Course Project Documentation

BRAVE BRICK

Group number: 288

CS101 Software Project

SUSHMENDRA KUMAR SINGH 140040080
(Team leader)

ARUN KUMAR KADAGANCHI 140040101

BALU RAJENDRA PRASAD 140040091

DHARAN KONDAVEETI 140040110

Table of Contents

1. Introduction.....2

2. Problem Statement.....2

3. Requirements.....3

4. Screenshots.....4

5. Implementation ……..11

6. Future Work.....13

7. Conclusion.....13

8. References.....14

Introduction:

Our project, “Brave Brick”, is an arcade game in which the player controls a brick and attempts to dodge various obstacles which appear from the top of the playing screen.
The player uses the left and right arrow keys to move the brick.

The obstacles are entirely random and therefore you will face a new set every time you play the game.

Touching the obstacles causes the game to the end, displaying your score. Your score is the number of levels you successfully manoeuvre through multiplied 5.

A level consists of 5 platforms and 8 small square shaped obstacles.

Problem Statement:

The aim of our project was to make a fun arcade game which anyone can play, using simplecpp and things we learnt from the CS101 course.

Requirements:

A) Hardware Requirements:

No external hardware is required for a working computer, i.e. a computer which has a working keyboard and monitor and is able to run Simple Codeblocks.

B) Software Requirements:

Simple Codeblocks or

Codeblocks IDE with

gcc compiler

standard library

simplecpp library

graphics.h library

math.h library

Some screen shots are attached below and some text explaining their significance.

Screenshots:

The first screenshot shows the interface when you run the game.

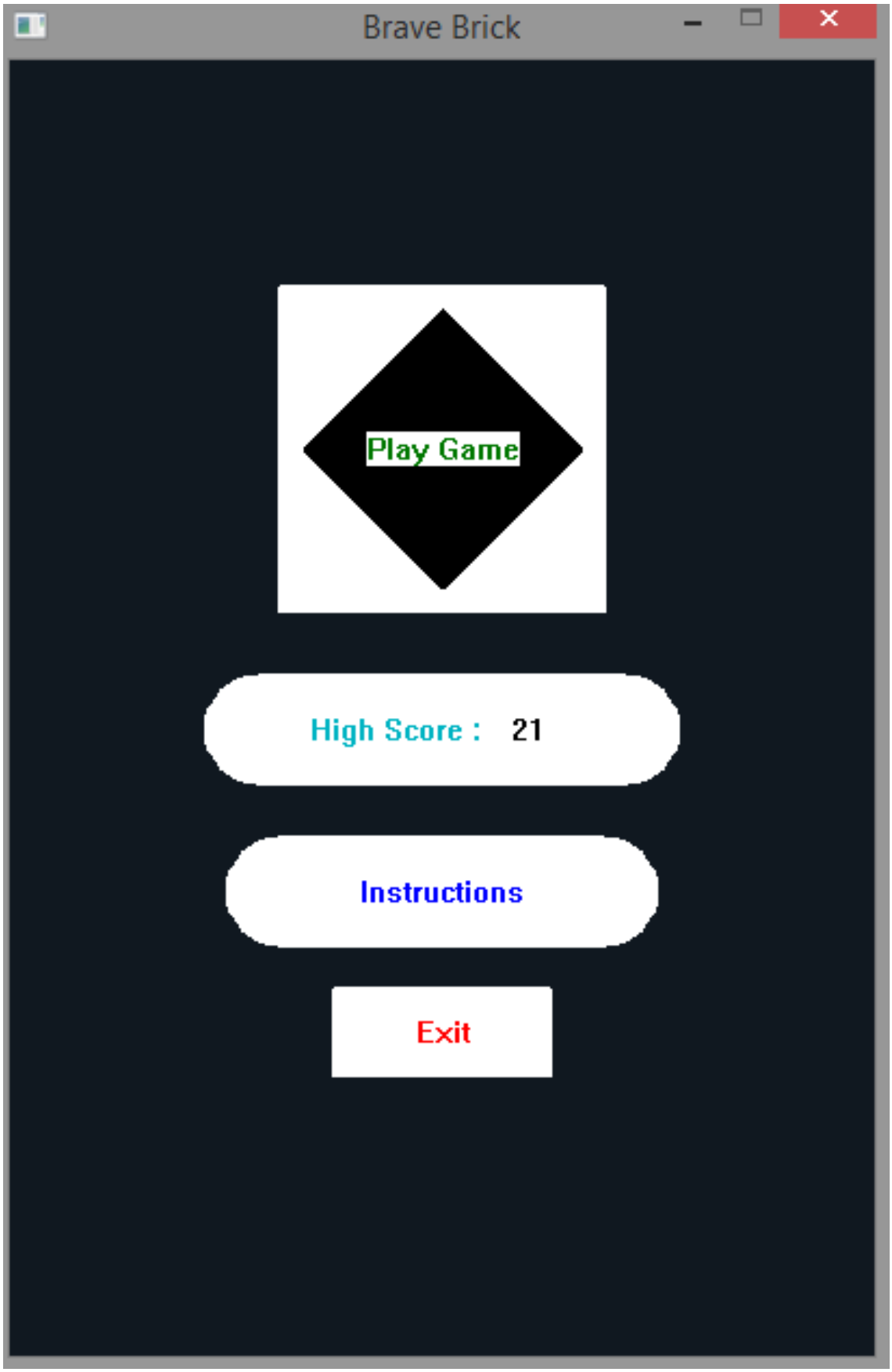
Clicking play game starts a new game.
Your highest ever score is displayed below the “Play Game” button.
Clicking on the instructions displays the instructions of the game, which tells first time players how the game works and how it is supposed to be played.
Clicking “Exit Game” exits the game.

