Technical Specification

Project name:	BlockBreaker
Students:	Stephen Mcdonagh 14518233
	Anas Alagtal 14511433
Supervisor:	Mark Roantree

Contents

1 Introduction

1.1 Overview

For our third-year project we decided to make our own version of the classic brickbreaker game and called it ‘Blockbreaker’. I have described below the different features we implemented and how we did so. To summarise, the goal of the game is to destroy all the blocks and get the highest score possible to do so. By choosing a more challenging difficulty, the player will be rewarded with more points

By doing large amounts of testing we were able to find many different bugs within our game and spend the time necessary to fix these problems. I believe the game we created a challenging game which the user will enjoy playing as well as good user interface.

Description

- The goal of the game is to complete all levels within the game and obtain the highest score possible.
- This score will be added to the database we created, and the user will try to beat other players score on the database.
- When a block is destroyed the user will earn points
- The amount of points will depend on what difficulty the user chooses.
- Difficulty within the game is determined by the speed of the ball. So if the user chooses a hard speed, the ball will be much faster than the easy speed
- A level is completed when there are no more blocks left to destroy
- The user has 3 lives to complete the game, if they run out of lives the game is over.

1.2 Motivation

The reason behind choosing this project was that we both had an interest in game development and thought it would be a good way to get experience. We felt there was many ways we could be creative by choosing this project and there were endless amounts of possible features we could add. As the game is old we wanted to make our own version of it different

We felt by doing a project around game development it would give us an insight into what it may be like to have a career in this field and if it would be something we would be interested in.

1.3 Research

Once we decided on the type of game we wanted to create, we then did some independent research and met back up a week later to discuss the different approaches we could take to developing out game. As game development is something that is new to us both we knew research was going to be very important.

We watched different videos on game development and learned so much from just these videos. We learned a lot about performance issues, reliability issues, testing and much more from videos. We found this very helpful and could apply what we saw in the videos to our own project.

2 Programs & Languages used

2.1 Languages used

Java was used for the implementation of our project

2.2 Programs used

Eclipse (IDE) was used in the development of our game.
Microsoft SQL server management studio – used for database

3 Problems & Resolutions

3.1 General Problems & resolutions

Problem: Neither group member has ever set up and connected an application to a database before. This was a feature we wanted to include from the beginning of our project.

Solution: A lot of research needed to be done to make sure we could get this working. We researched separately the best way to connect the game to the database and hold the users name and their high score.

Problem: Creating configuration file to hold properties such as ball speed, and points to be awarded to user.

Solution: This was something very new to both members. We stumbled across this when searching what is the best way to change the ball speed within the game. We originally planned to just use variables within our program but was told this is not the most efficient method. By using a configuration file, the application does not need to be re-complied after any changes have been made and this means saving time.

3.2 Game related problems & resolutions

Problem: Choosing how to move paddle

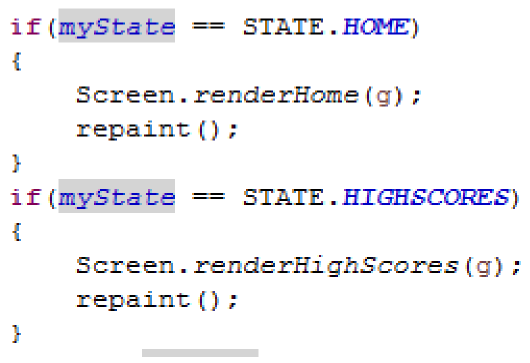
Solution: From the beginning we chose to use the ‘a’ and ‘d’ arrow keys to move the paddle, as this is common within PC games. After receiving some feedback from our user testing, we found that users would find it easier to understand by using the left and right arrow keys. This was an easy change to make and this meant the game was better for our users.

Problem: Ball bouncing in odd direction when hitting a block & ball rolling along paddle

Solution: This problem took us a long time to figure out. In total id say we spent nearly a week debugging our program. When we finally sorted this problem, we realised we just had to change one variable and the game worked perfectly. We had a similar solution to the ball rolling along the paddle which only happened when the ball hit the corner of the paddle.

