8 Ball Pool Program
Maintenance Documentation

Callum Hunter
Liam Chapman
John Kim

CONTENTS:

1. INTRODUCTION

1.1. Purpose

This documentation has been provided to aid in the maintenance of the 8 Ball Pool program which has been developed for a university course project.

1.2. Point of Contact

The developers for this program can be contacted via Github:

Callum Hunter - https://github.com/Hunca
Liam Chapman - https://github.com/LiamNChapman
John Kim - https://github.com/kimkw448

1.3. Project Reference

8th April 2019
- Written report on the project planning submitted.

27th May 2019
- Delivered the alpha release of the program.

12th August 2019
- Delivered the beta release of the program.

30th September 2019
- Delivered the final release of the program.

1.4. Glossary

SFML - Simple and Fast Multimedia Library
VSC - Visual Studio Code
MingW - Minimalist GNU for Windows

2. PROGRAM DESCRIPTION

2.1. Program Application

This 8 Ball Pool program was developed for the purpose of our year long project which was to create a program under the given constraints of 1000 lines of code or less with the theme ‘Small is beautiful’. This program has been developed in an attempt to provide a simple, yet entertaining game for up to two users to play in an offline environment.

2.2. Program Functionality

2.2.1. Input

The program will run and take user input from the keyboard and mouse.

- *Keyboard inputs:*

Arrow Keys	Alters power input of the shot. Also moves the cue ball when a foul has been made.
Spacebar	Initiates the shot to be made. Also places the cue ball in position.
R	Resets the game state to the start state.

- *Mouse Inputs:*

Left Button Hold	Used to select the cue for rotation and used to set the power for the shot on the power meter.
Mouse Movement	The movement of the mouse will be used to position the cue around the cue ball for the desired direction of the shot.

2.2.2. Display

The program will initially open to show the start game menu consisting of the 'Play' and 'Exit' buttons.
During play, the program will have a display on the top left corner of the game window.

- Player | Displays the player turn.
- Suit | Displays player suit (Solid or Striped)
- Remaining | Number of player balls remaining table.

On screen text will instruct players when the cue ball needs to be moved.
A cue aimer will display the line of the shot from the cue ball in relation to the direction of the cue.
The power meter is shown on the left hand side of the program window.

2.2.3. Rules

- Player one will initiate the break. They are given freedom to place the cue ball anywhere in a legal position behind the D line before they initiate the break.
- If a player sinks a ball, the suit of the ball is allocated to that player and the opposing suit is allocated to the opponent.
    - If two balls are sunk, the suit of the first of the two will be allocated to the player.
- For sinking their suited ball, a player will be given another turn.
    - If two or more of the players suited balls are sunk, then they will still only receive one extra turn.
    - If an opponent's suited ball is sunk in conjunction to the current players own suited ball, then the player's turn ends.
- For sinking an opponents ball, the current players turn ends.
- Fouls for sinking the cue ball will end the current players turn, allowing the opponent to place the cue ball anywhere on the table before they take their shot.
- Fouls for failing to hit any ball on the table with the cue ball will allow the next player to place the ball anywhere on the table before their shot is taken.
- If a player sinks the 8 ball before sinking all their allocated suited balls, that player will lose the game.
- While the table has the cue ball and the 8 ball remaining, if the white ball is sunk, then the player who sunk the white ball will lose the game.

3. SUPPORT ENVIRONMENT

3.1. Equipment Environment

3.1.1. Computer Hardware

The minimum requirement for users are computer systems that are newer than 2010 models. A monitor for display and keyboard and mouse inputs are required. You will also want to be connected to the internet for the purpose of downloading the necessary support software and the project source code.

3.2. Support Software

This program was written using the C++ coding language on Windows(x86) based computers. For the Windows release we include a VSC project, whereas for MacOS(x64) users, we include an Xcode project folder containing a MacOS compatible version of our program.

For both Windows and MacOS versions, we include compatible SFML libraries. On Windows, Make and MingW are used for

compiling the program, whereas on MacOS, an inbuilt function in Xcode will build the program.
A graphics editor of choice may be used in the case where sprites are required to be updated.

Visual Studio Code:
https://code.visualstudio.com/download
Xcode:
https://apps.apple.com/us/app/xcode/id497799835?mt=12
SFML:
https://www.sfml-dev.org/download.php
Make:
https://www.gnu.org/software/make/
MingW Configuration:
https://code.visualstudio.com/docs/cpp/config-mingw
GIMP (free graphics editor):
https://www.gimp.org/downloads/

3.3. Personnel

Individuals who wish to maintain this program should have an understanding of either the Windows or MacOS operating systems. They should have knowledge on the C++ coding language and should be familiar with the use of SFML libraries. For developers using Windows, they should also be familiar with using Make and VSC set up with MingW (instructions are provided in the web addresses in the previous section. Developers wishing to update the sprites should have experience using their graphics editor of choice.

4. PROGRAM MAINTENANCE PROCEDURES

4.1. Conventions

Throughout the source code, we will be using the K & R layout as shown below:

```
void checknegative(x) {
    if (x < 0) {
        puts("Negative");
    } else {
        nonnegative(x);
    }
}
```

Inline commenting is used for clarifying certain sections of the code.

- NB: The README.md document contains additional information regarding the functionality of methods and the program.

4.2. Verification

All of our inputs were sanitized and verified before any action in the program was taken. This ensured that all inputs were known before any processing took place.

4.3. Error Conditions

While maintaining this program, we expect developers to encounter errors should they decide to use Xcode on MacOS. This is primarily due to our program being developed on a Windows operating system and so we expect users to have portability issues.
To get the project to port over to Xcode, it requires all assets to be inside the Xcode project folder. It also requires all .cpp and .h files to be added into the ‘Compile Sources’ which is found under the ‘Build Settings’ in the Xcode Project.

4.4. Maintenance Procedure

The following section provides help for possible errors that a developer may encounter with this program:

- In the case of encountering input errors:
  All input processing and validation is done in the PlayerManager class and should be the first place to review and analyse for errors while debugging input.
  All input handlers are given through the SFML Framework and this is done by detecting when there are keyboard and/or mouse inputs or through the activation of specific events.

- In the case of encountering game physics errors:
  Physics components are found in the Physics class and the MovementManager class. Developers should review their changes in these methods, should an error occur regarding collision or movement of the ball objects (ball on ball collision, ball on wall collision, and ball in socket interaction).

- Game rules error:
  Majority of the game rules are handled in the GameManager class and are found in the BallSunk method. Depending on the current player and which ball is sunk, different variables on the player object are modified to enact a rule. When these variables are changed the BallSunk method can determine who has the next turn. Rules not involving balls being sunk are handled in the movement management class (fouls).