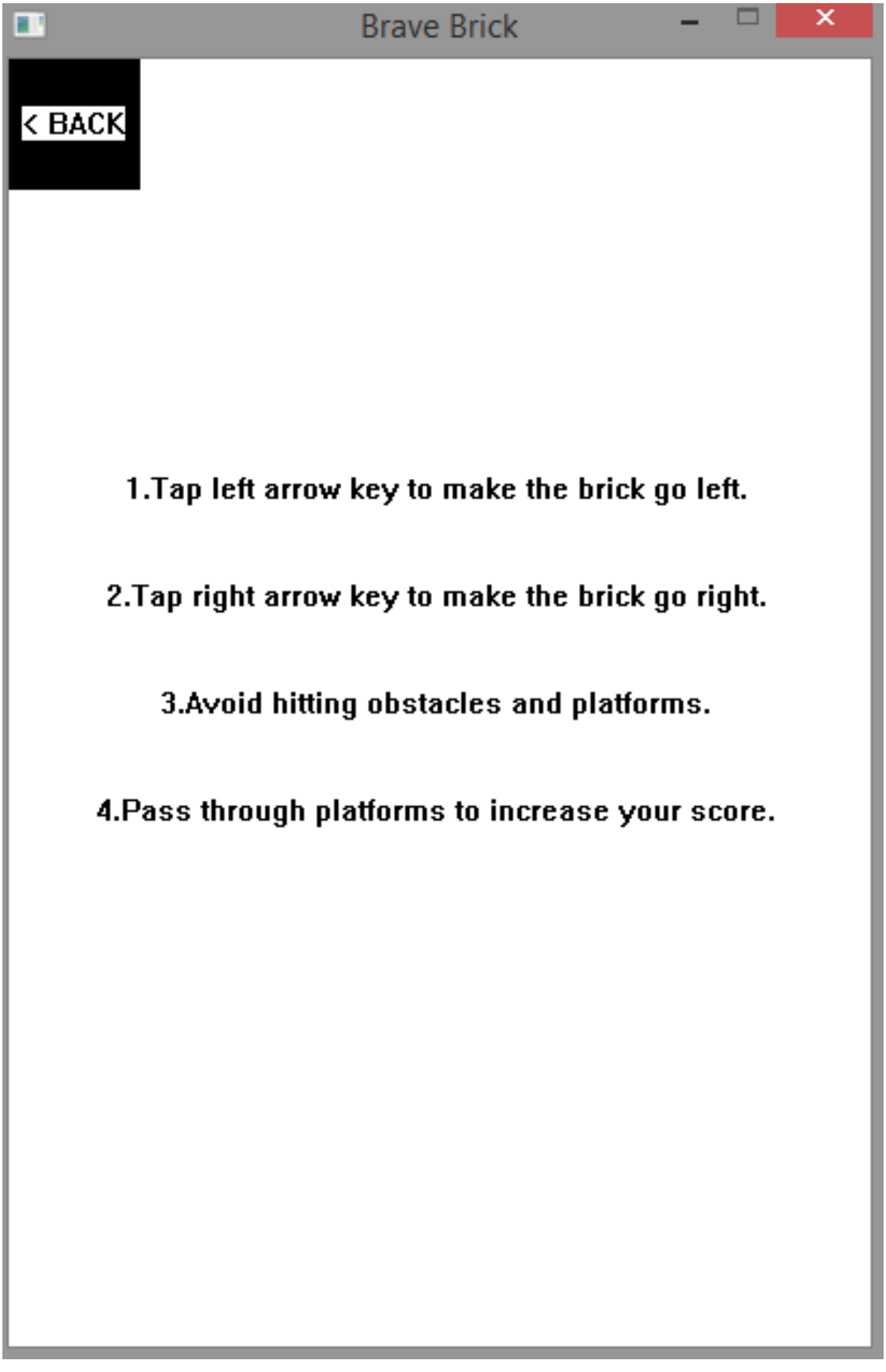
The second screenshot displays the instructions.