Problem: Moving the game from home to difficulty to level 1

Solution: This was something that we were unsure how to do even though it seemed like a pretty simple problem. After much research we came across an ‘Enum’ which is a special variable type within java. This allowed us to change from different states within the game and made our problem a very easy one to solve



```
if(myState == STATE.HOME)
{
    Screen.renderHome(g);
    repaint();
}
if(myState == STATE.HIGHSCORES)
{
    Screen.renderHighScores(g);
    repaint();
}
```

(Rendering graphics depending on game state)

4 Design

4.1 Design goals

When designing our project, we set out some goals in which we wanted to achieve. We wanted the game to be easy to play, make the user want to come back and play again and again, and overall be an enjoyable experience for the user. Below is a list of the design goals.

Easy to learn:

From personal experience of bad User interfaces and from our HCI module we had learned of the importance of a good UI to the user. A badly designed UI can make the user frustrated and potentially give up on what they originally planned to do. We took this into consideration when designing our UI and believed that if we kept it simple users would enjoy the experience more. We didn't want the game to be overly complex as this would make unexperienced people less interested in the game too. The objective of the game should be clear to the user too. Important information should be clearly visible to the user at all times.

Reliability:

This is without a doubt the most important part of developing a game. You want the final version of the game to run smoothly with no errors. This then brings in the important of unit testing and user testing as this will help you fix any bugs present. A reliable game removes any possibility of chance and therefore it's all down to the user's skill. A reliable game requires a strong implementation process and enough time should be set aside to make this possible.

An example within our game was that the ball would bounce in an odd direction and this was not the users fault. We were able to fix this problem and therefore make the experience a more enjoyable one for the user.

Documentation

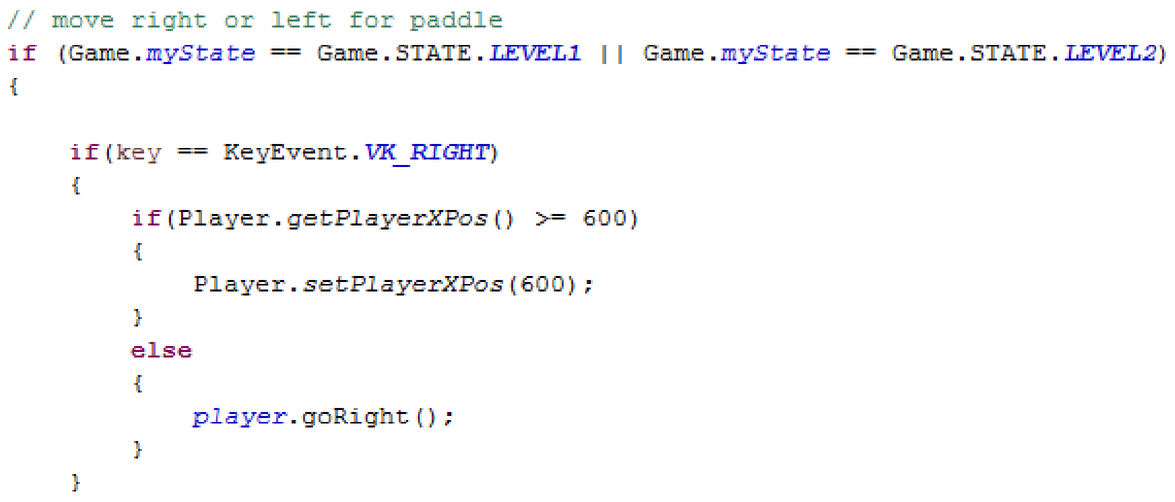
We were told about the important of documentation and throughout the process of this project we could see why. We had to create a functional specification which would outline how the game would be created, scenarios within the game and the business context. The technical specification reflects the initial and current design of the system. The user manual is a detailed step by step process of how the user will use your system. Finally, the blogs are used to keep track of your progress, problems occurred and solutions to these problems. Using all these different forms of documentation can really help when it comes to the implantation of your project.

4.2 Code examples

I have included some screenshots of the code used to perform different tasks within our game.

