Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Course No: CSE 1200

Course Name: Software Development-I

Group: B1

Course Instructors:

Nibir Chandra Mandal
Mohammad Marufur Rahman

Project Name:

Try Not to Fall

Make A Run for It

Group Members:

Animesh Das Chowdhury (190204058)

Tonmoy Banik (190204059)

Mohammad Tanveer Shams (190204060)

Sanjida Akter (190204076)

Table of Contents

1. Introduction

We were given a task which was to make a project (2D Game) in Visual Studio using iGraphics. In this project we used C++ language including concepts of File, Structure and Pointers. Inside the game our main target was to help the main character (Akash) pass all the **thirty pillars** in order to get rid of an abandoned inland.

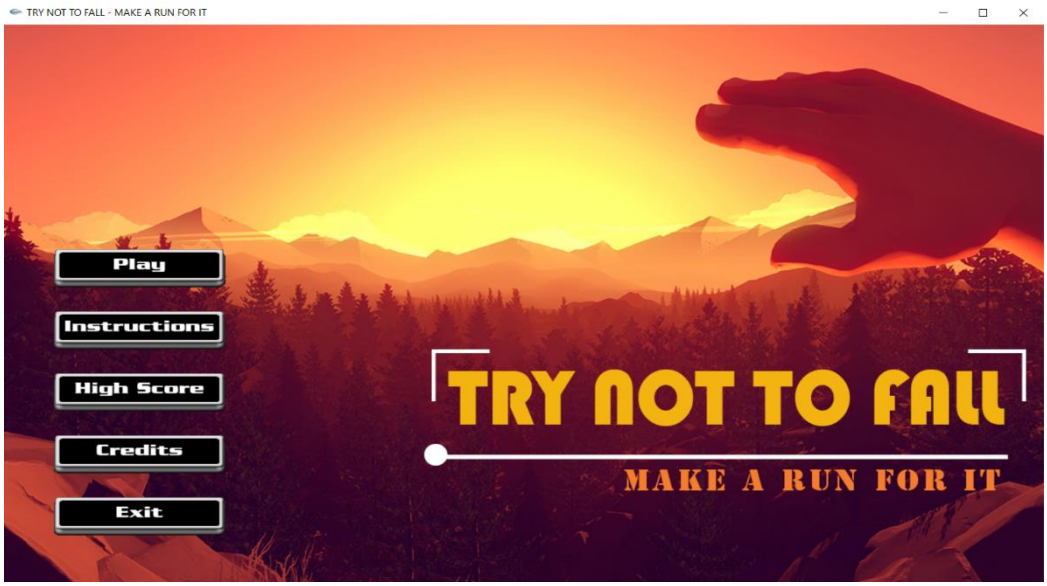
1.1. Project Overview

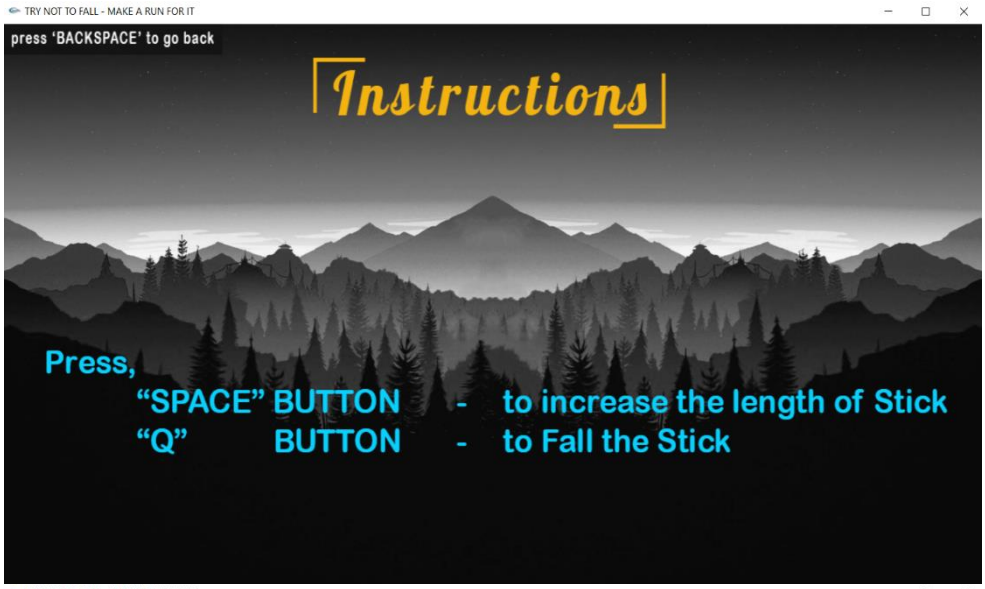
Our game is a simple storyline-based game. The player has a stick which has to be fallen from one pillar to next one and if player fails to fall the stick correctly, health decreases. If player runs out of health, the game overs. There are 3 different stages in our game based on difficulties (**Easy** = 5 health, **Medium** = 3 health & **Hard** = 1 health). As it is a story-based game, the game finishes after finishing **3rd stage**. A player can only input his/her name in high score board after passing all of the three stages in as short time as possible.

