# IT PAT Design Document

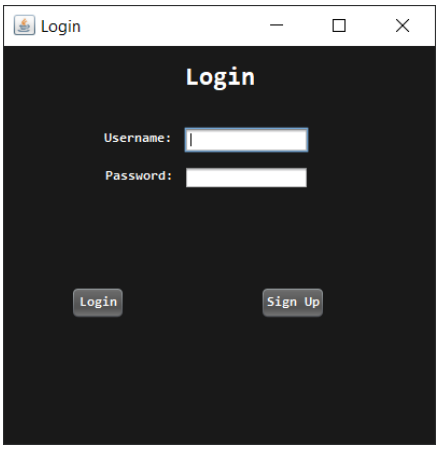
## Lost

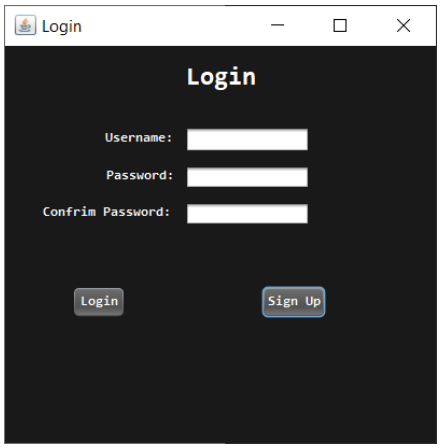
Kian Anderson

## Table of Contents

# 1. User Interface Design

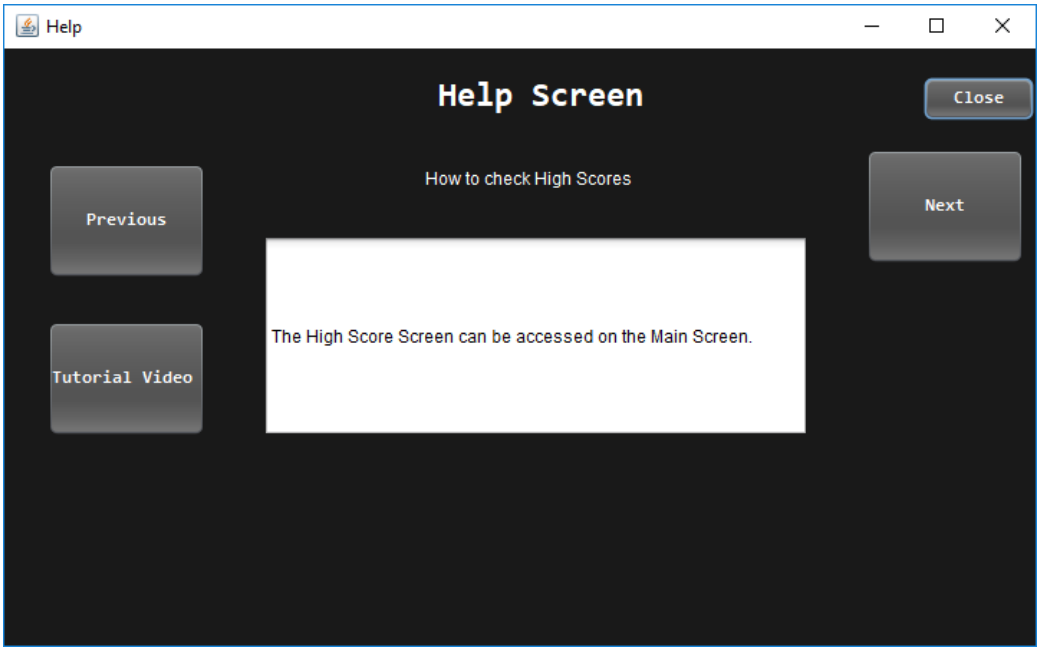
## 1.1 LogIn Screen




| | |
|---|---|
| Description | This screen is used to allow a user acess to the rest of the game and, if they already exist in the database they can Sign In, otherwise they will press the Sign Up button and create themselves in the database. |
| Data | - There will be a title "Login" at the top of the screen<br>- There will also be 1 Text Field and 2 Password Fields depending on whether the user is Signing In or Signing Up.<br>- There are 2-3 Labels, "Username", "Password" and "Confirm Password" which are used to indicate the text field they are next to. |
| Actions | **Log In**<br>Button for user to check username and password against the text file and if the user exists and the password is correct, the user will be taken to the Level Select Screen.<br>**Sign Up**<br>Button for user to create a new account, this will display an extra password field and a label "Confirm Password" when pressed, when the fields are completed by the user the two password fields are checked to see if they are identical, if they are a new user will be created in the text file. |

