

**This exemplar Unit 3 computing project was produced
by a Year 13 student**

Exemplar Project

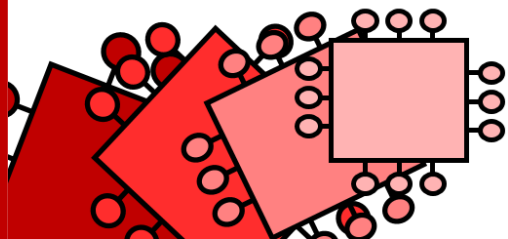
The project was marked at 62/70.

The mark was not adjusted during moderation.

Along with this document please see the following two files:

- Candidate 1 - (62 out of 70) - Additional comments.pdf
- Candidate 1 - (62 out of 70) - Mark Grid.pdf

These files show you, in detail how the best fit marking approach was applied to the actual marking criteria and provides an example of the additional commentary which was sent to the moderator to justify the teachers marks.



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

OCR GCE A COMPUTER SCIENCE PROJECT H446-03

Name :

[REDACTED]

Candidate Number :

[REDACTED]

Archway School :

[REDACTED]

Title of Project :

Penalty Shootout Game

H446-03 – PROJECT CONTENTS

TABLE OF CONTENTS

A. Analysis	4
B. Design	22
Systems diagram	22
C. Developing the coded solution (“The development story”)	43
D. Evaluation	123
Project Appendixes	146

A. ANALYSIS

DEFINING THE PROBLEM AND STAKEHOLDERS

PENALTY SHOOTOUT OUTLINE

My game is a two player penalty shootout, where one person is in goal and has the objective to prevent the other person scoring a goal by anticipating where the shooter is going to hit the football. The other person's aim is to score a goal by clicking a section of the ball which will determine where the shot will end up, if the goal is scored then the shooters points are incremented by one. Once the player has taken a single shot no matter if they score or not the shooter and the goalkeeper switch places. The first person to five points win, however if there is a draw then the shootout comes down to golden goal.

CONTROLS FOR THE SOLUTION

The games are easy to follow for anyone of a younger age or has limited knowledge of knowing how to use a computer, the keys used are the "W, A, S, D" for the goalkeeper and the shooter will be using the "LEFT CLICK" on the mouse.

TARGET AUDIENCE

The penalty shootout is a one vs one competitive sports game, which has a target audience of young boys between the ages of 5-14. The platform for this game is PC as it is the most suitable for the target market for instance if the player cannot afford a gaming console. The audience of the penalty shootout is aimed at is between the ages 5-14 as children start playing games on the computer around the age of 5. In addition, the capped age is 14 as the game is not going to appeal to teenagers and as much as it will to children as there are console games out there. The game shall have an easy to follow process as the menu will be simple with an option to show the users the controls they need in order to play the game. The game cannot be too complex as my chosen audience may struggle to follow otherwise which would lead the game being unsuccessful as not many people would invest their time in playing.

JUSTIFICATION OF HOW TO SOLVE THE PROBLEM CAN BE SOLVED BY COMPUTATION METHODS

THINKING ABSTRACTLY

My game will greatly be abstracted, this is due to the fact that the game needs a point system and other essential features to making the game suitable and simple for my stakeholders (aged 5-14).

Features of abstraction within the penalty shootout game:

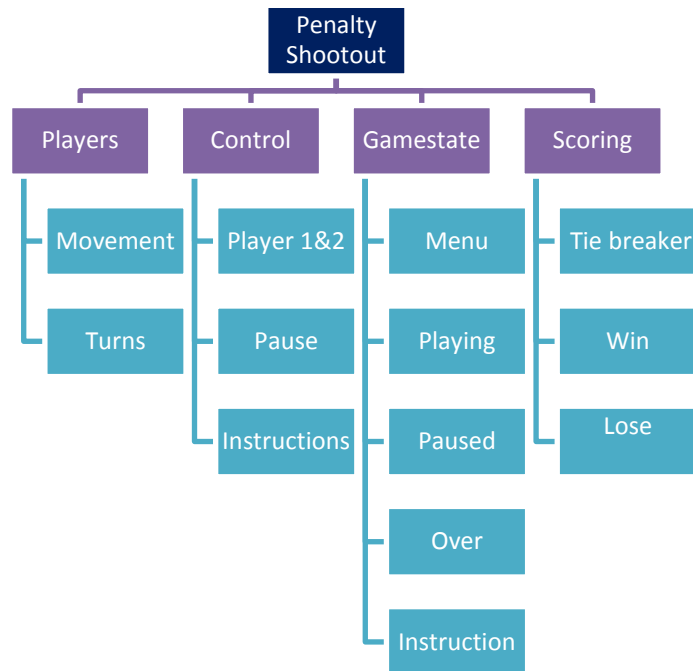
- Score board, this feature will show the users how many points they currently hold against their opponent. However in football there are normally some form of score boards however my game will have an animated version which will therefore be an abstraction.
- Animations, this is as the graphics will be way off from reality, the animations will be created using sprites. The animations are also more appealing for my stockholder as they won't be looking for the most realistic game on the market.
- Winning banner, once a player has won the game a banner will be displayed which will say "PLAYER X IS THE WINNER", this will clearly shows that the games state is in game over. This is abstraction as there is no banner saying who won, also in football there are teams not player one's and player two's.
- Pause button, the pause feature is a huge factor within the game as it allows the user to leave the game possibly do some homework, chores or go for tea and come back to the game. This is a features best suited for my audience as they are at the age when they have to do homework on a regular basis ECT... This is abstraction as there is no pause button in a real football match.

