. - Plays the alerted animation if alerted. - Plays the walking animation otherwise.

Source:

endboss.class.js, line 67

`enableMovment`

Internal function that continuously moves the endboss to the left. - Moves the endboss leftward by its speed value.

Source:

endboss.class.js, line 55

Class: Keyboard

Keyboard ()

`new Keyboard()`

Creates a new keyboard instance and sets up event listeners for key presses and touches.

Source:

keyboard.class.js, line 10

Methods

`buttonPressEvent()`

Attaches event listeners for touch start/end events on directional and action buttons.

Source:

keyboard.class.js, line 18

`keyPressEvent()`

Attaches event listeners for key down/up events on keyboard keys.

Source:

keyboard.class.js, line 63

Class: Level

Level(enemies, clouds, backgroundObjects, statusBar, salsaBottles, coins)

```
new Level(enemies, clouds, backgroundObjects, statusBar,
salsaBottles, coins)
```

Creates a new level instance.

Parameters:

Name	Type	Description
`enemies`	Array.<MovebaleObject>	An array of enemy objects.
`clouds`	Array.<DrawableObject>	An array of cloud objects.
`backgroundObjects`	Array.<DrawableObject>	An array of background objects.
`statusBar`	StatusBar	The status bar object for the level.
`salsaBottles`	Array.<ThrowableObject>	An array of salsa bottle objects.
`coins`	Array.<DrawableObject>	An array of coin objects.

Source:

level.class.js, line 20

Class: SmallChicken

SmallChicken()

```
new SmallChicken()
```

Creates a new small chicken enemy. - Loads the default walking animation image. - Sets a random starting position on the x-axis. - Loads walking and dead animation images. - Sets a random movement speed. - Starts the movement and animation loops.

Source:

small-chicken.class.js, line 23

Members

`enableMovment`

Internal function that continuously moves the chicken to the left. - Moves the chicken leftward by its speed value.

Source:

small-chicken.class.js, line 36

Methods

`animate()`

Starts the animation loop for the small chicken.

Source:

small-chicken.class.js, line 43

`enableAnimation()`

Internal function that continuously plays the appropriate animation. - Checks if the chicken is dead using the `isDead` function. - If dead, plays the dead animation, stops movement, and adjusts size/position. - If alive, plays the walking animation.

Source:

small-chicken.class.js, line 53

Class: StatusBar

StatusBar`(x, imgs, precent)`

`new StatusBar(x, imgs, precent)`

Creates a new status bar instance. - Loads the default image from the first element of `imgs`. - Stores all image paths in `IMG_STATS`. - Sets the initial position (x) and loads all images. - Sets the initial percentage value.

Parameters:

Name	Type	Description
`x`	number	The initial x-coordinate of the status bar.
`imgs`	Array.<string>	An array of image paths for different status levels.
`precent`	number	The initial percentage value for the status bar.

Source:

status-bar.class.js, line 20

Methods

`resolveImageIndex() → {number}`

Determines the index of the image to be displayed based on the current percentage value. - Returns 0 for 0% status. - Returns 1 for 1-20% status, and so on (up to 5 for 81-100% status).

Source:

status-bar.class.js, line 46

Returns:

- The index of the image to be used based on the percentage.
Type

number

`setPercentage(percentage)`

Sets the new percentage value for the status bar and updates the image accordingly.

Parameters:

Name	Type	Description

`percentage`	number	The new percentage value (0-100).

Source:

status-bar.class.js, line 33

Class: ThrowableObject

ThrowableObject(x, y)

`new ThrowableObject(x, y)`

Creates a new throwable object at the specified position. - Loads the default image (`salsa_bottle.png`). - Loads additional images for throwing and splash animations. - Sets the initial position (x, y). - Starts the throwing motion.

Parameters:

Name	Type	Description
`x`	number	The initial x-coordinate of the object.
`y`	number	The initial y-coordinate of the object.

Source:

throwabel-object.class.js, line 34

Members

`animateTrow`

Internal function that continuously plays the throwing animation. - Plays the throwing animation frame from the `IMG_TROW` property.

Source:

throwabel-object.class.js, line 76

`throwforward`

Internal function that continuously updates the object's position during the throw. - Checks if the object is still above ground (considering a specific force of 370). - If above ground, moves the object forward by 12 pixels. - If not above ground, stops movement and triggers the `salsaHit` function. - Clears the `spalsh` interval (for a splash animation).