# 1.2 Help Screen

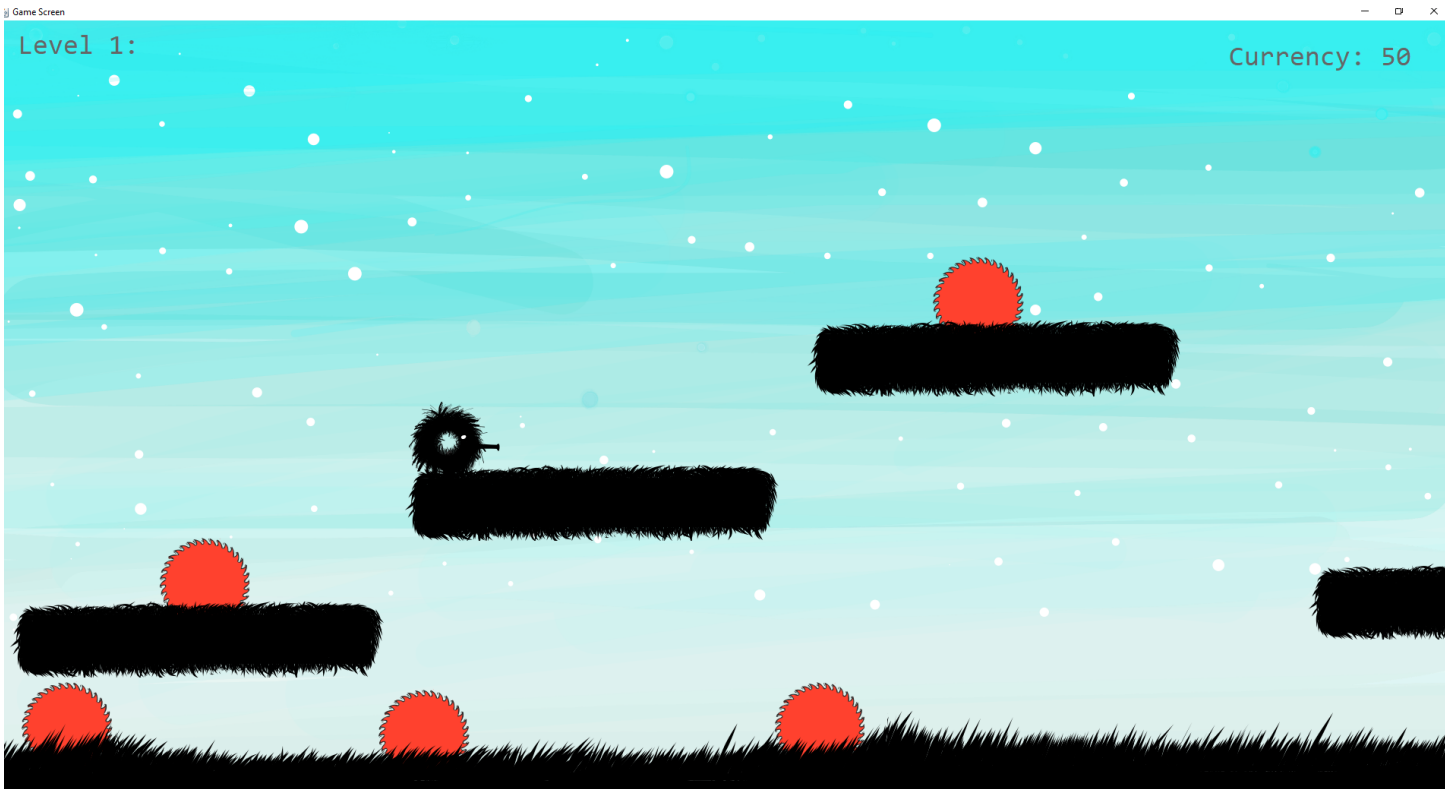



| | |
|---|---|
| Description | This screen will display 10 relevant help topics from the text file that can be chosen by pressing the next and previous buttons. There is also a tutorial video that can be played to help the user understand the objective of the game better. |
| Data | - This screen will display information regarding a topic in a Text Area.<br>- Text will be displayed on the text area from the text file<br>- By pressing next and previous the user will see different help topics.<br>- There are 10 help topics that will help the user in any questions they may have about the game. |
| Actions | **Next**<br>This button when pressed will display the next help topic and information in the text area.<br>**Previous**<br>This button when pressed will display the previous help topic and information in the text area.<br>**Tutorial Video**<br>This button when pressed will display a tutorial video that will explain the basic concepts of the game to the user.<br>**Close**<br>This button will close the Help Screen and open the Main Screen. |