THINKING AHEAD

- Controls, the controls need to be easy to access for example if player one needed to use the mouse's "left click" as well as "WASD" and player two needed to use the "arrows", this shows that the creator hasn't thought ahead about the composition of the controls. Due to this error the audience are not going to enjoy the game to its full potential as the controls are awkwardly positioned. Although this seems like a minor issue it can still lead to a great error; thinking ahead would of allowed to designer to easily location the problem and solve it accordingly.
- Pitch layout, the layout needs to be suitable for the users to give a more realistic effect, if you write the program to certain coordinates and then decide to use a more suitable pitch layout you will struggle as the pitch may not fit the scale to what you have written your game to. Thinking ahead allows to game to be programed to fit the pitch.
- Scoring system, if the scoring system is not effective then the game will not be as good as intended. This is as the game needs a simple, clear and fair system which identifies and solves tie breaker. This way of thinking ahead reduces conflict over who the real winner is.

THINKING PROCEDURALLY AND DECOMPOSITION

To break down my problem I am using a top down modular design



I have broken my game down to four main problems that need to be solved in order for the game to function properly, these are:

- Players, as they are key for the game to even work so they are a necessity and these are based around movement via the controls and in what order that they have their turns as the shooters.
- Controls, these need to be programmed correctly as they essentially control the players and can make or break the game. There will also be a pause button to allow the users to take a break if required.
- Gamestates, this is important as it takes the game to different stages, the menu state will display the main menu where the player can select “play”, “instructions” or “exit”. The playing state is when the game is under way and being played, when the state is paused, the paused screen will be loaded until the playing state is resumed. Finally, the game over state is when the “PLAYER X IS WINNER” banner is loaded up.
- Scoring, is import as it holds purpose for the game as it determines the winner of the game and can alter the game’s outcome. For example if the points are at a draw then a tie breaker will be run to find the winner. The score board also increases pressure for the players due to the fact that they could lose.

THINKING LOGICALLY

- The main menu, this is a decision point as there will be options to “Play”, open the “Instructions” and to “Exit”. This is thinking logically as there is a decision that needs to be made and the game will loop the game state of menu which will run until a decision is made.
- When the football is hit, this is a decision point as the game loops until the direction and power is applied to the ball. Once selected to ball heads to the area of the goal proceeding with the program.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

- The game over state is looping and waiting constantly until there is a winner, once there is a winner the code will once again proceed and display the screen for game won.
- There is also a decision to be made when the keeper goes to save the ball as they need to go a certain direction to save the ball and previous shot.

THINKING CONCURRENTLY

- The players switch places while the scoreboard is being incremented or left the same, this reduces the delay between how long the user has to wait before taking their shot at goal. This is thinking concurrently as the program is running two processes simultaneously which increases the games effectiveness to provide a good gaming experience for the target audience.

RESEARCH

Note: This is the closest reference I could get to my game although they are largely different, however other games are unavailable to me while making this project as they are block within the school.

FIFA 16 GAME CAPARISON

Fifa16, the basis of the penalty shootout within this game are fairly similar to my penalty shootout, in fifa 16 there is a bar in the bottom right corner of the screen which has a slider constantly moving across it, this determines how much power you will apply to the ball. If you land the slider on the green section of the bar then the power applied is at its best possible, if you hit the yellow/amber section the power applied is slightly higher so you need to aim the ball more carefully. Finally if the bar lands in the red then too much power is applied which means the shot will miss the goal.



This is what the bar looks like

In the penalty shootout in Fifa 16 just above the power bar is the players' name, who is taking the penalty, and in the bottom left hand side is the name of the goal keeper which will attempt to save the shot. This differs from my game as I won't have any names for my players, this will allow my game to have a simple complex which is more suited for my target audience as they may not know who the players are. Next to this displayed name is a badge for which club that player is a part of, once again I shall not implement this element as the audience may not know what the club is called that player is contracted to.



Name and club badges

The graphics in Fifa 16 are a lot better than the graphics in mine as I will be using sprites and cartoon animation as I believe that cartoons will entertain my target audience more than graphics that represent reality to a greater level.



This image shows how the graphics within Fifa 16 represent reality as much as possible

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

As fifa 16 is trying to represent reality as much as possible it means that they want to have a crowd within the background, as my aim is not to represent reality to a certain level but to provide a game suitable for 5-14 year olds I shall not be using a background with a crowd which moves. I shall be using a still cartoon sprite background which is suitable for the players, to determine what design of pitch I should have I will ask a participant what their options are, and if there are any improvements that I can act upon for my game.



This shows the crowd behind the goal and how it increases the effect of reality

Fifa as has advanced mechanics within in the game as you are able to see shadows which correspond with the movements of the players and the ball, whereas mine does not have that level of speciality. In addition to the shadows Fifa also has realistic movement effects, for example the ball moves in a realistic manor and so do the players resented in the game, and my game simply doesn't reach that level of complexity.



This shows the shadows from the ball, goalkeeper and goal posts.

INTERVIEW PLAN

Main points

- What makes a game successful?
- How many players should be able to play?
- How important is design?

Topic development

What makes a game successful?

- 1) Does the game need to represent reality to make it a success?
- 2) Should my game have sound effects?
- 3) Would you prefer a tie breaker system or should the game be left as a draw?
- 4) What would you say a reasonable score limit for winning is?
- 5) Would you like your player to be personal? (E.g. you can give them a name)
- 6) Do you think an instruction menu would reduce confusion?

How many players should be in the game?

- 1) Do you prefer games which you can play with your friends?
- 2) Do you like competitive games?
- 3) Would you prefer a game which allows you to have the computer to yourself?

How important is design?

- 1) Does a game's look determine whether you play it or not?
- 2) Which background do you prefer?

A)



B)



C)



B)

INTERVIEW TRANSCRIPT WITH BILLY EVANS

Section1-Successful games

Q Does the game need to represent reality to make it a success?