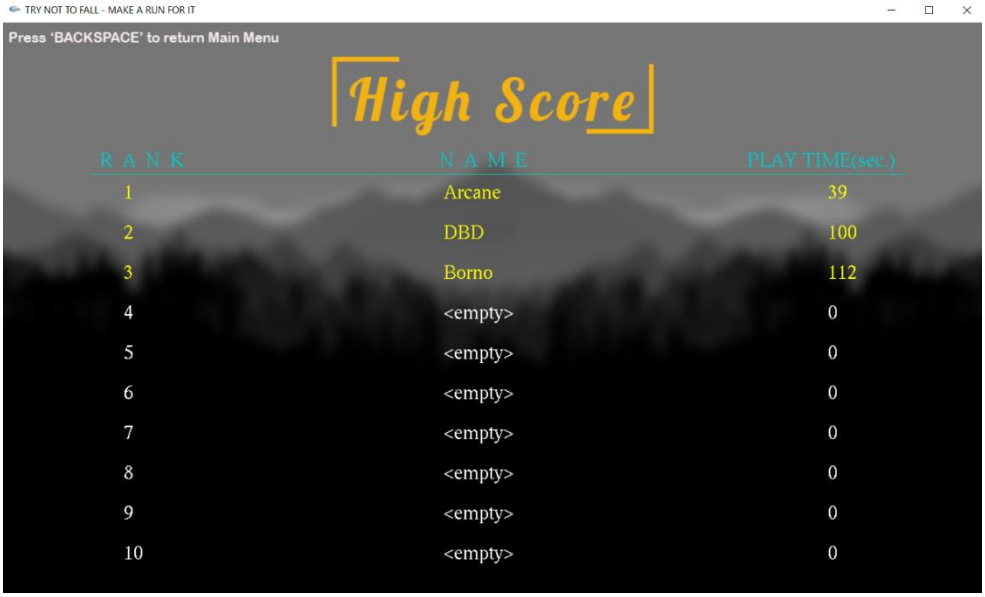
1.2. Game Idea & Storyline

We got the idea of our project from a famous android game named **Stick Hero.** We just tried to rebuild the game by mixing up it with a short story. We harvested our storyline idea from the song **Bibiya** which is one of the famous songs of Bangladeshi rock band **Shunno.** Our storyline is basically like our main character (**Akash**, an orphan boy) and his sister (**Bibiya**) lives together. While fishing, one day Akash got washed away by a storm and after sometime he found himself in an island. Now the game starts here where the player is going to help Akash in order to get rid off this island to meet her sister back. We calculated the time while playing the game and saved it as high score after player finishes the whole game.