# 1.3 Best Time Screen

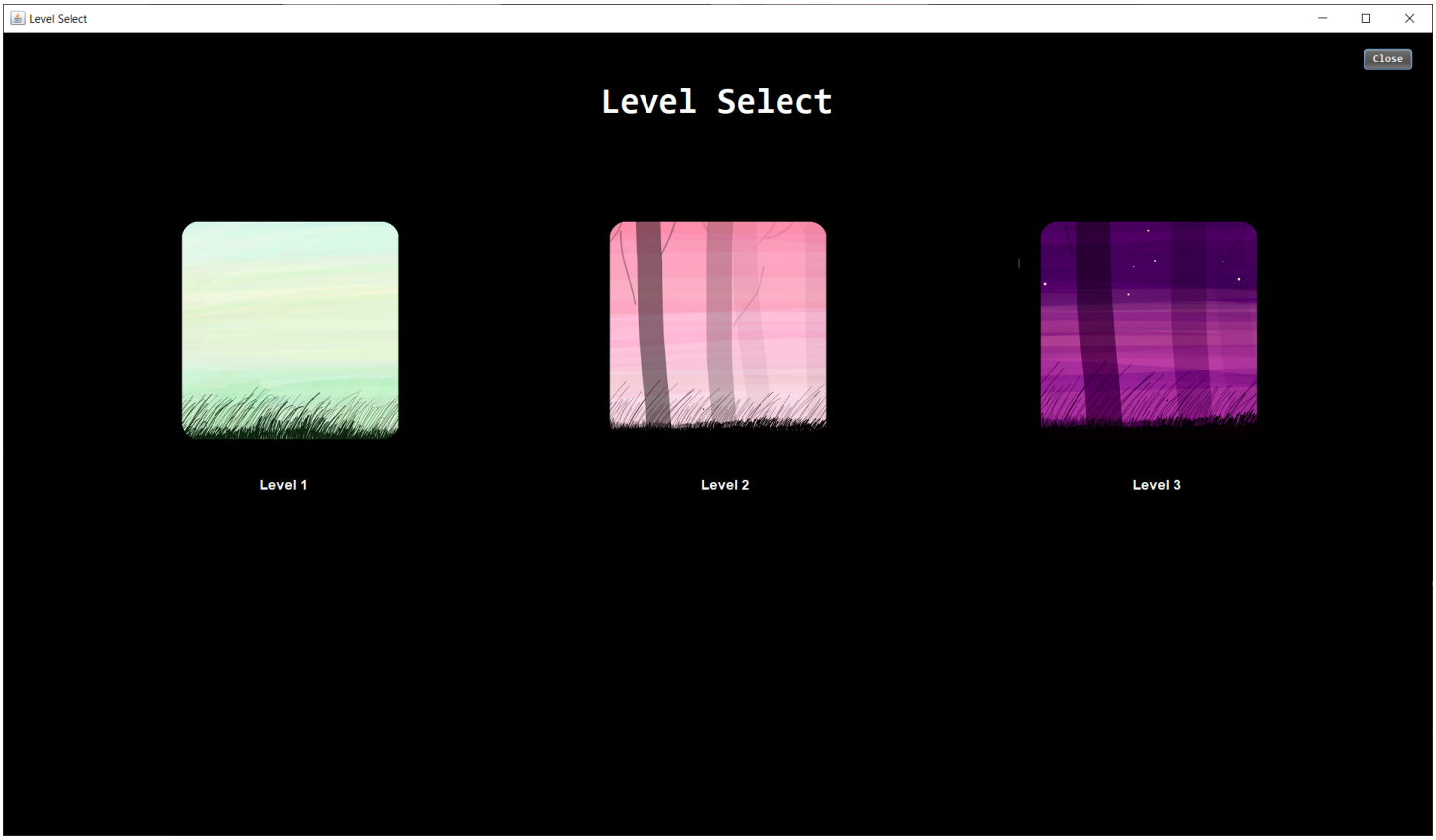

| | |
|---|---|
| Description | This screen will display the best time the user has achieved on each of the 3 levels in the game in a Text Area as well as the best time achieved for each level in the database as a comparison to what the user achieved. This screen will also have a button to close the screen. |
| Data | - This screen will display the 3 best times of the user in each level in a Text Area that will read data from the database<br>- There will be a Label "Best Time Achieved by" + Username which will be personalized to the user by their username. |
| Actions | **Close**<br>This button will close the Best Time Screen and open the Main Screen. |

# 1.4 Game Screen



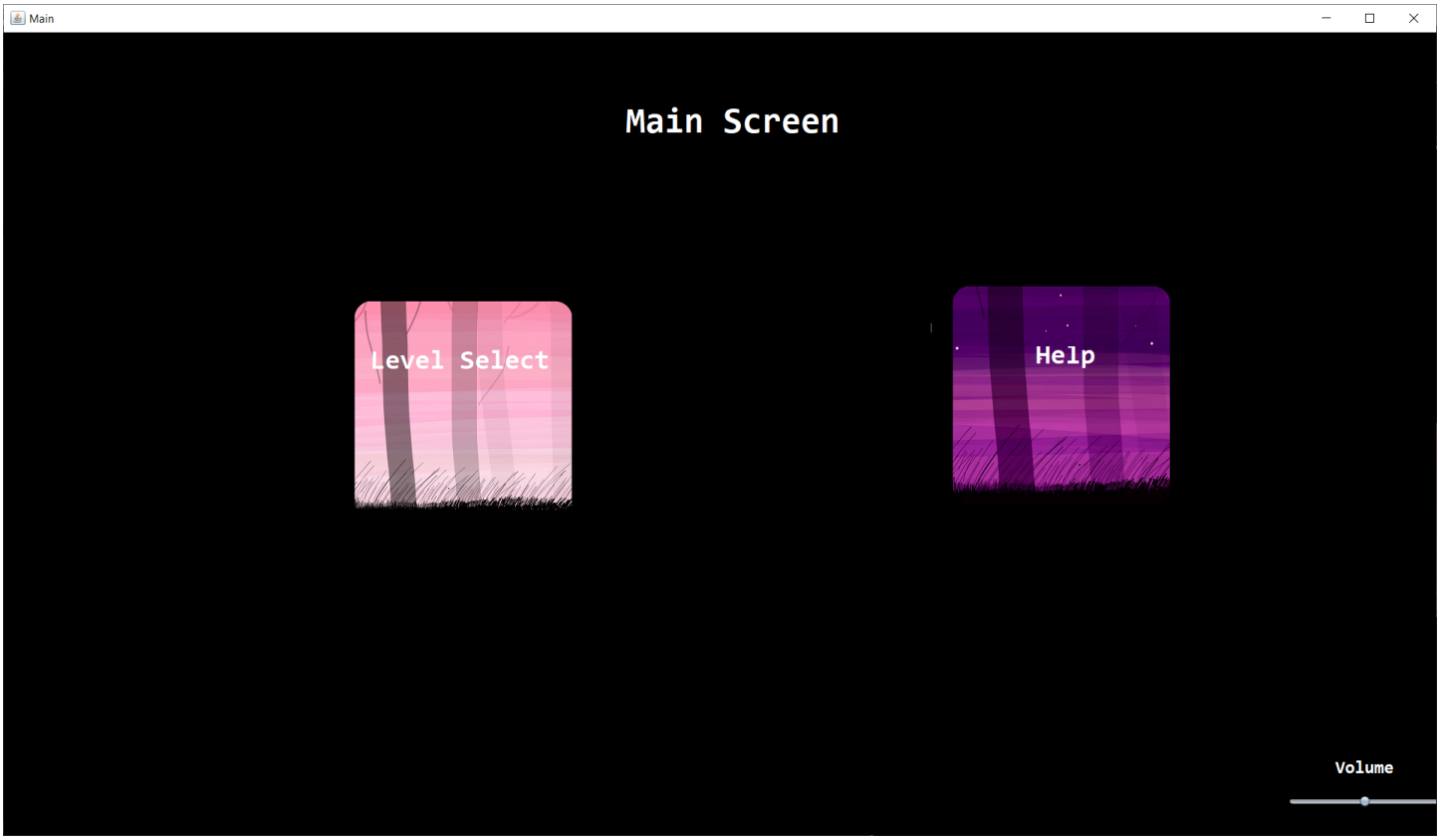

| | |
|---|---|
| Description | This screen will display two Labels, both with an icon which will create the character and the background. The character will be controlled by user input on the WASD keys. |
| Data | - This screen will consist of a character at the center of the screen which is actually a Label with a icon.<br>- The Character will move horizontally across the screen which will be determined by the user input and vertically which will be determined by the floor and object collision.<br>- The background is static and is a Label with an icon and will fill the entire screen.<br>- There will be a label on the bottom of the screen which will act as a floor. |
| Actions | **Space**<br>**Space** will make the character jump.<br><br>**WAD**<br>**A** will make the player move Left.<br>**D** will make the player move Right |