A game should be colourful and exciting, so it doesn't need to be realistic.

Q Should my game have sound effects?

A It would be more interesting with a sound effects, you could have a cheering sound effect for when a player scores and a booing for when the player misses.

Q Would you prefer a tie breaker system or should the game be left as a draw?

A It might be better as a draw, as it means the game is less complicated.

Q What would you say a reasonable score limit for winning is?

A Around five as it's not too high and not too low.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Q Would you like your player to be personal? (E.g. you can give them a name)

Yeah, I think that would make the game better as it means the game will be more adapted to me.

Q Do you think an instruction menu would reduce confusion?

Yes it would reduce confusion, I am always trying to find the controls on other computer games.

Section2-Number of players

Q Do you prefer games which you can play with your friends?

Yeah, I like to play games with my friends as it makes games more fun.

Q Do you like competitive games?

Yeah it makes the game more interesting and enjoyable.

Q Would you prefer a game which allows you to have the computer to yourself?

No, as it's probably better with two players as it means I can play with my friends.

Section3-Design

Q Does a games look determine whether you play it or not?

I would rather play a game that is colourful and exciting instead of a game which doesn't look fun.

Q Which background do you prefer?

A, as it's a stadium so it will be more professional.

A)



B)



C)



INTERVIEW DATA REVIEW

Requirements-basics

Graphics: Cartoon animation, this is more appealing to my target audience as they don't look for high quality representations of reality within a game. The cartoons will be taken from a sprite sheet and those sprites will be moved across the screen by redrawing the sprite to a different coordinate, this will give the illusion of the sprite moving.

Sound effects: Cheering when goal is scored or game is won, booing when the shot is saved or missed. The cheering sound gives the player a sense of achievement which will lead to them wanting to play the game again. The booing and cheering are both real occurrences which give the impression that the player is more involved in the game.

Tie breakers: The golden goal idea won't be implicated within my game as it would increase confusion and mainly because it would be too difficult to program with the skills I have. On top of

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

that, with the time limit for when the project has to be completed, I believe it's just not possible for me to use the golden goal feature.

Score limits: The first person to reach five points wins, however if there is a draw at the end of the five shots the players have then the game ends as a draw, this is due to the fact that I will not be able to implement the golden goal feature.

Characters: You have the option to pick a name for you player to include a personal aspect which will hook the player into the game, to ensure that there is a higher chance they will return to play this game again.

Instruction menu: Make a clear and effective instruction page to show the user what the controls are, this is important as the target audience are between the ages of 5-14 therefore they need clear instructions so they are not confused.

Number of players: The game will be one player unfortunately as I am unable to ensure that the game will function correctly with two players as I simply don't have the programming ability within the time frame I have. Due to this the player will be playing against the computer instead of their friends.

Competitive: This is a must for my game as football is a competitive sport regardless of reality, therefore the player should have an equal opponent, and this means that difficulty levels should be used to entertain the player. This way there ability is matched and the game is made to be more competitive.

Players: Unfortunately, as explained before the multiplayer feature shall not be implemented into my game.

Game appearance: The game has to look appealing to young children therefore it shall be bright and enticing, I plan for the game to be animated which should welcome children to play it.

Background: The background is one with the goal in the stadium which gives a more realistic look, the background shows the standers where the crowd will be watching.

Scoring

The score board will be in the top right of the screen and will have a bar of ten white circles in total, five for the computer and five for the person playing, when a goal is score the left most white circle for a set of five turns green, and if the shot is missed then the left most white circle for a set of five turns red. To the left of one of the sets of five circles is the players gamer tag that they wish to enter, and to the left of the five circles of the other set is the gamer name CPU so they know they are playing the computer.



This is a potential scoreboard as it's in the early stages of development, the model to the far left is what a scoreboard could look like after a game, as shown they scored two out of the five shots. The middle and right score boards are what the computer's and the players score boards will look like. The font used for the gamer tags is called 'Showcard gothic'. As the player is able to customize their gamer tag, which will be selected as soon as the game begins, the name next to the score board will be affected by the entered name.

The end screen

When the player wins the game will switch to a different screen where it either says PLAYER X WINS with a cheering sound affect or to a screen which says UNLUCK TRY AGAIN with a booing sound effect. The banners will come with a 'return' button in the bottom right of the screen which will take them back to the main menu, where they are able to restart the game, open the instruction menu or exit. The final way to end the game is with a draw, so when a draw occurs the end banner is displayed and there is a sound effect that was running during the game. This sound effect will be a backing song to give the game more of a vibe.

Game states*Menu*

This is a state where the home menu is displayed which holds a list of options to choose from. The first begin 'PLAY', which will take the player into the playing state where the game is under way, the first thing the player does when the game starts is to enter a gamer tag for their player. The second in the list is 'INSTRUCTIONS' which will take the user to a screen where the instructions are clear to interpret, the screen will consist of a picture the control and an arrow pointing to the component explain its function.

Playing

This is the stage where the game is underway, at the very start of the state the player is asked to input a gamer tag for their character, the playing state lasts as long as the game runs for unless the user enters the paused game state.

Paused

This a state where the games progress can be frozen while the user attends to other activities, within this state there is an option to return to the home menu or to continue with the game.

Over

See end screen above.

Instructions

The instruction state displays a clear diagram of what the controls are for the game, there will also be a home button in the bottom right of the screen. The control diagram will consist of a keyboard with the keys used to play the game highlighted and pointed to by an arrow, the same will happen for the mouse.