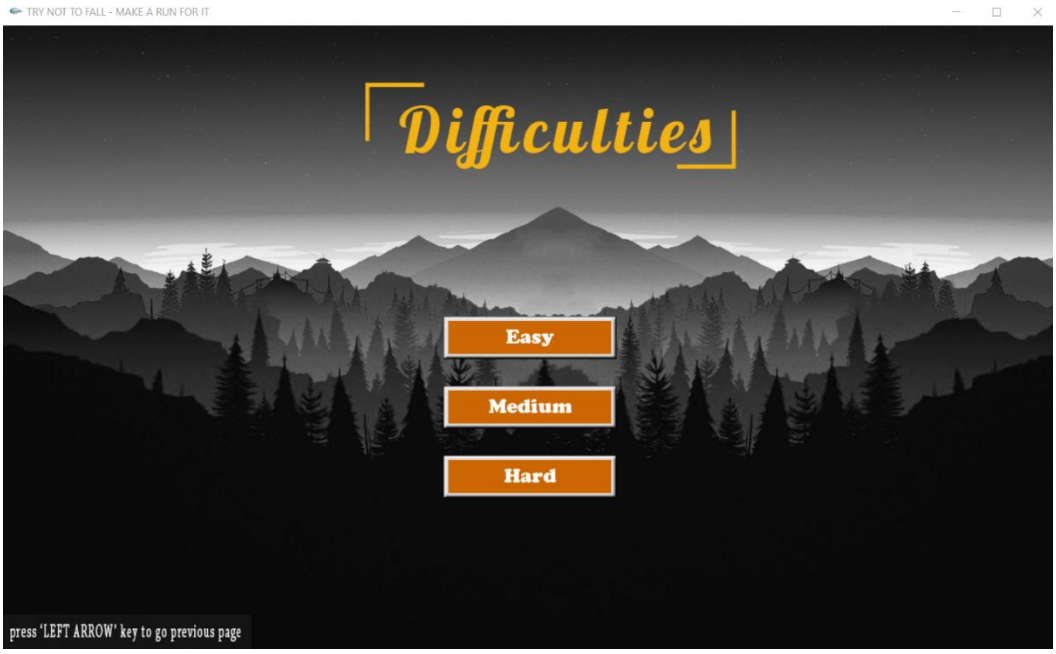
1.3. Cut-outs of Project



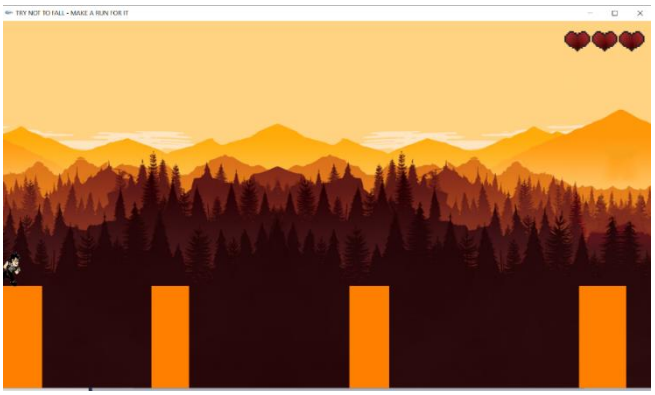

TRY NOT TO FALL - MAKE A RUN FOR IT
press 'BACKSPACE' to go back
Instructions
Press,
"SPACE" BUTTON - to increase the length of Stick
"Q" BUTTON - to Fall the Stick


TRY NOT TO FALL - MAKE A RUN FOR IT
Press 'BACKSPACE' to return Main Menu
High Score
RANK NAME PLAY TIME(sec.)
1 Arcane 39
2 DBD 100
3 Borno 112
4 <empty> 0
5 <empty> 0
6 <empty> 0
7 <empty> 0
8 <empty> 0
9 <empty> 0
10 <empty> 0

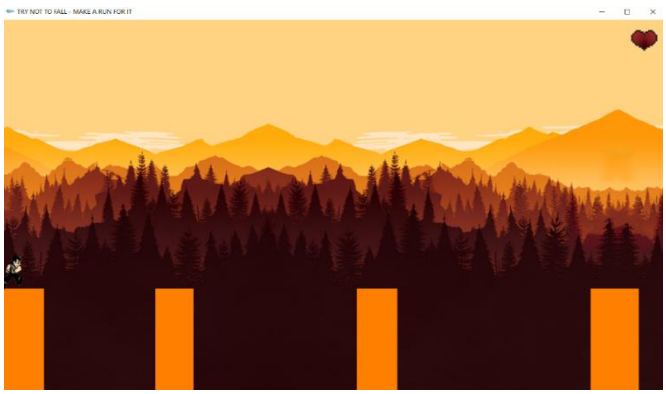

TRY NOT TO FALL - MAKE A RUN FOR IT
press 'BACKSPACE' to go back
Credits
Animesh Das Chowdhury
(190204058)
Tonmoy Banik
(190204059)
Mohammad Tanveer Shams
(190204060)
Sanjida Akter
(190204076)
Storyline & Sound
Inspired From
Game Concept
Inspired From



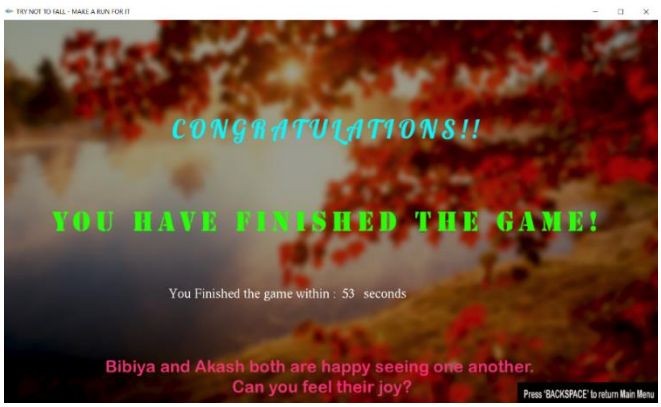
Easy



Medium



Hard



Game Finish Screen

2. Game Mechanism and Functions

Our game mechanism is very simple and user friendly. Player has to use just two buttons which are **Space** and **Q.** Pressing **Space** button generates the stick. When player presses **Q** it makes the stick to fall and moves the character from one pillar to next one (if the length is sufficient). Otherwise, it will reduce player's health.