# 1.5 Level Select Screen




| | |
|---|---|
| Description | This screen is used for the player to select which level they would like to start. |
| Data | - The screen will consist of a Panel that is the background that is set to the colour black.<br>- There will be a Label "Level Select" at the top of the screen.<br>- There are 3 labels under each button to indicate which level the button will take the user to. |
| Actions | **Level 1**<br>This button will take the user to Level 1.<br>**Level 2**<br>This button will take the user to Level 2.<br>**Level 3**<br>This button will take the user to Level 3.<br>**Close**<br>This button will close the Level Select Screen and open the Main Screen |

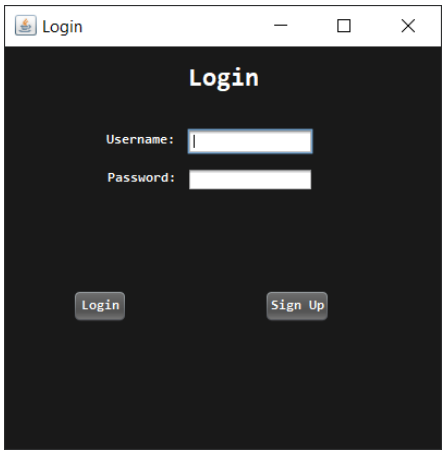
# 1.6 Main Screen

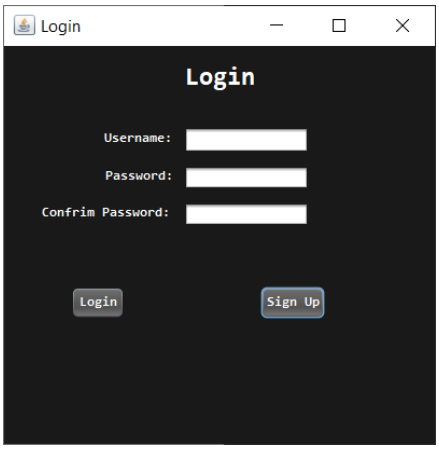



| | |
|---|---|
| Description | This screen is the screen the user will see after they have logged in, it will consist of a slider to adjust the volume of the game and a button to view the best time screen as well as a button to log out. |
| Data | - There is a Label "Main Screen" at the top of the screen.<br>- There is a panel set to the colour black to create a backrground colour. |
| Actions | **Audio Slider**<br>This slider will adjust the volume of the music in relation to the position of the slider.<br>**Best Time**<br>This button will take the user to the best time screen.<br>**Log Out**<br>This button will take the user back to the Login Screen. |

# 2. Sequencing and Algorithms

## 2.1 Log In Screen



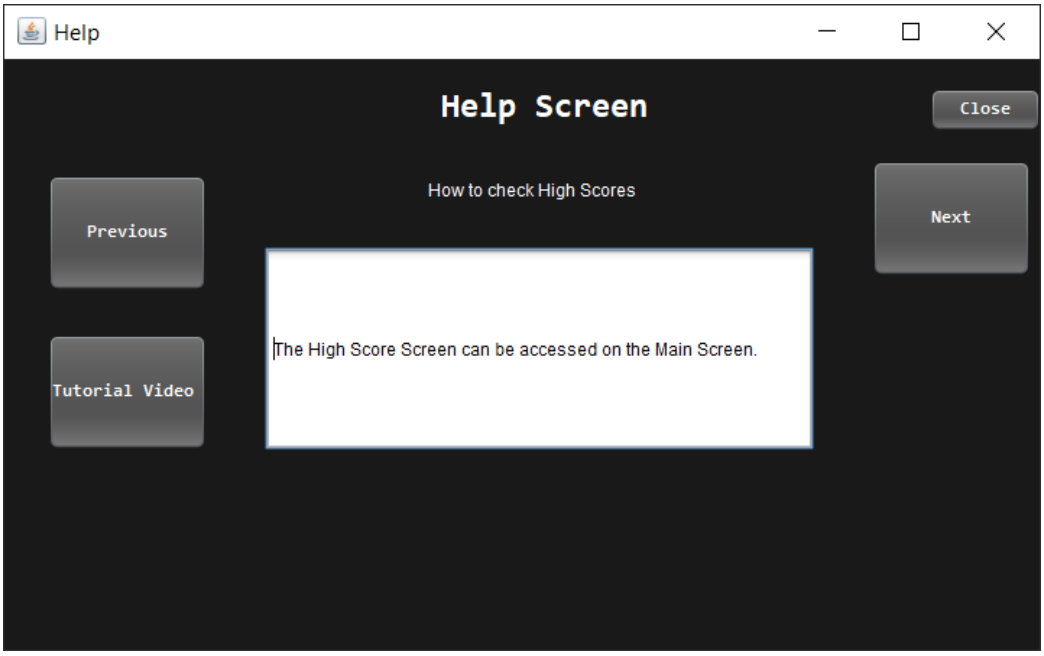

**Login:**

```
if btnLogin pressed
    while (Scanner has next)
        if(line contains username)
            exists ⟵ true
        end if
    end while

    if (exists is true)
        password ⟵ password from user.txt for username
        if password equals password entered
            mainScreen set visible
        end if
    end if
end if
```

**Sign Up:**

```
if btnSignUp pressed
    lblConfirmPassword set Visible
    txtConfrimPassword set visible
    if password equals confirmPassword
        addUser(name, password)
    end if
end if
```

