Ragamuffin Technical Documentation

Reid Cook, Scott Serafin, Brandon Price, Elton Gbollie, Luis Cruz Quispe

Section 1 API Calls

Function	Method/Endpoint	Responses	Description
registerUser(username, password)	POST http://localhost:3000/user	200, 400, 500	Registers a new user by sending a POST request to the server.
loginUser(username, password, errorText)	POST http://localhost:3000/login	200, 500, 404, 401	Logs in a user by sending a POST request to the server
sendScoreToServer(score, username)’	POST http://localhost:3000/update level1score	200, 500, 404	Sends player score for level 1 To the server
updateTotalScore(username)	PATCH http://localhost:3000/user/total-score	200, 500, 404, 400	Updates user’s total score on the server
sendTopScores()	GET/POST http://localhost:3000/leaderboard/top https://eope3o6d7z7e2cc.m.pipedream.net	200, 400	This function gets the top five scores from our database and sends then to the public API
sendScoreToServer2(score, username)	POST http://localhost:3000/update level2score	200, 500, 404	Sends player score for level 2 To the server

sendScoreToServer3(score, username)	POST http://localhost:3000/update level3score	200, 500, 404	Sends player score for level 2 To the server
leaderboard1.js API call	GET http://localhost:3000/leaderboard /level1	200, 400	This fetches the top 5 scores for level 1
leaderboard2.js API call	GET http://localhost:3000/leaderboard /level2	200, 400	This fetches the top 5 scores for level 2
leaderboard3.js API call	GET http://localhost:3000/leaderboard /level3	200, 400	This fetches the top 5 scores for level 3

Section 2 Database Outline

Database Type: SQLite3

Database File: myGameDB.db

Table: users

Columns

Id: this is the primary key, so each user has a unique id.

Username: username used for logging into the game

Password: password used for logging into the game

Level_1_score: holds the highest score a specific user has on level 1

Level_2_score: holds the highest score a specific user has on level 2

Level_3_score: holds the highest score a specific user has on level 3

Section 2.1 Database Description

initializeDb.js is run before the player can start playing the game. This script creates the database myGameDB.db, and creates a new users table in SQLite. This table is used to store users data, so they can login and save their high scores.

Section 3 Class Diagram Outline

Class Name	Description	Methods
login.js	Allows the user to login and register into the game. Upon login goes to level select scene. loginUser(username, password, errorText)	preload() create()
level-select.js	Allows user to choose a level, or customize their character. Can go to level 1, level 2, level 3, or customize scenes. Upon choosing a level, the value of the most recently selected color is passed into the level scene called.	preload() create() startGame() startGame2() startGame3() enterShop()
level1. Js, level2,js, level3,js	These are the scenes that start each level respectively. Upon completing the level, you go to a leaderboard scene. These scenes use the color variable passed to it to determine what color the player should be.	preload() create() collectStar(player, star)
leaderboard1.js, leaderboard2.js, leaderboard3.js	These scenes display the leaderboard of each level respectively. They allow the player to either go back to the main menu, or to restart the current level.	preload() create()

shop.js	This scene allows the player to view a list of all the different colors you can play as. Upon clicking a color, it passes the value of the color back to the level select screen with the variable name “color”.	preload() create() chooseRed() chooseGalaxy() chooseBlue() chooseYellow() chooseParody() chooseGreen()

Section 3.1 Class Diagram Outline for Objects

Game Object	Description	Methods
Map	This object is a tilemap imported from Tiled. This is then used to add tileset layers on top of the map.	addTilesetImage(tileset, tiles) createLayer(layerName, tileset, 0, 0)
Player	The player object is added into every level scene. A sprite and gravity are both added to the game object at the start through phaser. Button presses from the user impact the physics of the player as well, doing actions such as jumping, and dashing.	setVelocityX(velocity) setVelocityY(velocity) anims.play(animation, state) body.onFloor() setGravityY(gravity)
portal	This portal object is placed at the end of every level scene. A sprite is applied to the portal, as well as a collision. When the player collides with the portal, they are taken to the associated leaderboard screen.	setCollision(collision) create(xCoord, yCoord, name)

floor	This tileset is used for the ground. Using the tilemap, a layer is created using the sprites from floor. Collision detection is applied to this layer, so the player is able to run over top of it.	setCollision(collision) stars.setCollisionByExclusion([collision])
spikes	This tileset is used for the spikes. Using the tilemap, a layer is created using the sprites from spikes. Collision detection is also applied to this layer, so it knows when the player lands on it. When the player lands on a spike, the scene is restarted.	setCollision(collision) stars.setCollisionByExclusion([collision]) forEachTile((tile), function)
stars	This tileset is used for the stars. Using the tilemap, a layer is created using the sprites from stars. Collision detection is applied to this layer, so it knows when the player collects a start. When the player runs into a star, the collision box of the star is disabled and the sprite turns invisible. Additionally the star collection count increases.	setCollision(collision) stars.setCollisionByExclusion([collision]) forEachTile((tile), function)