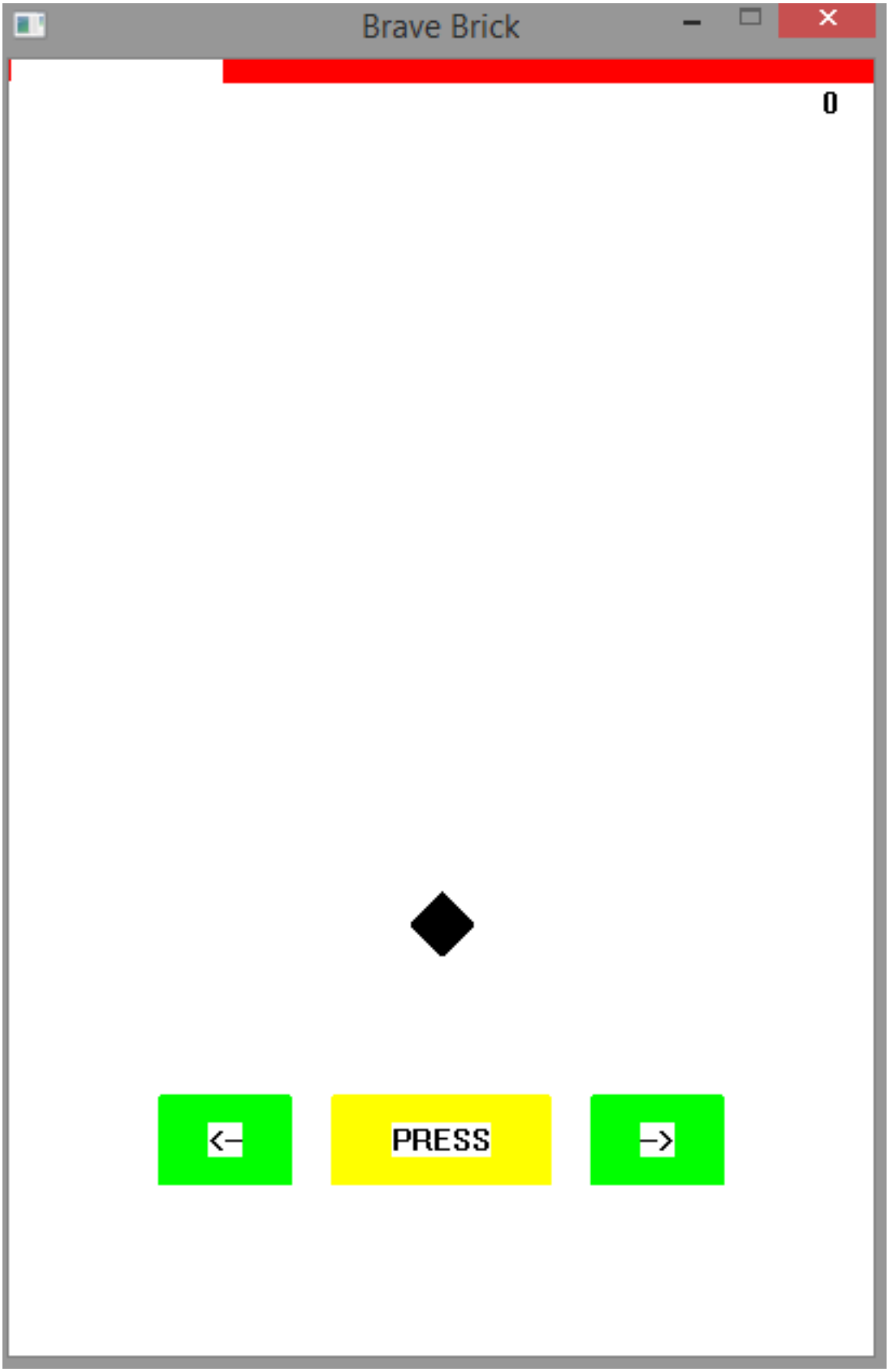
The third screenshot shows the playing screen at the beginning of the game.