# 2.2 Help Screen




**Previous:**

if btnPrevious is pressed
    display previous topic
end if

**Next:**

if btnNext is pressed
    display next topic
end if

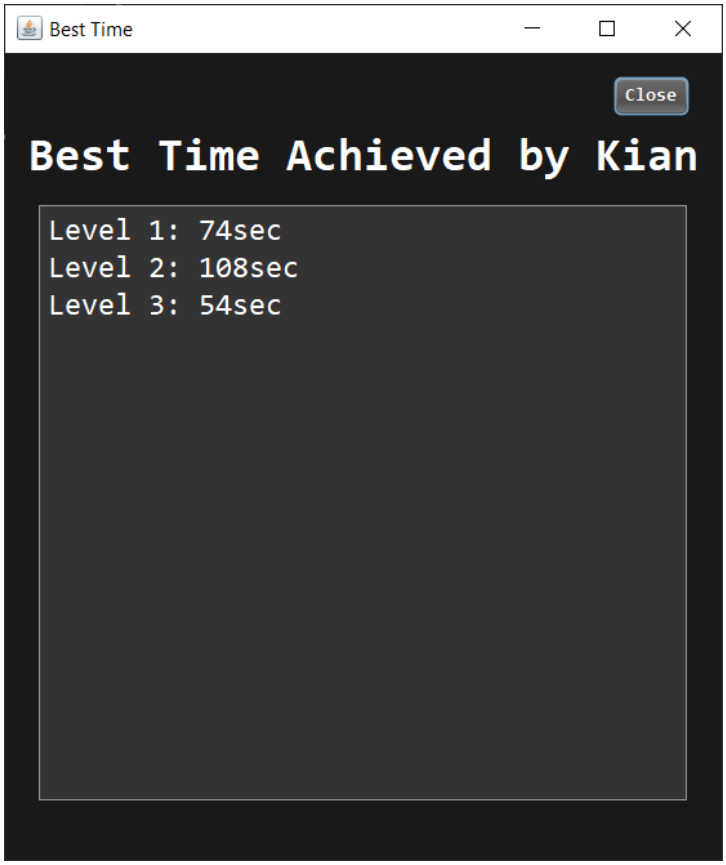
**Tutorial Video:**

if btnTutorial is pressed
    display tutorial video
end if

**Close:**

if btnClose is pressed
    close help screen
end if

# 2.3 Best Time Screen




**Best Time Screen:** (opens when user presses Best Time Screen button on Main Screen)

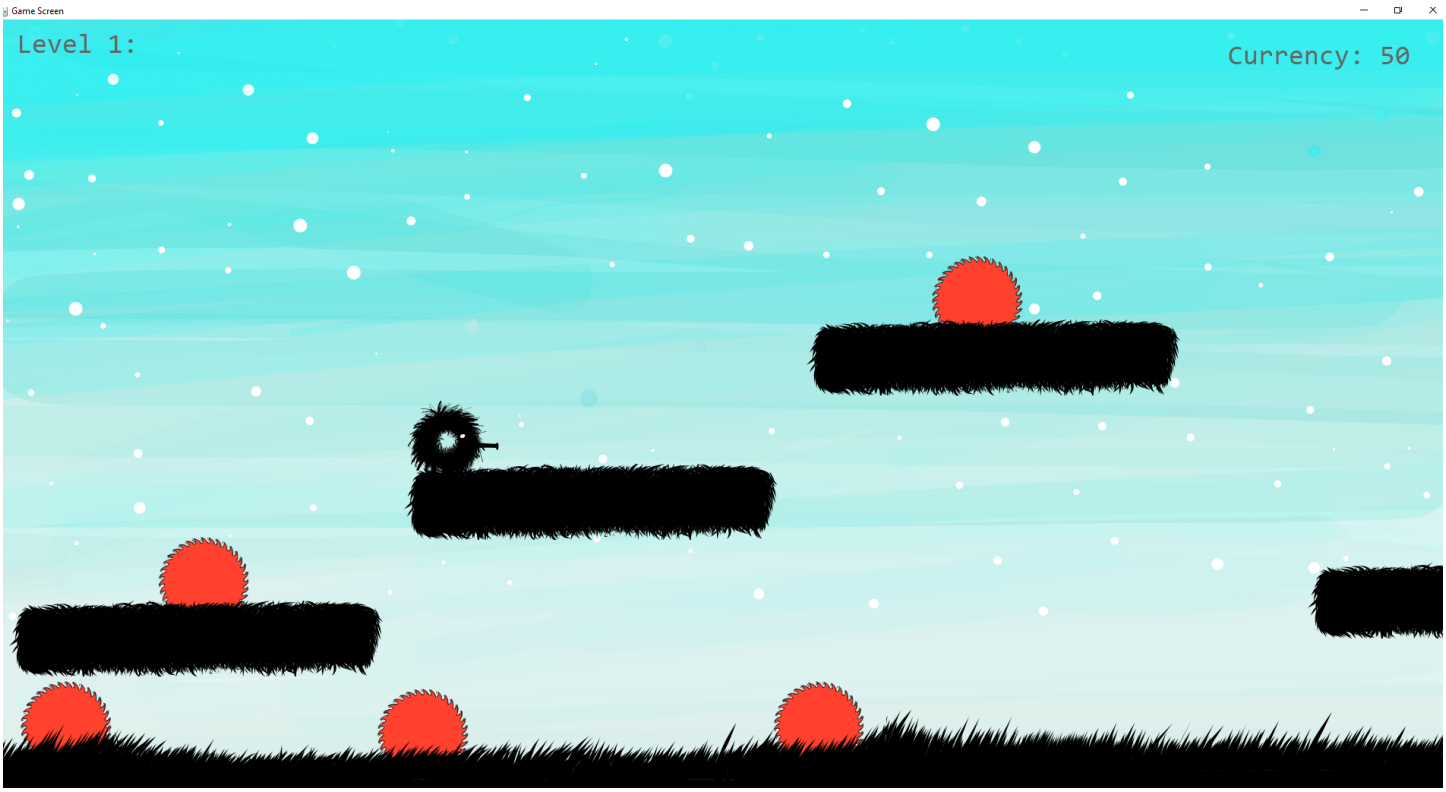
display "best time achieved by" + user

display "level 1: 74sec next line
level 2: 108sec next line
level 3: 54sec"

