Design Report

(COS10009)

Name: Brandon Bao Quan LAI

ID: 102787664

# **Table of Contents**

[**Table of Contents** 2](#_Toc167299649)

[**1.0 Overview of the program** 3](#_Toc167299650)

[**2.0 Structured chart of the program** 4](#_Toc167299651)

[**3.0 Description of core program’s functionality and how it works** 5](#_Toc167299652)

[**4.0 Data Dictionary of the program** 6](#_Toc167299653)

# **1.0 Overview of the program**

The goal of the program is to let users play a game of bouncing ball and train their reflexes. It also lets users challenge their own scores which is the time taken to complete the game. The game also aims to let the user enjoy their time and experiment the C language coding, as well as syntaxes of C language.

# **2.0 Structured chart of the program**

Int (main)

initGameObjects()

currentState == GAME OVER

renderGameOver()

handleEvents(&quit)

close()

Program stopped/ quit

currentState == PLAYING

render()

update()

handleEvents(&quit)

currentState == Menu

renderMenu()

handleMenuEvents(&quit)

True

false

loadMedia()

True

false

Init()

# **3.0 Description of core program’s functionality and how it works**

The program that is created is a bouncing ball game. The program first starts with a start menu that uses text file from arial.ttf to render the text, called ‘Click any button to start’. This will be loaded if the library and the SDL formats are correct, and all the files needed exist. The function used to initialize this is ‘renderMenu()’ and ‘handleMenuEvents(&quit)’. The ‘renderMenu()’, presents the word on the menu if the font exists and sets it to start the game when any button on keyboard is pressed, while the ‘handleMenuEvents(&quit)’ function, handles the quit effectively if x button is pressed to quit the program.

After the user has pressed a key on the keyboard, it automatically jumps into the game. The game consists of functions such as ‘update()’, ‘render()’, ‘handleEvents(&quit)’. The ‘render()’ function renders the blocks, ball and presents the best time as well as the time ticking. The ‘update()’ function makes the ball move, destroys blocks, collides with the paddle and the screen, checks the health, recalls ball if one health is depleted, and saves the best time if it is completed. The ‘handleEvents(&quit)’ function allows the program to run and quit effectively.

The game over section contains the function called ‘renderGameOver()’, ‘handleEvents(&quit), and ‘close()’. The ‘renderGameOver()’ function presents the end screen and the words at the game over section. The ‘handleEvents(&quit)’ function effectively executes the quit function. The ‘close()’ function prevents memory leaks, cleans up the program and closes it effectively.

Additional functions used to make this program run is ‘renderText()’ which renders the text for the time. The other functions are ‘areAllBlocksDestroyed()’, ‘loadBestTime()’, ’saveBestTime()’, ‘initGameObjects()’, ‘loadMedia()’, and init(). The ‘areAllBlocksDestroyed()’ function checks if all the blocks is destroyed. The ‘loadBestTime()’ function reads the best time from a text file. The ‘saveBestTime()’ function writes the best time into the text file. The ‘initGameObjects()’ function initializes the game function. The ‘loadMedia()’ function loads the ttf font file and assigns it to the variable font. The ‘init()’ function checks the library to make sure it is functioning.

# **4.0 Data Dictionary of the program**

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Variable Type | Description | Value |
| SCREEN\_WIDTH | define | Width of the screen | 800 px |
| SCREEN\_HEIGHT | define | Height of the screen | 600 px |
| BLOCK\_ROWS | define | Rows of the blocks | 5 rows |
| BLOCK\_COLUMS | define | Columns of the blocks | 10 columns |
| BLOCK\_WIDTH | define | Width of each block | 64 px |
| BLOCK\_HEIGHT | define | Height of each block | 20 px |
| PADDLE\_WIDTH | define | Width of the paddle | 100 |
| PADDLE\_HEIGHT | define | Height of the paddle | 20 |
| BALL\_SIZE | define | Size of the ball | 15 |
| Ball.x | Float in a struct | X coordinates for the ball variable used in operators | Undefined/ Unassigned value for operators |
| Ball.y | Float in a struct | Y coordinates for the ball variable used in operators | Undefined/ Unassigned value for operators |
| Ball.velX | Float in a struct | Horizontal Velocity of the ball variable used in operators | Undefined/ Unassigned value for operators |
| Ball.velY | Float in a struct | Vertical velocity of the ball variable used in operators | Undefined/ Unassigned value for operators |
| Paddle.x | Integer in a struct | X coordinates for the paddle used in operators | Undefined/ Unassigned value for operators |
| Paddle.y | Integer in a struct | Y coordinates for the paddle used in operators | Undefined/ Unassigned value for operators |
| Paddle.width | Integer in a struct | Width of the paddle used in operators | Undefined/ Unassigned value for operators |
| Paddle.height | Integer in a struct | Height of the paddle used in operators | Undefined/ Unassigned value for operators |
| Block.y | Integer in a struct | Y coordinates for the block used in operators | Undefined/ Unassigned value for operators |
| Block.x | Integer in a struct | X coordinates for the block used in operators | Undefined/ Unassigned value for operators |
| Block.active | Boolean in a struct | Existence of each block |  |
| SDL\_Window\*window | Assignment of variable | Variable for window | NULL |
| SDL\_Render \* renderer | Assignment of variable | Variable for renderer | NULL |
| TTF \_Font \*font | Assignment of variable | Variable for font | NULL |
| Ball ball | Assignment of variable from struct | Variable for ball as a type of Ball | Values inside the struct Ball |
| Paddle paddle | Assignment of variable from struct | Variable for paddle as a type of Paddle | Values inside the struct Paddle |
| Block blocks [BLOCK\_ROWS] [BLOCK\_COLUMNS] | 2D array | blocks as a type of Block under defined values as array | [5,10] |
| health | Integer | Health of the user in the game | 3 |
| startTime | 32 bit unsinged integer | Starting time of the game | 0 |
| elapsedTime | 32 bit unsinged integer | Time since the SDL library ran | Undefined variable |
| bestTime | 32 bit unsigned integer | Time record in text file | 4981278 |
| GameState.MENU | Struct variable | To state the state of the game | MENU |
| GameState.PLAYING | Struct variable | To state the state of the game | PLAYING |
| GameState.GAME\_OVER | Struct variable | To state the state of the game | GAME\_OVER |
| GameState currentState | Assignment of variable for struct | currentState is a variable for enum GameState | Values inside struct GameState |
| timeText | char | Ticking time of the game | Variable is used in operators |
| Ball | struct | Name for struct Ball | x, y, velX, velY |
| Paddle | struct | Name for struct Paddle | x, y , width, height |
| Block | struct | Name for struct Block | x, y, active |
| GameState | struct | Name for struct GameState | MENU  PLAYING  GAME\_OVER |