Controls

Controls for the goalkeeper are as follows:

'Left Arrow'- moves the goalkeeper left

'Right Arrow'-moves the goalkeeper right

“(Space bar) - Makes the goalkeeper jump for a period of time

Controls for the shooter are as follows:

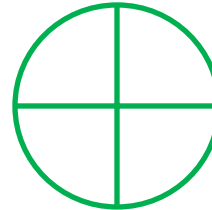
“(space bar)- Applies power to the shot

'A' moves the crosshairs for the shot to the left

'D' moves the crosshairs for the shot to the right

'S' moves the crosshairs for the shot to the down

'W' moves the crosshairs for the shot to the up



*Crosshair's to show
where the shot is going
to end up*

Other controls:

In order to pause the game hit the letter 'p' and the pause menu will be displayed, in order to navigate through the menu's use the 'LMC'(left mouse click) this will work for selecting a function are the main menu where the instructions, exit and play options are located. The LMC also works on return buttons at the pause menu and on the home button when the end page is shown.

Movement

The goal keeper can move left when the left arrow is pressed, right when the right arrow is pressed and finally the keeper is able to jump when the spacebar is pressed. The ball movement is dependent on where the shot is positioned, the ball will make a direct line towards the point where the outcome is intended. If the goal is saved the ball will return along the path it had just taken to show the illusion of the shot being deflected on the contrary if the ball goes in the goal the ball will drop to the bottom of the net and the white circle dedicated to that shot will turn green. If the shot was missed the white circle would of turned red to signify a missed penalty. The goalkeeper will take two seconds to move from one goal post to the other without stopping in between, in addition to this the ball will take one second to reach the goal from the penalty spot once kicked.

Turns

My game is a turn bases penalty shootout therefore the turns of the user has to be identified and determined to make the game a success. The player has the first shot at goal and once there shot has been taken the player then switches to the goal keeper where they will attempt to save the computers shot. When to player switches to the goal keeper the CPU's scoreboard will be activated so the outcome of the shot can be marked down. Each turn should roughly be around fifteen seconds each in order to time the game moving and competitive, although there will not be any timer counting down the time expected is only an estimated.

View

The game will be from a first person perspective so you see the ball on the penalty spot and the goal from your screen. The camera view will also be in a still camera position so there won't be any sway on the display, the background itself will be a goal within a stadium.

Main menu

There will be a main menu where the first option on the top will be to 'play' the game, the second option beneath the first option is to open the 'instructions' page. Finally, the third option it to 'exit' the game.

Graphics

The game is a cartoon penalty shootout therefore sprites will be used in order to make the game animated.

Sound effects

The sound effects within the game are cheering when the winning page is loaded, booing when the game lost page is displayed and there will be a suitable backing track thought the whole game it will also be played on the end screen if the game turns out to be a draw.

FEATURES FOR THE PROPOSED SOLUTION**FEATURES EXPLAINED**

I have written up the requirements that are more than certainly going to be within my end product and will justify my reasons for including or excluding certain features.

Features	Justification/limitation
Cartoon graphics	As suggested by Billy within the interview a cartoon based game would appeal to my audience as it makes it bright and interesting.
Sound effects relevant to the game	In the interview the idea of sound effects giving a more realistic impression for the game was brought up.
Games ending in draws instead of having a tie breaker	This is due to the time frame and my limitation of programming skill where I simply wouldn't be able to complete a functional 'golden goal' feature as explained previously.
A score limitation of five points	During the interview I asked Billy what he deemed a suitable score limit for the game was and he gave the response of five as it's not too much which would end up dragging the game on and not too little the game will end too soon.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

There user will be able to customize their gamer tags	Billy thought that the game would be more adapted to him if this feature was implemented, this is a good sign as it shows that there is an interest within the game that may make the game better than the competitions.
A clear instruction menu	In the interview it was said that an instruction menu would reduce confusion which is an extremely important feature as if the user doesn't know how to play the game they are more than likely to move on to another game.
The game will be single player	Although in the interview with Billy he stated that playing games with friends is better I am unable to apply multiplayer features to my game as I don't have the programming ability within the time frame for this project.
The games appearance will be bright and colourful	Billy mentioned in the interview that he would rather play a game that stands out with colour and excitement instead of a game that looks dull in colour.
The game is set in a stadium	The stadium gives a more realistic impression for my game, the stadium was the background that Billy picked within the interview.
The score board will display the player's gamer tag next to it	This Makes it easy for the player to know which scoreboard is theirs and reduces confusion, this idea comes from Fifa16's penalty shootout.

SOFTWARE AND HARDWARE REQUIREMENTS

HARDWARE

Item	Justification
Monitor	Allows the player to spectate the game.
Keyboard	Required so the user can use the controls to save the goal and to shoot.
Mouse	Allows the user to navigate through the menu of the game.
Speakers/ headphones	In order for the player to hear the sound effects within the game.
2Mb of hard drive space	A single sprite is 15KB's in size, the background is roughly 4KB therefore I will not need so much memory. The backing sound is going to require space to run as well.
2Gb of RAM	The average computer holds 2Gb's of RAM, the likely scenario is that there is more an enough RAM than we actually need to run the game.
33MHz processor	The minimum requirement of processor to run the software.