**Close:**

if (btnClose is pressed)
close Best Time Screen
end if

# 2.4 Game Screen




time ⟵ get current time in milliseconds

**keyListener**
keyPressed
if (Right key is pressed)
character direction ⟵ right
set character icon to right character image
end if

if (Left key is pressed)
character direction ⟵ left
set character icon to left character image
end if

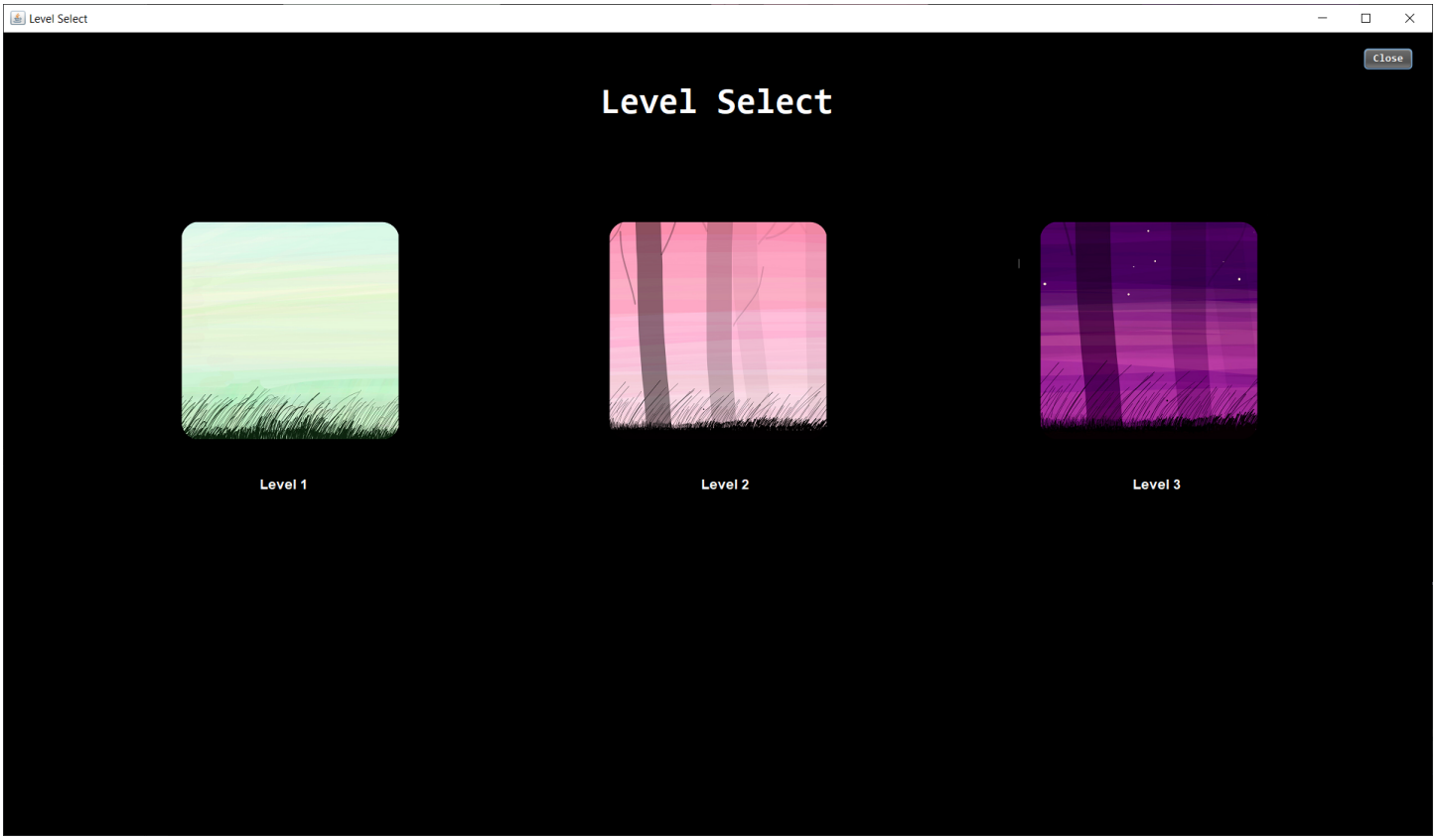
if (Up Key is pressed and player jumping is false)
jumping ⟵ true
set timer for jumping so character jumps for 350 milliseconds
jumping ⟵ false
falling ⟵ true
if (character collides with floor)
falling ⟵ false
end if
end if

```
        end keyPressed


        keyReleased
            if (Right key is released or Left Key is released)
                player stop
            end if
        end keyReleased

end keyListener


refreshLocations (x : JLabel)
        x ⟵ player direction
        set position of x to 5 units of character direction
end refreshLocations


resetLocations (x : JLabel)
        i ⟵ initialPosX - lastPosX
        set position of x to current position - i
end resetLocations


labelsArr (i : int) : JLabel
        labels[] ⟵ obstacle1; platform1; ... (every object on screen)
        return labels[i]
end labelsArr


if (character collided with chest)
        addCurrency(50)
        run chest animation
        dispose of chest
end if


if (character collides with obstacle)
        kill character
        set character location to start of level
end if


if (character collides with finish object)
        stop user input
        addCurrency(100)
        dispose screen
        go to level select screen
end if
```

```
for

thread run
    time ⟵ get current time in milliseconds
    while (game is running)
        wait 10 milliseconds

        if (foreground x position ⟵ initial position)
            if (character direction is right)
                character move right
            end if

            if (character direction if left)
                character move left
            end if
        end if

        if (character position > reference location)
            for (amount of objects)
                move all objects depending on character direction
            end for
        end if

        if (character is jumping)
            character jump
        end if


        if (character collides with floor)
            player falling ⟵ false
        end if

        if (character collides with obstacle)
            display "You died"
            pause for 1 second
            reset player to initial position
            reset all objects to initial position
        end if

        if (character collides with chest)
            chest visible ⟵ false
            addCurrency(50)
            display new currency
            chest collided ⟵ true
        end if




        if (character collides with finish)
```

```
                display "Level Finished"
                addCurrency(100)
                game closed ⟵ true
                close screen
            end if

            for (loop ⟵ 0 to amount of obstacles, inc by 1)
                animate obstacles by changing to next picture in array
            end for

            if (6th game tick)
                animate player by changing to next picture in the array
            end if

            public addCurrency(int i)
                currency ⟵ currency + i
                display new currency
            end method


        end while
        time ⟵ get current time in milliseconds - time
        set user best time

    stop thread run
```