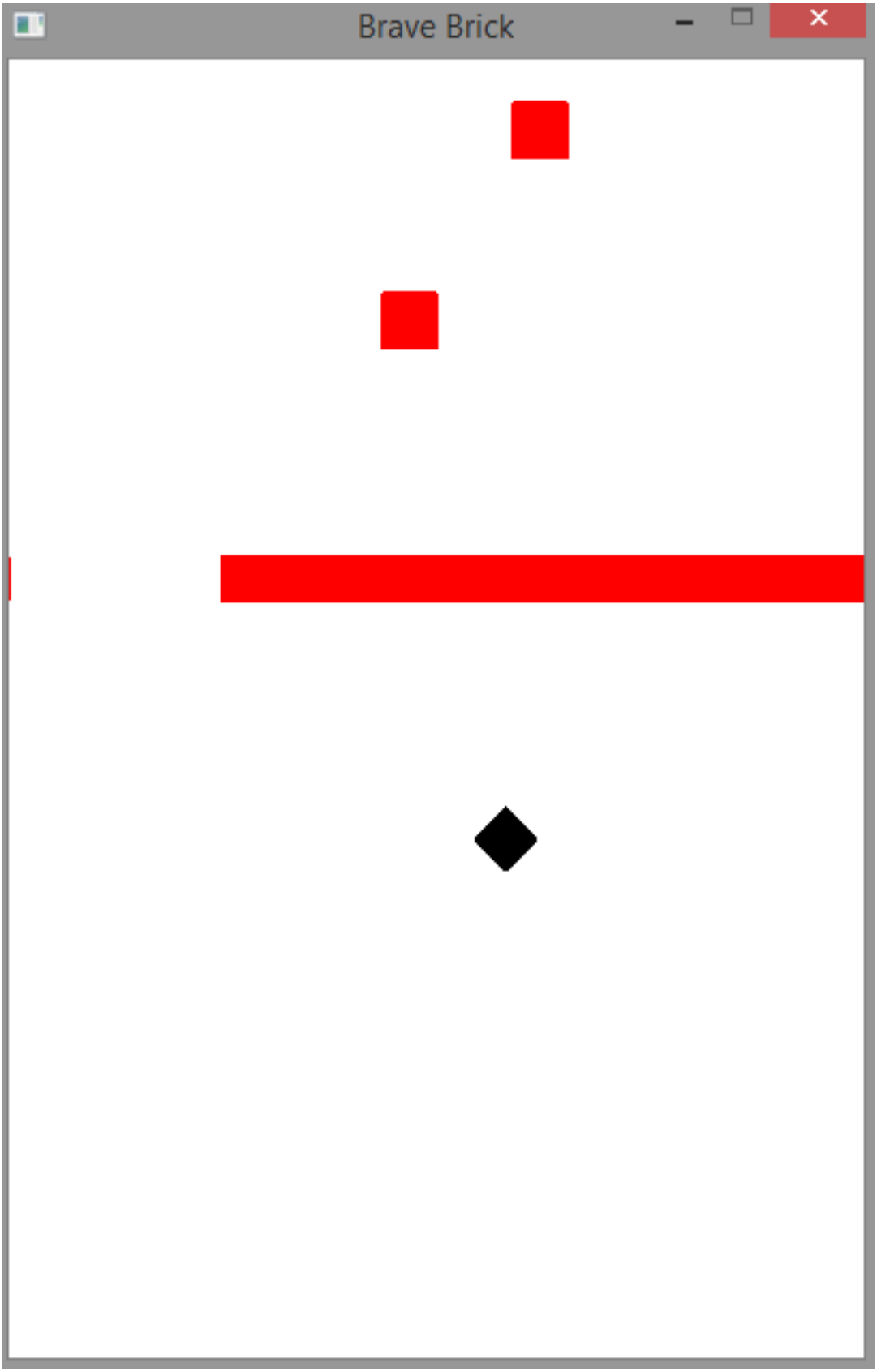
The fourth screenshot shows the playing screen at a random point in the game.