SOFTWARE

Item	Justification
Monkey X run-time library	Allows the game to be run by the user without them actually having to download the monkey x themselves.
A platform which is any of the following: Windows Android Flash OS X IOS HTML5 Linux WP7/WP8 Xbox 360	These are the platforms that will effectively use the hardware in order to run the game
If you use windows desktop then you will need the following:	These are the applications to allow the game to run a Windows platform.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

<p>OpenGL API for graphics rendering</p> <ul style="list-style-type: none">• OpenAL API for audio• MinGW 4.8.1 or Microsoft Visual C++ Express 2010 (MSVC)• OpenAL Windows drivers	
<p>If you use Android then you will need the following:</p> <ul style="list-style-type: none">• Android SDK. You only need the "SDK Tools" version, not the "Eclipse+ADT Plugin" one• Java SE JDK 32bit version• Apache Ant Java library	<p>These are the applications to allow the game to run a Android platform.</p>
<p>If you use ios then you will need the following:</p> <ul style="list-style-type: none">• Apple Mac computer running OS X• XCode developer tools with iOS SDK	<p>These are the applications to allow the game to run a ios platform.</p>
<p>If you use HTML5 browser target then you will need the following:</p> <ul style="list-style-type: none">• An HTML5 capable browser such as Chrome, Firefox, Opera, Safari, Edge or IE	<p>These are the applications to allow the game to run a HTML5 platform.</p>

SUCCESS CRITERIA

REQUIREMENT SPECIFICATION

Requirement	Justification	Reference
Game has a 1 st person view from the person taking the penalty.	This gives the game a more realistic effect and I don't have the time to implement another moving character.	<i>Proposed solution</i>
Still camera view for the screen	This makes the game easier to play as it means the shots can be more accurate.	<i>Proposed solution</i>
The goal keeper moves at a speed where it takes two seconds to move from one goal post to the other	This is a reasonable speed as it's not as fast as the ball which means the person shooting has a very possible chance of scoring.	<i>Proposed solution</i>
The ball takes one second to move from the penalty spot to the goal	This is so the shooter has a higher chance of scoring than the goalkeeper has with saving the ball.	<i>Proposed solution</i>
The time for each player to have a turn shooting should be around fifteen seconds as long as they haven't paused the game	The game shouldn't have a slow atmosphere as it will appeal less to my target market, they would prefer a fast paced lively game.	<i>Proposed solution</i>
The game will be a single player game where you play the CPU	I won't be able to implement the multiplayer feature due to my ability in programming within the time frame I have for the end product to be made.	<i>Interview data requirements</i>
<ul style="list-style-type: none"> • " "(space bar)- Applies power to the shot • 'A' moves the crosshairs for the shot to the left • 'D' moves the crosshairs for the shot to the right • 'S' moves the crosshairs for the shot to the down • 'W' moves the crosshairs for the shot to the up 	<p>The space bar is an effective way of applying power to the ball, as the longer it is held down the more power is given.</p> <p>'W,A,S,D' is used inside of the arrows to avoid confusion with in the game.</p>	<i>Proposed solution</i>
'Left arrow'(move left) , 'Right arrow'(move right) and	These are simple and clear controls which mean my audience will be	<i>Proposed solution</i>

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

'space'(jump) will control the goal keeper	able to play the game without an issues which may prevent their interests in my game.	
'Left mouse click' is used to navigate through the menu's and return buttons	This is a simple and effective way of navigating through menus of the game.	<i>Proposed solution</i>
The instruction menu will have a diagram of the keyboard and mouse which are labelled with their functions	Billy stated in the interview that the instruction menu is an important feature as it reduces confusion, the labelled diagram is better than a list of instructions as it gives a visual aid which will appeal more to my audience.	<i>Interview data requirements</i>
'P' will pause the game until the game is resumed with a 'left mouse click' on the resume tab	The pause feature is a useful requirement as it allows the player to come back to the game, this means they are more likely to return for another go.	<i>Fifa 16</i>
The main menu will have three options of PLAY/INSTRUCTIONS/QUIT	These are the essential options as if the user is unable to play the game there is no point of creating it, these options are a clear way of directing the user to selecting what they wish to proceed with.	<i>Proposed solution</i>
When a goal is scored the relevant white circle will turn green and if the goal is missed or saved the circle will become red	This is the method I plan to use for the score board as it removes the number system of points and uses a more appealing visual score board which will be more enticing for my audience.	<i>Proposed solution</i>
When the game ends the end screen is displayed with cheering and 'PLAYER X WINS' if they win, and booing sound effects with 'BETTER LUCK NEXT TIME' if they lose. Finally, if the player draws with the CPU the banning 'IT'S A DRAW' appears and the backing track which will have been playing throughout the game will continue to play.	This shows achievement for the player once they win, and they are likely to play again to get feeling of achievement once again, on the contrary if they lose the phrase 'BETTER LUCK NEXT TIME' leads them to replay the game in order to attempt the win.	<i>Proposed solution</i>
Graphics are will be cartoon sprites	In the interview Billy informed that he would prefer to have a cartoon game as it looks more interesting for my target market instead of a reality representation.	<i>Interview data requirements</i>
There will be a suitable backing track	This backing track will reduce silence within the game which could	<i>Interview data requirements</i>

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

	possibly lead to the audience getting bored.	
There will be a maximum score limit of five	This score limit is one that Billy chose and it is a very reasonable one as there are not so many that the game drags on and not so little that the game ends too soon.	<i>Interview data requirements</i>
Customizable gamer tag	This feature is strongly looked upon by Billy and is also a feature used within Fifa 16 so I've decided to use it in order to improve my game.	<i>Interview data requirements and Fifa 16</i>
Background image of a goal within a stadium	Although my game isn't realistic as other games such as fifa 16 due to the cartoon graphics and other the movement effects etc, the stadium means that I can have an element of realism with the game. This is also the background which Billy picked within my interview as well.	<i>Interview data requirements</i>
To be bright and colourful	The game needs to have an enticing vibe to it in order to persuade children between the ages of 5 and 14 to play it, I believe that this can be done by using bright colours to appeal to children.	<i>Interview data requirements</i>