2.1. Structures and Arrays

While making this project we used in total four structures (pillar, stick, background & menu button). We also saved all the values of stick and pillars (axis, height, width) using arrays. We also used arrays to store all the cut-outs of our backgrounds and character. In high score part, we used array to store and sort the high scores.

2.2. Built-in Functions

In order to work perfectly and simply we used some of Visual Studio built-in functions like-

2.2.1. iMouse () & iPassiveMouseMove ()

We used these two functions to do the work of hover and click in menu page. We also used these to play and pause the sound of click;

2.2.2. iKeyboard () & iSpecialKeyboard ()

These two functions were used in transitions between two game phases, player name input, stick rotation and stick generate.

2.2.3. iInitialize ()

Initialized the game by using this.

2.2.4. iSetTimer () & iPauseTimer ()

Each and every animation seen in this game (pillar movement, character movement, stick rotation, background movement etc.) were made by using these two functions.

2.2.5. iSetColor (), iLoadImage () & iShowImage ()

We used **iSetColor ()** in order to colour the pillars and sticks. **iLoadImage ()** and **iShowImage ()** were used to load images from project folder and show them in the project while playing the game.

2.3. User Defined Functions

We used some own created functions to sort and use the code segments according their works.

2.3.1. Generate, Selection and Movement

setBackground () & **pillGen ()** functions were used to generate background, pillars and sticks. **PillMov ()**, **BackgroundMotion (), CharacterMotion ()** and **angle_change ()** to show animations through game.

2.3.2. GameState Functions

In this game we used a lot of game state functions (i. e. **GameState00()**, **GameState01()** etc.) to sort the codes according to game phases so that whenever we can debug codes easily.

2.3.3. High Score Functions

sortleaderboard (), **updateleaderboard ()**, **leaderboard ()** & **clockcapture ()** were used to sort, store and update the whole score board. Here we also used two Files (**Name** and **Score**) to store them as a text document.

3. Problems Faced & Solutions

While implementing our concept of the game into codes we faced a couple of problems both conceptual and logical. However, we were able to solve them together by acquiring a better understanding of the concept of our game's mechanism and implementing modified logics for its code.

3.1. Pillar Generating Problem & Solution

At first, to generate pillars for the game we used three **pillGen ()** functions and initialized all three of them at the very beginning which resulted in always showing the pillars generated only by the third **pillGen ()** function as it overrode the other two functions for pillar generation.

We solved this problem by generating the pillars of each level in their respective levels as the game progressed by calling the **pillGen ()** functions only when the player progressed to the next levels.

3.2. Stick Generating Problem & Solution

Our first concept was to generate a stick and keep using that for the whole game which while implementing turned out to be much difficult and the stick was only working for the first pillar. Then we generated individual sticks for each pillar maintaining the same co-ordinates for both the pillar and the stick. However, doing this solved the problem for the first level only, as the level passed the sticks were still in the state it was in the previous level and couldn't be modified.

We solved this problem by using a boolean variable in the stick's structure and resetting it with each level being completed.

3.3. Pillar Movement Problem & Solution

We defined a **PillMov ()** function and used **iSetTimer ()** for the transition of the pillar and stick from one to another and used **iPauseTimer ()** to stop the transition after shifting to the crossed pillar. To do this we at first used the logic that the timer will pause when the next pillar's x co-ordinate will become less or equal to zero. However, this logic made the timer stop permanently as the 2nd pillar's co-ordinate reached zero.

We maintained a boolean variable in the pillar structure and set the timer to pause or resume based on that variable which solved our problem.

3.4. Scoreboard Sorting Problem & Solution

While sorting the scoreboard, we at first, just sorted the scores in ascending way which resulted in the score to be always zero, as the score was always greater than **zero** and we pre-initialized the scoreboard with all zeros.

We solved the problem by swapping the last score in the scoreboard with the updated score and re-sorting the scoreboard handling the problem with zero.

3.5. Sound Problem & Solution

We faced a number of problems during adding sound to the game as the implementations of sounds and their transitions were not the same as we wanted at first. We had to get a better grip on the bass.h library and its functions and implemented them carefully in order to solve the problems with sound.

4. Ways to Improve

Several things could have been improved with more time and a better understanding of all the built-in functions and their implementations. We could've displayed and updated the score while the player was playing the game. Some more sound effects could've been added and the animation could've been smoother with more slices and more optimized implementation of the **iSetTimer ()** function. The code could've been more optimized and easier to modify with a more practical hold of the functions and their implementations creating the possibility of adding more effects like the fall effect of the character.

5. Conclusion

As beginner level students we tried our best to create our project as flawless as possible. During this work we gathered a lot of ideas and experiences about iGraphics and Visual Studio. Our instructors and other members also helped a lot to make this project fruitful.