# 2.5 Level Select Screen




**Level 1:**

```
if btnLevel1 is pressed
        Game1Screen set Visible
end if
```

**Level 2:**

```
if btnLevel2 is pressed
        Game2Screen set Visible
end if
```

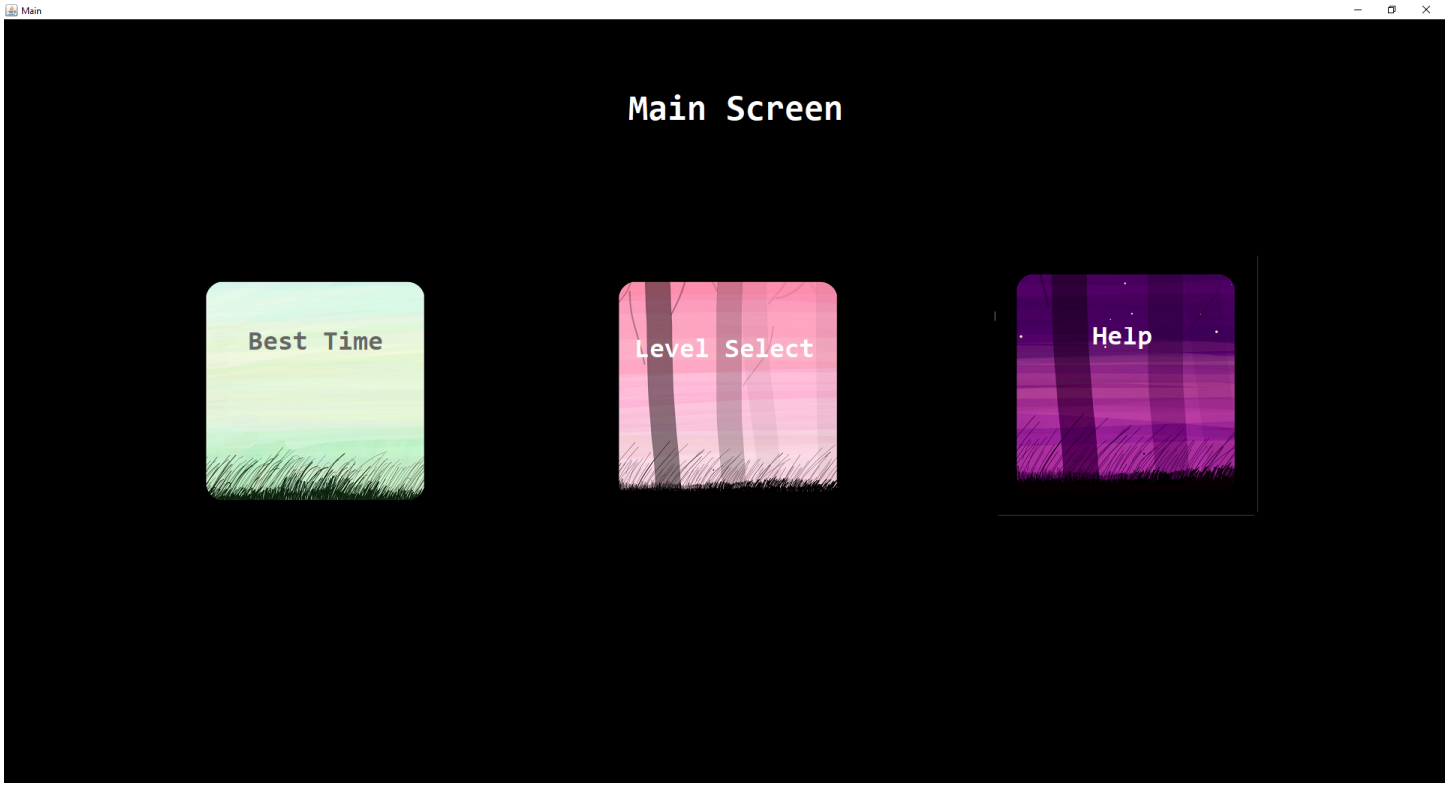
**Level 3:**

```
if btnLevel3 is pressed
        Game3Screen set Visible
end if
```

**Close:**

```
if btnClose is pressed
        close Level Select Screen
        show Main Screen
end if
```

# 2.6 Main Screen




**Level Select:**

```
if btnLevelSelect is pressed
        LevelSelectScreen set Visible
end if
```

**Help:**

```
if btnHelp is pressed
        HelpGUI set Visible
end if
```

# 3. Class Design

## 3.1 Character Control Class

| CharacterControl | |
|---|---|
| -dx: int | The current x coordinate of the character on the screen. |
| -x: int | The current direction of the character on the x-axis must be moving, either 0, 1 or -1. |
| -y: int | The current direction of the character on the y-axis must be moving, either 0, 1 or -1. |
| +Constructor(x : integer, y : integer, dx : integer) | Constructor. |
| +getX(): integer | Gets the current x-position of the character and returns it. |
| +getY(): integer | Gets the current y-position of the character and returns it. |
| +getDX(): integer | Gets the current direction of the character and returns it. |
| +setX(integer i) | Sets the current x-position of the character. |
| +setY(integer i) | Sets the current y-position of the character. |
| +setDX(integer i) | Sets the current direction of the character. |
| +characterMove(integer i) | Moves the character depending on the direction of the direction of the character. |
| +characterJump() | Makes the character jump. |
| +toString(): string | Returns a string of the data in the class. |