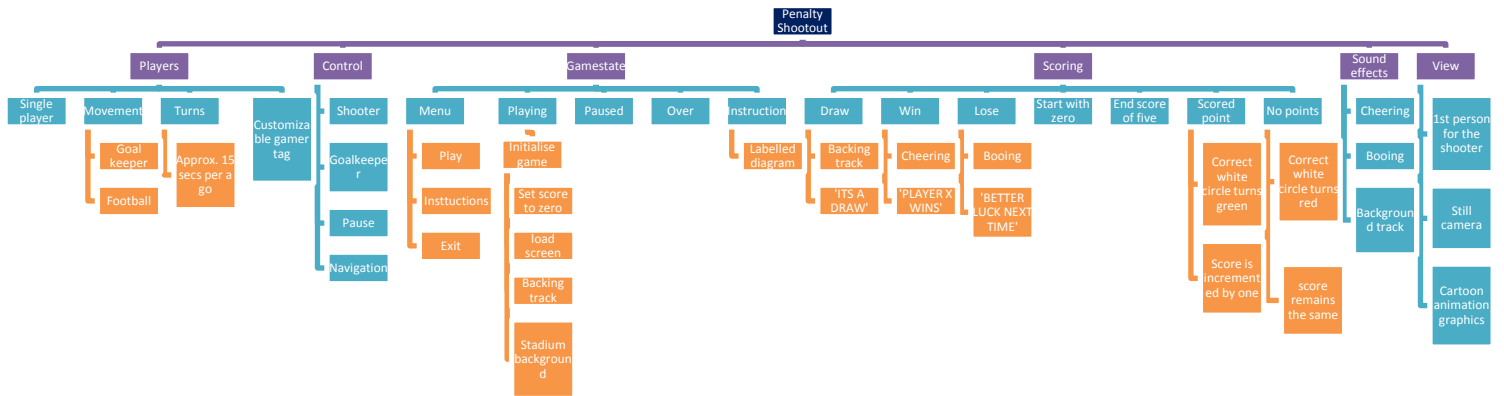
Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

B. DESIGN

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

SYSTEMS DIAGRAM



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

PROPOSED SCREEN DESIGN AND USABILITY FEATURES

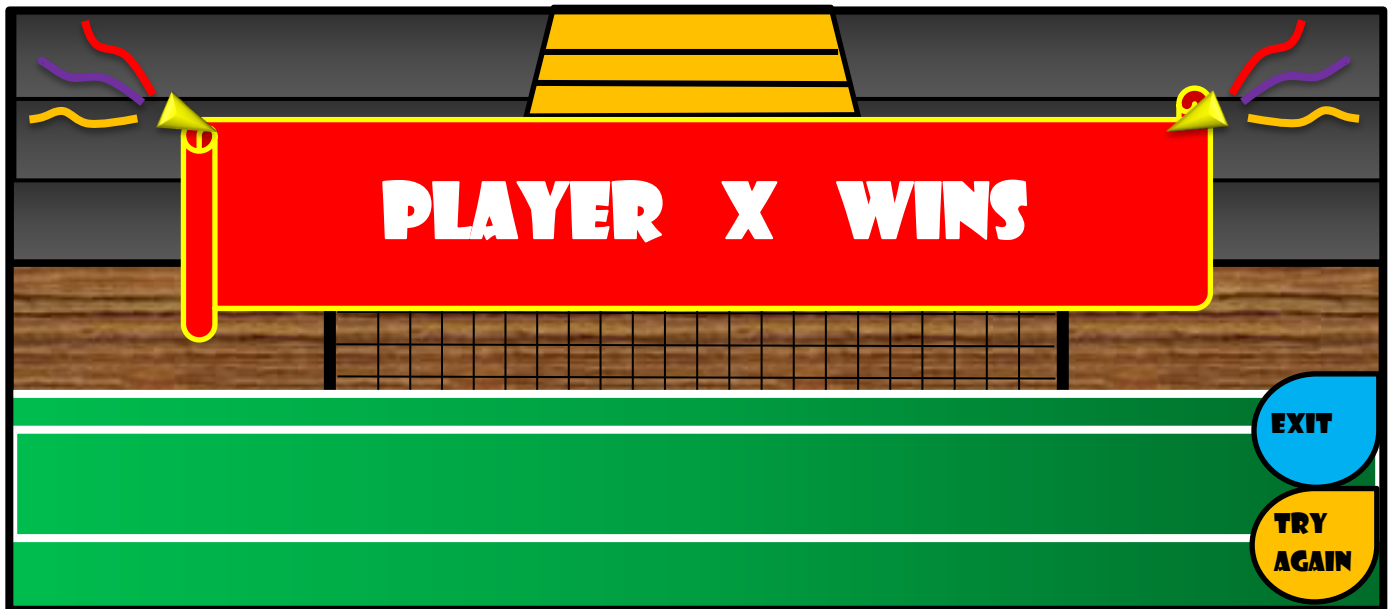
SCORING

Scoreboards



This shows the design of the scoreboard I intend on using for my game, the far left model shows what a potential game outcome could be, the middle model shows the CPU's scoreboard and the right shows the users scoreboard.