Source:

throwabel-object.class.js, line 60

Methods

`salsaHit()`

Called when the object hits the ground. - Plays the splashing sound effect with lower volume. - Deactivates the object. - Stops movement and clears throwing and animation intervals. - Starts a new interval to play the splash animation.

Source:

throwabel-object.class.js, line 87

`throw()`

Starts the throwing motion of the object. - Applies gravity with a specific force (370). - Plays the throwing sound effect.

Source:

throwabel-object.class.js, line 48

Class: World

World`(canvas, keyboard)`

`new World(canvas, keyboard)`

Creates a new World instance.

Parameters:

Name	Type	Description

canvas	HTMLCanvasElement	The canvas element to use for rendering.
keyboard	Keyboard	The keyboard object for handling player input.

Source:

world.class.js, line 27

Methods

addObjectsToMap()

Helper function to add a collection of objects to the map (presumably for drawing). - Iterates over each object in the collection and calls `addToMap` on it.

Source:

world.class.js, line 208

addToMap(mo)

Draws a single object on the game canvas. - Flips the object's image if the `otherDirection` flag is set. - Calls the object's `draw` method to render it on the canvas. - Flips the image back if previously flipped.

Parameters:

Name	Type	Description
mo	DrawableObject	The moveble object to be drawn.

Source:

world.class.js, line 222

backgroundMusic()

Plays or pauses the background music based on the global `music` variable.

Source:

world.class.js, line 64

`constantRepeat()`

Starts a repeating loop for core game logic updates. - Checks for collisions. - Throws salsa bottles based on player input and cooldowns. - Checks for game over condition. - Manages background music playback.

Source:

world.class.js, line 52

`draw()`

The main game loop function responsible for drawing and updating the game world. - Clears the canvas. - Applies camera translation. - Draws movable objects on screen. - Draws static objects on screen. - Cancels camera translation. - Calls `drawAnimation` to request another animation frame.

Source:

world.class.js, line 156

`drawAnimation()`

Requests an animation frame for the next draw cycle. - Schedules a callback function (`draw`) to be called by the browser for the next animation frame.

Source:

world.class.js, line 197

`flipImg(mo)`

Flips the object's image horizontally on the canvas by manipulating the context. - Saves the current canvas state. - Translates the context by the object's width. - Scales the context horizontally by -1 (mirroring). - Mirrors the object's x-coordinate for correct positioning.

Parameters:

Name	Type	Description
`mo`	DrawableObject	The moveble object to be flipped.

Source:

world.class.js, line 241

`flipImgBack(mo)`

Resets the image flipping applied in `flipImg` and restores the original state. - Multiplies the object's x-coordinate by -1 to undo mirroring. - Restores the canvas context to its previous state before flipping.

Parameters:

Name	Type	Description
`mo`	DrawableObject	The moveble object that was flipped.

Source:

world.class.js, line 255

`gameEnded()`

Stops the game loop and animations when the game ends. - Clears the `keepChecking` interval. - Cancels the `animationRequest` after a short delay (500ms). - Sets `gameEnde` to true to indicate game ended state.

Source:

world.class.js, line 108

`lost()`

Called when the game is lost. - Plays the losing sound. - Creates a new end screen object. - Ends the game.

Source:

world.class.js, line 94

`movableObjectsOnScreen()`

Draws all movable objects currently within the viewport. - Calls `addObjectsToMap` for various object categories (background, clouds, etc.).

Source:

world.class.js, line 169

`setWorld()`

Initializes references between the world and its components (character, collisions).

Source:

world.class.js, line 39

`staticObjectsOnScreen()`

Draws all static objects on screen. - Applies camera translation (negative) before drawing status bar and end screen. - Cancels camera translation afterwards.

Source:

world.class.js, line 184

`throwBrake()`

Checks if enough time has passed since the last throw to allow another throw. - Calculates the time difference between the current time and the last throw time. - Returns true if at least 200 milliseconds have passed.

Source:

world.class.js, line 142

`throwObjects()`

Throws a salsa bottle if certain conditions are met. - Checks for keyboard throw key press, sufficient salsa bottles, and a throw time brake. - Creates a new `ThrowableObject` instance and adds it to the world. - Decrements salsa bottle count and percentage. - Updates the status bar with the new salsa percentage. - Sets the last throw time for the throw brake. - Calls `character.idleEnd()` to handle character animation.

Source:

world.class.js, line 125

`won()`

Called when the game is won. - Plays the winning sound. - Creates a new end screen object. - Ends the game.

Source:

world.class.js, line 81