4.2.1 Key pressed (moving left or right)

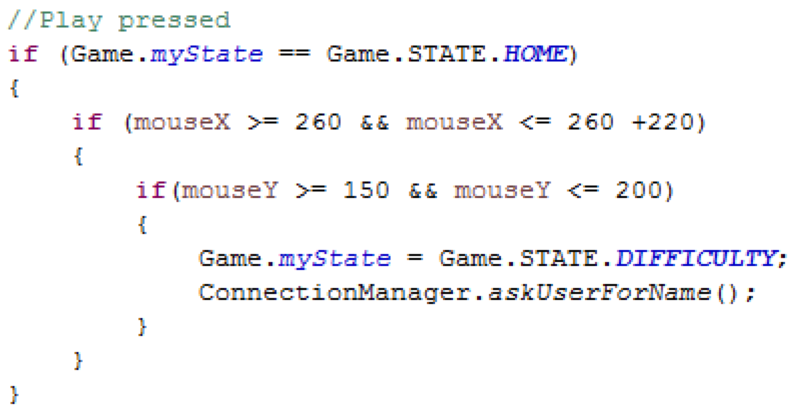
This was of importance as the user will need to move the paddle to hit the ball. The state must be either LEVEL1 or LEVEL2 as this is only when this method is of importance. We had to make sure the players position doesn't go out of the JFrame which is why we set it to 600, else it will move right.



```
// move right or left for paddle
if (Game.myState == Game.STATE.LEVEL1 || Game.myState == Game.STATE.LEVEL2)
{

    if(key == KeyEvent.VK_RIGHT)
    {
        if(Player.getPlayerXPos() >= 600)
        {
            Player.setPlayerXPos(600);
        }
        else
        {
            player.goRight();
        }
    }
```

4.2.2 Mouse pressed event

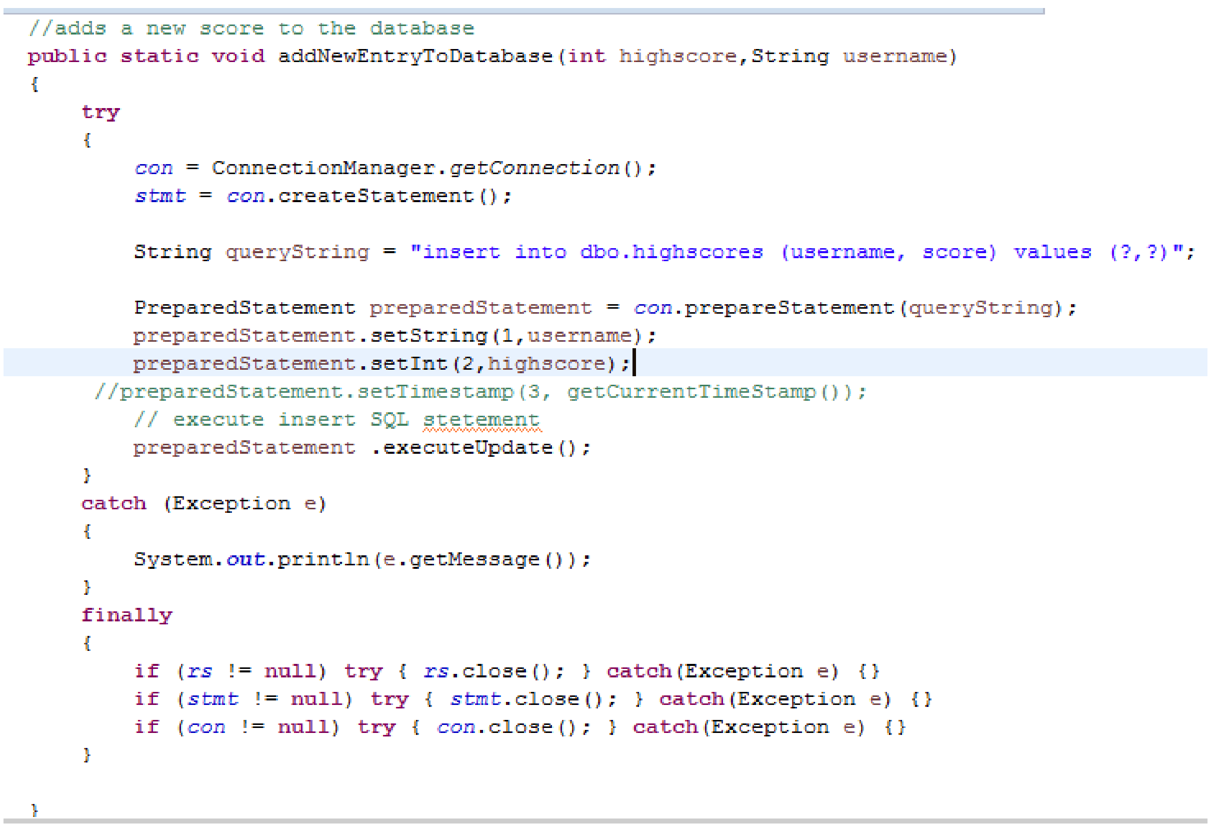
Within this if statement we are checking if the mouse has been pressed within some specific co-ordinates. Within these coordinates if the button 'PLAY'. This will bring the user to the next state which is difficulty choice and we also ask the user at this stage to input their name. We want them to input their name, so we can later on add it to the database with their score.



```
//Play pressed
if (Game.myState == Game.STATE.HOME)
{
    if (mouseX >= 260 && mouseX <= 260 +220)
    {
        if(mouseY >= 150 && mouseY <= 200)
        {
            Game.myState = Game.STATE.DIFFICULTY;
            ConnectionManager.askUserForName();
        }
    }
}
```