Winning end page

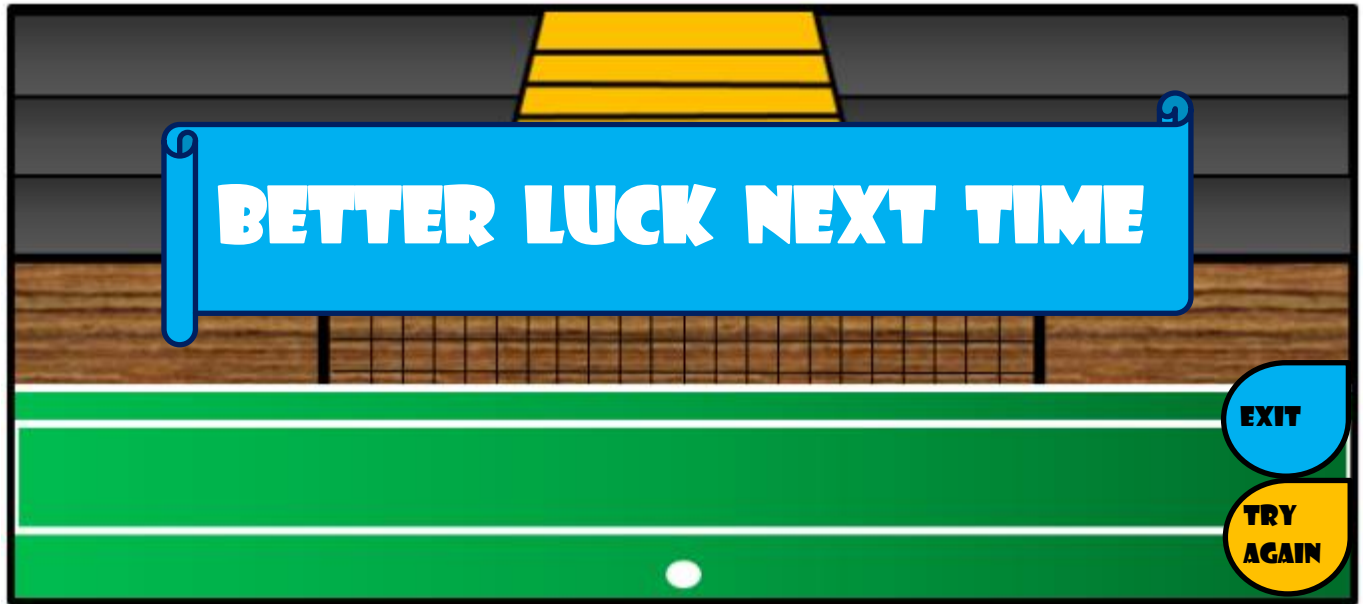


The banner uses contrasting colours which allow it to clearly stand out and to show an achievement, in addition to the winning screen there will be cheering music in the background. The confetti adds to achievement and it gives more of a difference between the winner and the draw pages.

Candidate Name: [REDACTED]

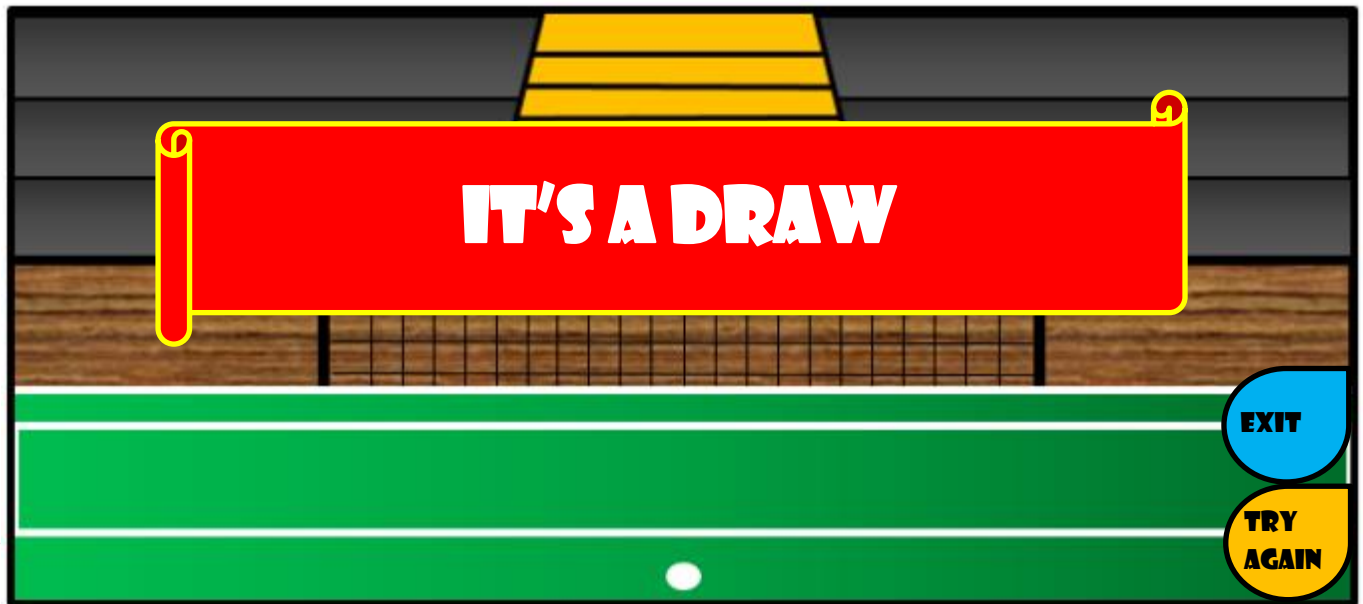
Candidate Number: [REDACTED]

Losing end page



This design uses the same basics as the game winning page such as the stadium theme, background colours, gradients, textures and fonts have been used. However I have changed the colour of the banner to a less inviting one as it shows that player has lost, the text also has an influence as it is telling the user that they will play the game again in order to succeed and win.

