TECHNICAL DESCRIPTION - BLOBA IN SPACE

Authors

Name	*Position*	*Email*
Jessie Chen	Technical Writer	jesch136@student.liu.se

Updates

Sprint	*Version*	*Date*	*Modification*	*Person in charge*	*Verified by*
3	0.9	2014-12-04	First version	Jessie Chen	David Hedin
3	1.0	2014-12-09	Updated after inspection	Jessie Chen	Jakob Boman

Flow of calls in the code

This is a brief description of the flow of the calls in the code for developers to read before reading comments and code. This is to give a general understanding of which function is called where and in what order. For detailed descriptions of all functions go to the GitHub application workspace.

Game states

There are a total of 10 different game states, total of 5 for the Menu and 5 for the different game phases. Depending on the game state the remote buttons will enable different functions. All game states can be found in the custom_onkey.lua that is comprised of one function, the **onKey()**.

Menu

The application starts with the function **story()** that navigates the user to the actual main menu. The main menu navigates in the "menu" game state and enters the following functions that all can be found in menu.lua:

- **Avatar()**
- **highscore()**
- **tutorial()**
- **Level()**

Avatar() initializes the page to customize the user's avatar. Functions used are:

- **chooseParam()** that navigates between the changeable parameters top, middle and bottom
- **chooseItem()** that navigates between the items available for each parameter
- **unlock()** is a function that unlocks the items along the levels cleared in the game
- **drawAvatarScreen()**

highscore() initializes the highscore page that shows the total amount of stars collected by the user. The only other function related to it is the **drawHighscoreScreen()**.

tutorial() is the sole function handling the tutorial pages.

Level() initializes the page to choose levels. Functions that are used in the level menu are: **chooseLv()** and **drawLevelScreen()**
This page shows the 4 levels and the stars the user has earned for each level.
The levels as well as the avatar items are unlocked throughout clearing the levels. The profile values are saved with the **saveProfil()** and to use the values they are called with the **loadProfil()**. The parameters saved are the amount of stars earned each level, the total amount of stars earned and the highest level reached.

Game

After choosing an available level to play the game starts with the specified difficulty settings with the function **start()**. The antagonistic slobs are handled in slobs.lua.

- **createSlob()** initializes the slobs that are shown on the screen with their respective properties, the boss is separately handled in the same functions through if-statements

- **drawSlobs()** draws the slobs and also handles the on and off of the boss shield.
- **getRandomLevelSlob()** bases the random spawning of slob levels on the current level of the playable character
- **makeHappy()** is called after a successful collision with a slob and the slob is returned to its original state as a blob.

The playable character Bloba is similarly handled in blob.lua that also include its movements, summarized in **moveBlob()**, and its growing and shrinking properties, **growBlob()** and **decreaseBlob()**.

Collisions

The game play is all about colliding with the right character. This is handled in the collision.lua.

- **detectCollision()** detects collisions based on **checkIfOverlaping()**
- **beginColliding()**
- **isColliding()**
- **endColliding()**

The three states of collision enables the blinking properties that insinuate being hurt after a bad collision; with a higher numbered slob.

Game Over & Level Complete

Game is over when Bloba reaches level 0 and this is detected also in the **decreaseBlob()**. When it reaches 0 the function calls **gameOver()** in game.lua that enters the "game_over" game state.
When Bloba reached the set level and the level is complete, it is detected in **beginColliding()** and the **levelCompleted()** from the game.lua is called. High scores concerning stars and level are saved to the profile in the **levelCompleted()**. After completing the last level the ending phase of the game is initialized through the dance function, which shows a cute dance as an ending to the user.