4.2.3 Add to database

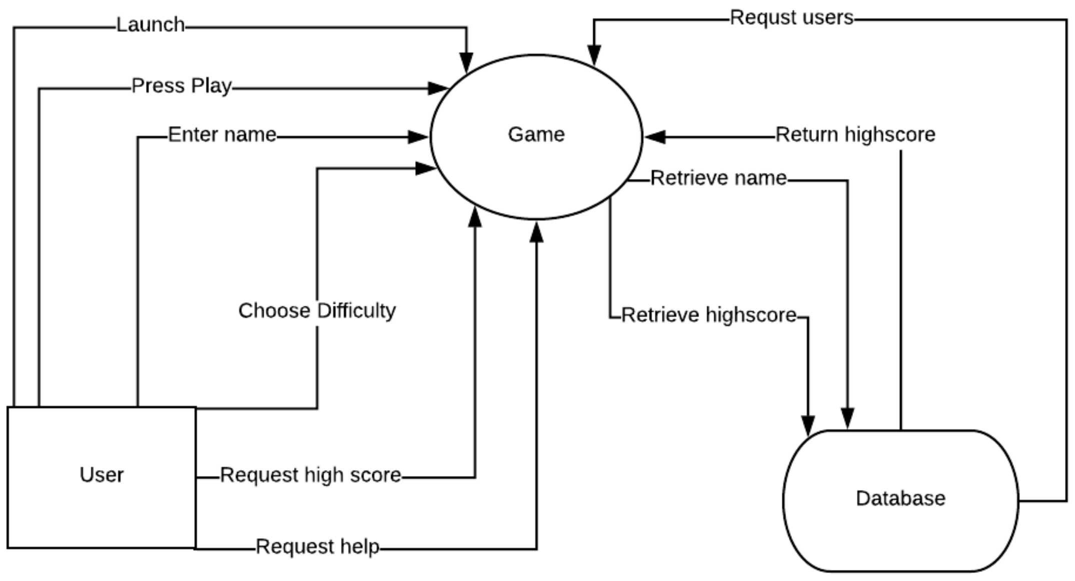
Within this method, we are getting a connection to our database, if it fails we will be told of this. We have our query which is insert into our database the name and score of the user. This will then update the database and if the user has got a new high score they will be able to view this in the high score section.