## 3.2 User Class

| User | |
|---|---|
| -userName: string | Stores the Username of the User. |
| -bestTime: string | Stores the Best Times of the User for each Level. |
| +Constructor(userName : string, bestTime : string) | Constructor. |
| +setUserName(string i) | Sets the userName that is read from the text file. |
| +setBestTime(integer i) | Sets the Best Time that is read from the text file. |
| +getUserName(): string | Gets the Username of the User. |
| +getBestTime(integer i): int | Gets the Best Time of the User for a Specific Level depending on the input, eg. 1 for Level 1. |
| +getLevel(integer i): int | Gets the level the user is currently on. |
| +toString(): string | Returns a string of the data in the class. |

## 3.3 Collision Detection Class

| CollisionDetection | |
|---|---|
| -collide: boolean | Stores the Username of the User. |
| -collideX: integer | Stores the X coordinate of the collision between the two objects. |
| -collideY: integer | Stores the Y coordinate of the collision between the two objects. |
| +Constructor(collideX : integer, collideY : integer) | Constructor. |
| +setcollideX(integer i) | Sets the collideX variable. |
| +setcollideY(integer i) | Sets the collideY variable. |
| +getcollideX(): integer | Gets the collideX variable. |
| +getcollideY(): integer | Gets the collideY variable. |
| +detectCollision(): boolean | Detects if there is a collision and returns true if there is, the user will then be able to get the coordinates of the collision. |
| +toString(): string | Returns a string of the data in the class. |

## 3.4 Animations Class

| Animations | |
|---|---|
| | |
| +sawImageArr(integer i): ImageIcon | Stores an array of saw images to create animations. |
| +playerRightImageArr(integer i): ImageIcon | Stores an array of right player images to create animations. |
| +playerLeftImageArr(integer i): ImageIcon | Stores an array of left player images to create animations. |
| +toString(): string | Returns a string of the data in the class. |

# 4. Persistent Storage Design

## 4.1 User.txt

| Field | Type | Description | Example |
|---|---|---|---|
| userName | String | Stores users name, updated when user is created. | Kian |
| userPassword | String | Stores users surname, updated when user is created. | Anderson |
| bestTimes | String | Stores users best times in seconds, updated when a new best time is achieved on a level. | 53#86#76 |
| userLevel | int | Stores the level the user is currectly on. | 1 |

**Text File Structure:**
userName#userPassword#bestTimes#1

**Sample:**
Kian#Anderson#312#1
Luke#Payne#512#2
Josh#Langley#343#3

## 4.2 Help.txt

| Field | Type | Description | Example |
|---|---|---|---|
| helpTopic | String | Stores the help topic, is never updated. | How to check Best Times |
| helpInfo | String | Stores the information on the help topic, is never updated. | Go to the main screen and press the Best Time Button. |

**Text File Structure:**
helpTopic#helpInfo

**Sample:**
How to check High Scores#The High Score Screen can be accessed on the Main Screen.
How to login as a new user#When in the login screen, select the sign up button.
How to Sign In#Enter your Name and Password in the Login Screen.

# 4.3 Levels.txt

| Field | Type | Description | Example |
|---|---|---|---|
| Obstacles | Int | Stores the X and Y coordinates of the obstacles. | 400,500 |
| Platforms | Int | Stores the X and Y coordinates of the platforms. | 400,500 |
| Collidables | Int | Stores the X and Y coordinates of the collidables | 400,500 |

**Text File Structure:**
object#X-coordinate,Y-coordinate

**Sample:**
obstacles#400,500#123,542#534,123
platforms#234,234#543,123#873,982
collidables#123,543#987,345#132,623

# 5. Explanation of Storage Design

## 5.1 Primary Memory (Class Design)

### 5.1.1 Character Control Class

- The Character Control Class is needed to move the character and to make the character jump, it is also needed to get the X and Y coordinates of the character. It controls all the movement of the character.
- The Character Control Class stores the current X and Y coordinates of the character, it also stores the direction of the character on the x and y axis.
- It is instantiated when a the user selects a level from the Level Select Screen, and is instantiated in the GameScreen.

### 5.1.2 User Class

- The User class is needed to store and access User information for later use in the game, such as for the Best Time Screen to get the user Best Times and the Username of the user.
- The User Class stores the Username of the User and the Best Times of the User.
- It is instantiated after the user successfully is logged in the Best Time Screen when it is needed.
- The User Class gets its data from User.txt, it extracts the User's name, the User's password, the User's best time and the level the user is currently on.

### 5.1.3 Collision Detection Class

- The Collision Detection Class is needed to detect if two objects have collided and return if whether they have or not, and to provide the X and Y coordinates of the collision.
- The Collision Detection Class stores a Boolean whether the two objects have collided and the point of collision.
- It is instantiated after the User has selected a level in the Character Control Class.

### 5.1.4 Animations Class

- The Animations Class is needed to make an array of image icons that are used to animate the character and other objects in the level such as the obstacles.
- The Animations Class stores image icons in an array.
- It is instantiated when the user selects a level in the Level Select Screen.

# 5.2 Secondary Storage

## 5.2.1 User.txt

- Stores username, userPassword and userHighScore
- The game needs he username to identify the user, it needs the password to verify the user and the high score to show the user his previous high scores. It also stores the level the user is currently on and which levels the user has completed.
- It is accessed when the user either Signs in, it then identifies and verifies the user, it is accessed when the user Signs Up to create a new user in the database and it is also accessed when the user has completed a level to store the best time and to access the best time in the best time screen.

## 5.2.2 Help.txt

- Stores helpTopic and help information.
- The game needs the helpInfomation to display in the HelpScreen and the helpTopic to identify the helpInfomation.
- It is accessed when the user selects a help topic to be displayed in the Help Screen and when the user opens the Best Time Screen as it loads the user Best Times and the Username of the user.

## 5.2.2 Levels.txt

- Stores the position of obstacles, platforms and collidables.
- The game screen needs the positions of all the objects to position them in the correct place depending on the level the user selects.
- It is accessed in the game screen to correctly position the objects in the Game Screen, it is accessed when the user selects a level.