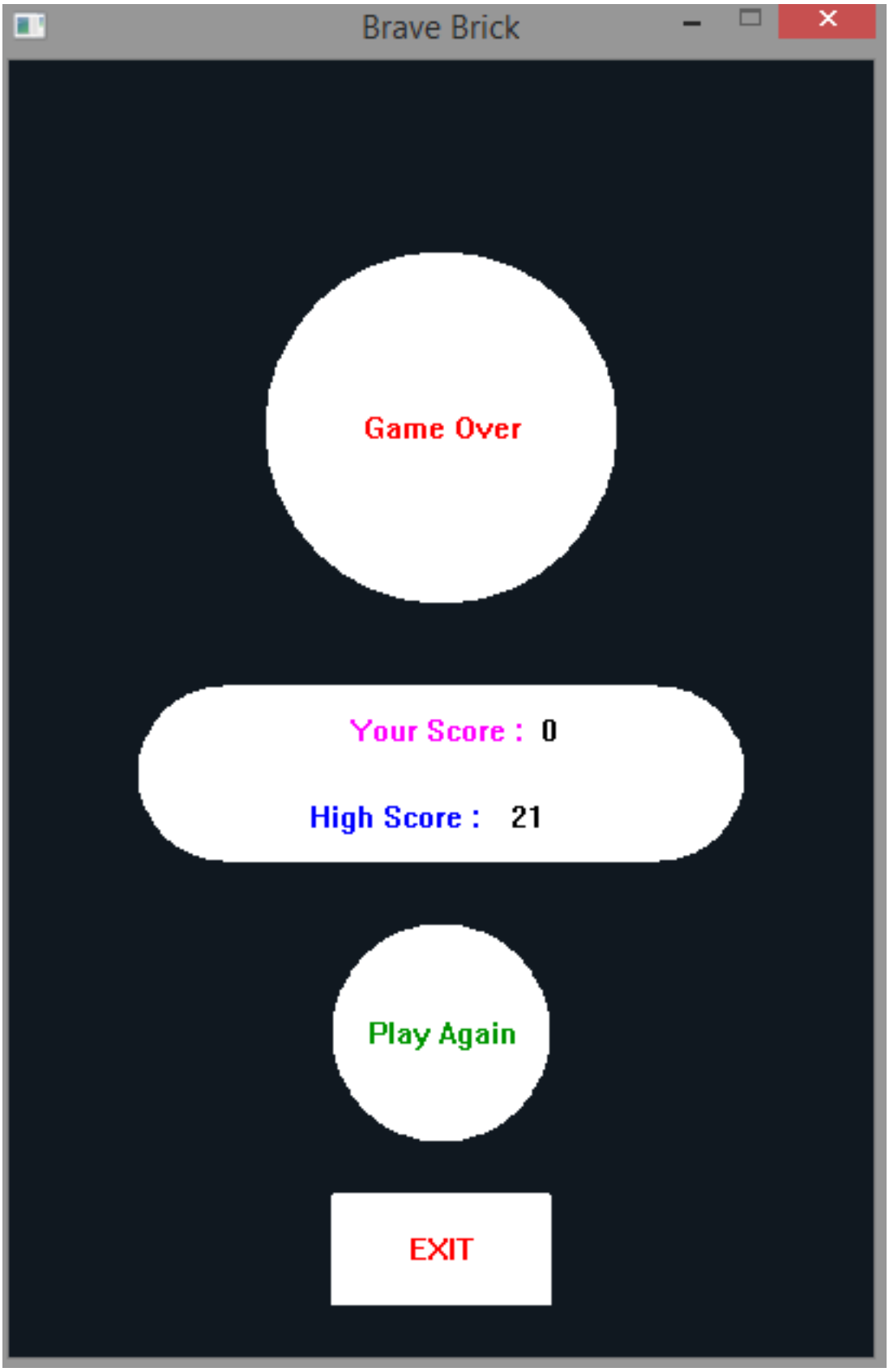
The fifth screenshot shows the game over screen, which shows up when the brick collides with one of the obstacles or hits the boundaries of the playing screen.