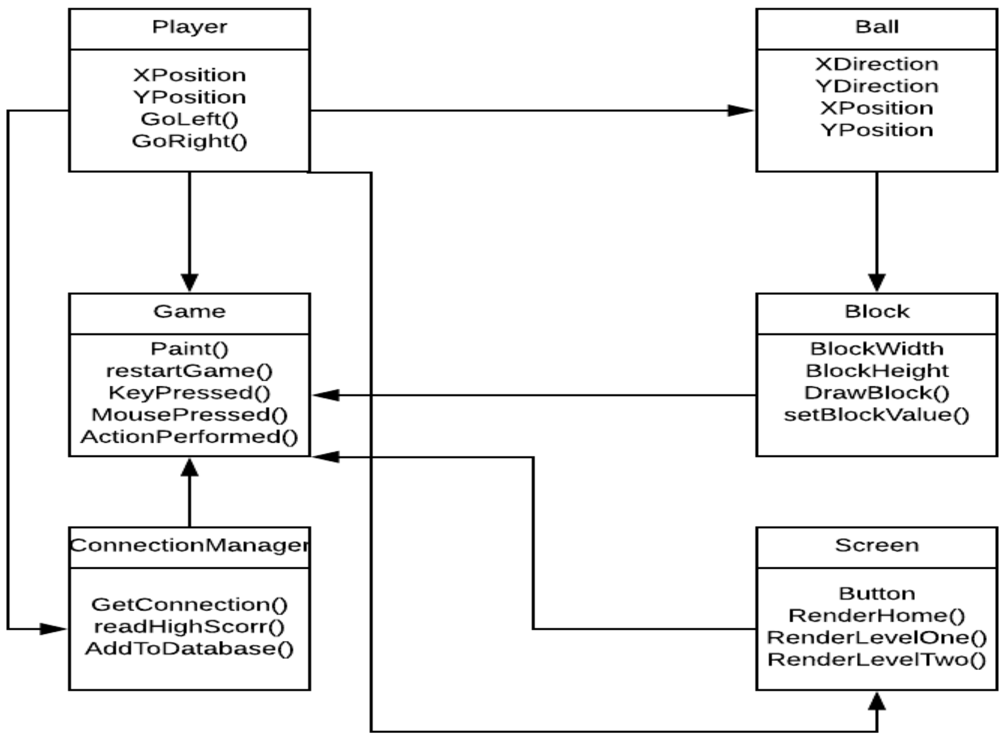
```
//adds a new score to the database
public static void addNewEntryToDatabase(int highscore,String username)
{
    try
    {
        con = ConnectionManager.getConnection();
        stmt = con.createStatement();

        String queryString = "insert into dbo.highscores (username, score) values (?,?)";

        PreparedStatement preparedStatement = con.prepareStatement(queryString);
        preparedStatement.setString(1,username);
        preparedStatement.setInt(2,highscore);
   //preparedStatement.setTimestamp(3, getCurrentTimeStamp());
        // execute insert SQL stetement
        preparedStatement .executeUpdate();
    }
    catch (Exception e)
    {
        System.out.println(e.getMessage());
    }
    finally
    {
        if (rs != null) try { rs.close(); } catch(Exception e) {}
        if (stmt != null) try { stmt.close(); } catch(Exception e) {}
        if (con != null) try { con.close(); } catch(Exception e) {}
    }

}
```

5 High-Level Design

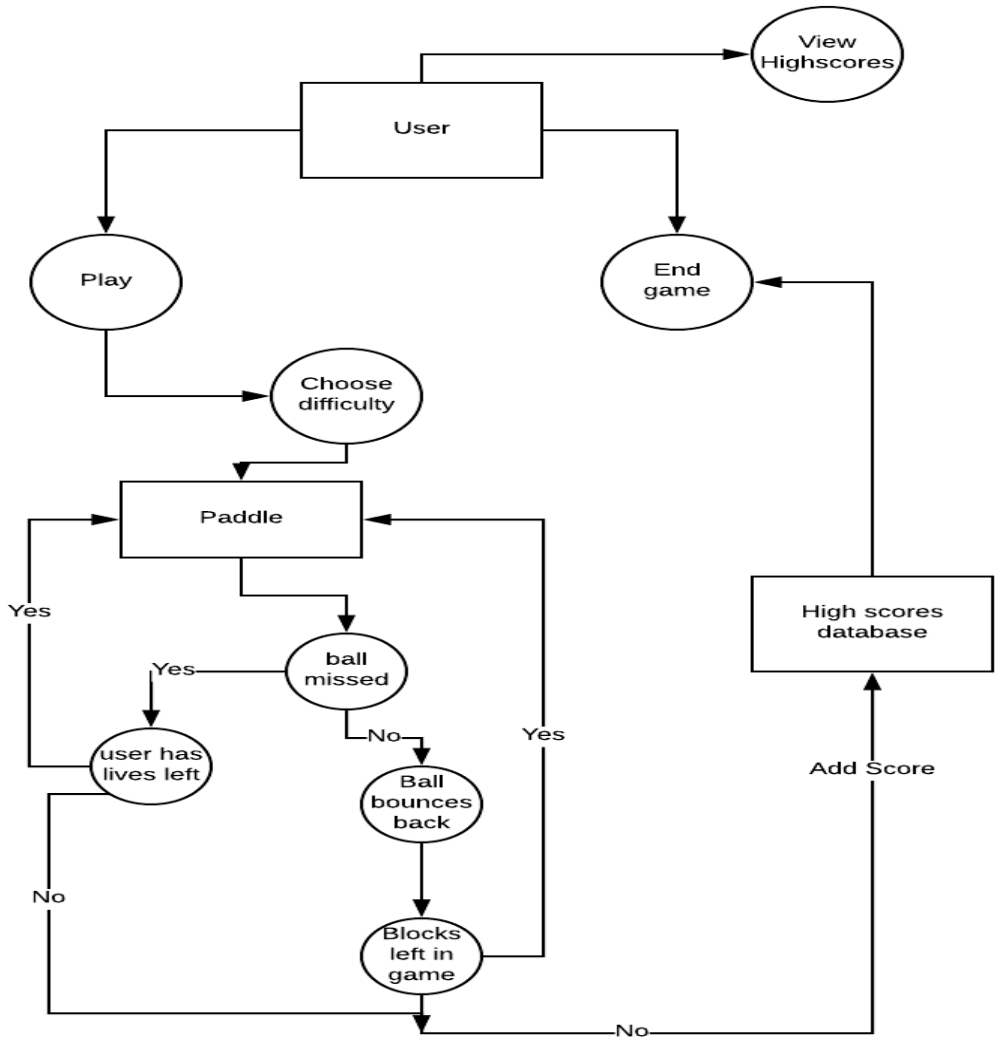
5.1 Context Diagram



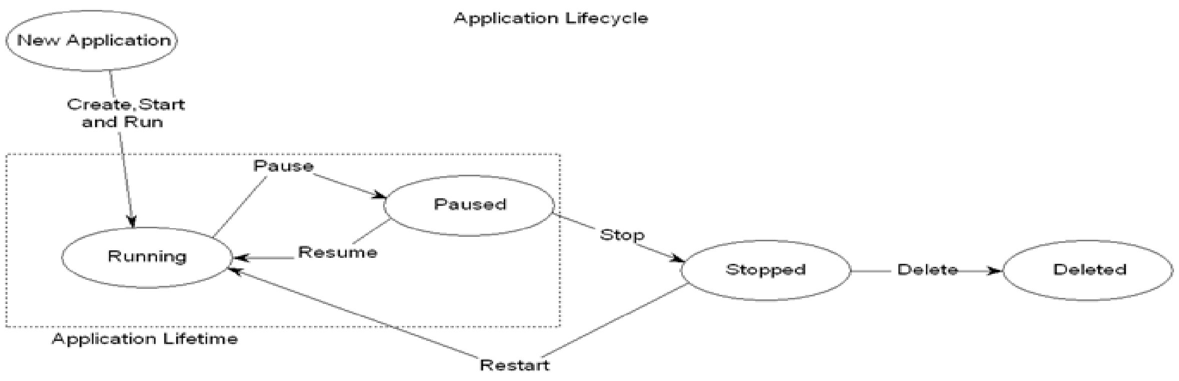
5.2 Object Model



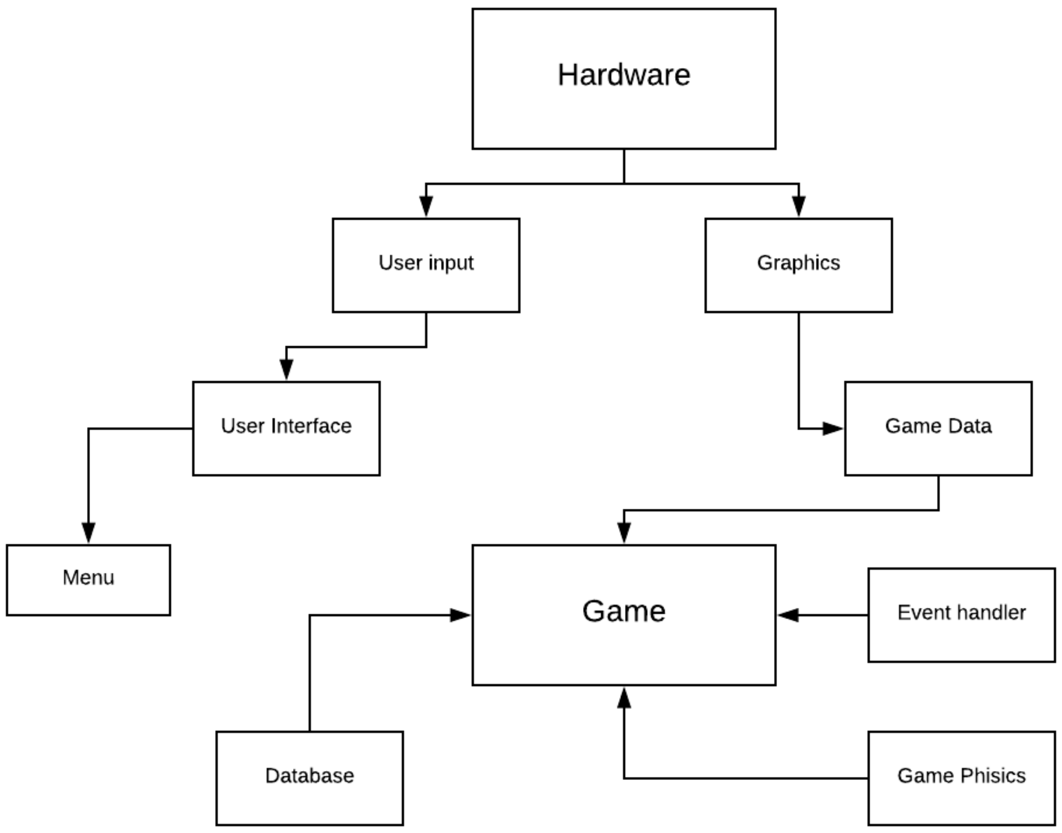
5.3 Data Flow Diagram



5.4 Application life cycle



6 System Architecture



7 Installation Guide

7.1 System requirements

- 64-bit (x64) processor
- Not much memory is needed
- No internet access is needed

7.2 Overview of system

- Uses Microsoft SQL server management studio for the database
- Configuration files used for speed, score and for database queries.
- Java was used to develop game.

7.3 Set up configurations

- Download and install Eclipse IDE, this is what we used for the development of our game
- The newest version of Java is needed
- Download sqljdbc42.jar and put it in an external jar folder
- Config.properties is in the res folder
- Run the game in Eclipse and the JFrame will open.

8 Results & future work

8.1 Results

From our experience working on our project we believe it went well. As this was our first time doing a large-scale project, we felt we were well organised and got everything finished that we would have liked. We believe that by setting ourselves realistic targets this lead to us finishing the project in the small time-frame we had.

From working on this project, we can say we learned a lot. We learned how to connect our project to a database which is something completely new to both members of the group. We learned about configuration files and how to use them appropriately. We felt the difficulty choice added an extra element to our game and the configuration file made that possible. We learned about the importance of documentation and keeping track of your progress throughout the timeline of the project. We believe that this project will certainly help when we must do our fourth-year projects and can take what we learned into consideration next year.

8.2 Future work

If we were to continue working on the project or maybe had a bigger timeline to work on the project here are some features, we would like to add to our current project.

- Put our database on the cloud, this isn't a necessity, but we feel it would be a nice feature
- Add more levels to the current ones
- Possibly implement a login feature
- Add some extra features (such as extra ball is added on level 2)
- If we added more levels, then we would like to add the possibility for the user to save their progress