Brave Brick
Play Game
High Score : 21
Instructions
Exit


Brave Brick
< BACK
1.Tap left arrow key to make the brick go left.
2.Tap right arrow key to make the brick go right.
3.Avoid hitting obstacles and platforms.
4.Pass through platforms to increase your score.


Brave Brick
0
<-
PRESS
->


Brave Brick


Brave Brick
Game Over
Your Score : 0
High Score : 21
Play Again
EXIT

Implementation of our project:

Instructions to run our program:

Download simple codeblocks from http://www.cse.iitb.ac.in/~ranade/simplecpp/ Follow the instructions and install the program.

Download the files BraveBrick.cpp, BraveBrick.exe and high_score.txt.

Run the BraveBrick.exe file to play the game.

YouTube URL for our game's video tutorial:
https://www.youtube.com/watch?v=GYsXU5Z0MYw&feature=youtu.be

Key Problems Faced In the Project and How They Were Solved:

The first problem we faced was checking, while the brick was moving, if the user pressed left or right control key and if he did which one he pressed. This was solved using the GetAsyncKeyState() function.

Coordinating the movement of the brick with the obstacles was a problem we faced early on and we had to change our entire approach towards the code.

We had to make 6 functions, 4 for the movement of the brick
1. left_when_pressed
2. right_when_pressed
3. left

4. right

These functions were created to enable the user to change the direction of the brick at will.

And two functions to check whether the brick hit the platforms or obstacles.

1. check
2. check_last,

These two functions were used to check, at every step of the movement of our brick whether the brick had hit an obstacle or not.

Once these two problems were solved, the remaining part, i.e. creating the moving obstacle structure did not pose too much of a problem.

Scope for improvements and additions in the future (Future Work):

Power-ups can be added to enhance the game playing experience.

The code could be stream-lined to reduce the amount of memory it takes up and make the program faster.

The movement of the brick could be changed to make it seem more like an actual projectile.

Conclusion

All the elements we said we would add, in the SRS, have been added to our game, except for the fact that our brick doesn’t move in a parabolic path, but in a straight line.

References:

An Introduction To Programming Through C++ by Ranade;

www.dreamincode.net

www.codeacademy.org

www.higherorderfun.com