Draw end page

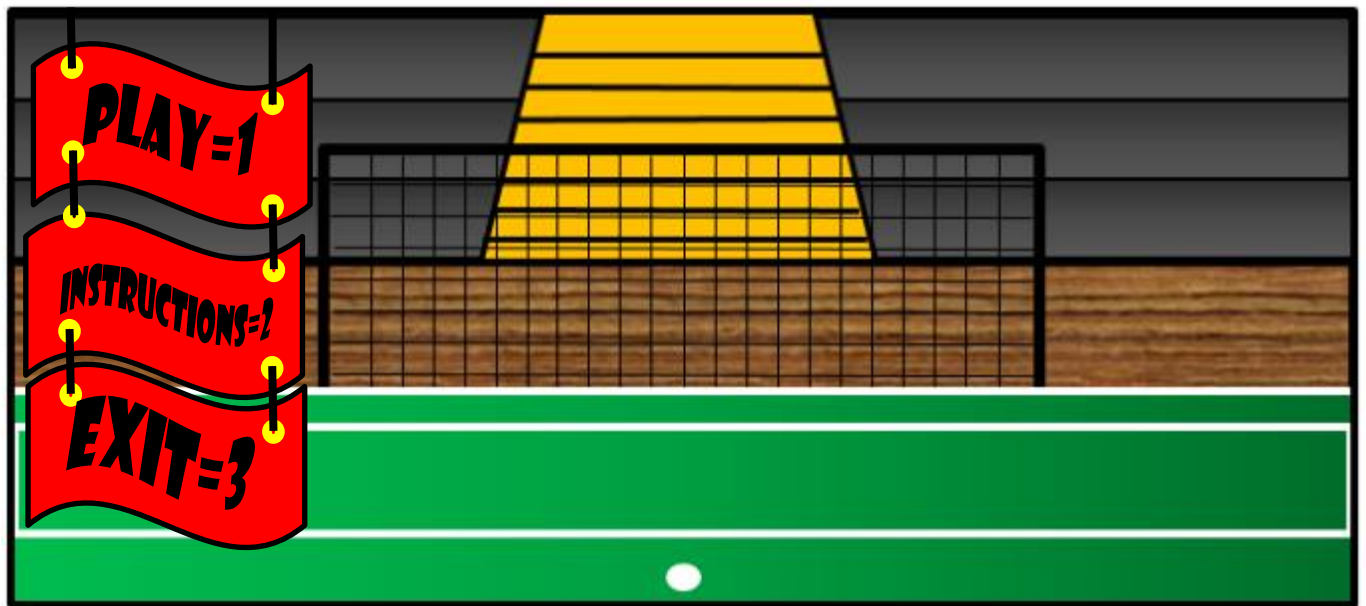


Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

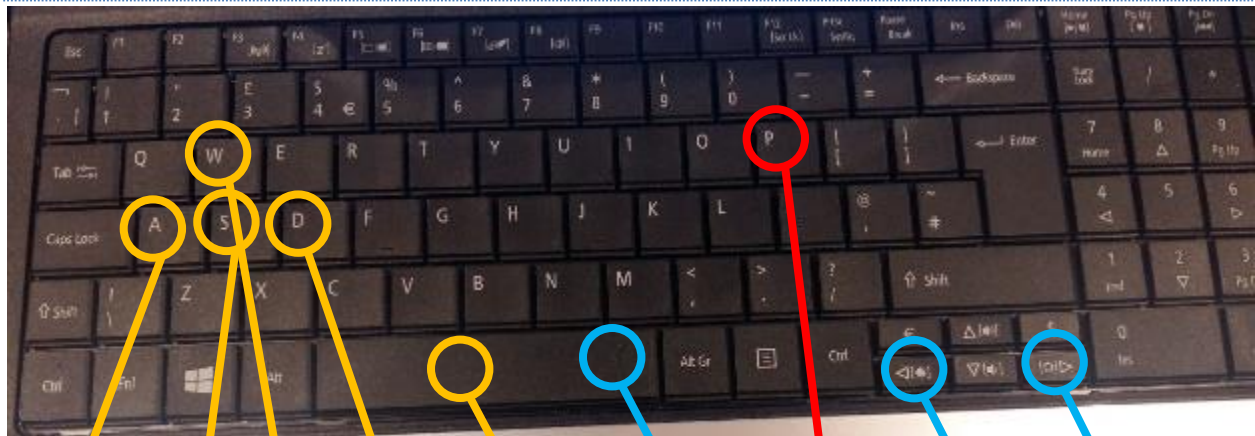
These designs are simple but effective as it clearly suits it's purpose, there are features within the design which are impacted by Billy and his desired outcome of the end product. The background was based upon Billy's choice to have the stadium background to give a more realistic impression, I used Fifa 16 with it's penalty shootout layout to create a design for my game. The pitch layout was one of the aspects I used from Fifa 16 as it shows understanding of the actual game of football, along with the stairs within the middle of the bleachers in the crowd, the symmetrical layout is aesthetically pleasing which will have a good influence on my target audience. I have used Microsoft Word to create this design, with this application I have utilised it's features taking advantage of the gradient tool, along with the textured images, I have used the gradient feature on the pitch showing a light effect, in addition I used this effect on the seats in the stadium to show shadows within the crowd. I used the texture tool upon the banner to show that it is made from wood to give a natural look, within the banner I used a font called 'SHOWCARD GOTHIC' which is one I believe suits the game and it's cartoon animated theme. All irrelevant detail has been removed, such as the scoreboards holding the current score along with the players and the crowd, this makes a more simplistic design which is one that I believe suits my target audience.

MENU



This design shows the options hanging from string to give a more interesting approach, I have used the colours red and yellow again as they are good contrasting colours and they brighten up the screen, the background shows the user the pitch which they will be using whilst playing the game so it keeps to the theme of football.

INSTRUCTION MENU



**A MAKES THE
CROSSHAIRS MOVE
TO THE LEFT**

**S MAKES THE
CROSSHAIRS MOVE
DOWN**

**W MAKES THE
CROSSHAIRS MOVE
UP**

**D MAKES THE
CROSSHAIRS MOVE
TO THE RIGHT**

**SPACE BAR APPLIES
POWER TO THE
SHOT**

**SPACE BAR MAKES
THE GOALKEEPER
JUMP**

P, PAUSE

**RIGHT ARROW
MOVES THE
GOALKEEPER
RIGHT**

**LEFT ARROW
MOVES THE
GOALKEEPER LEFT**



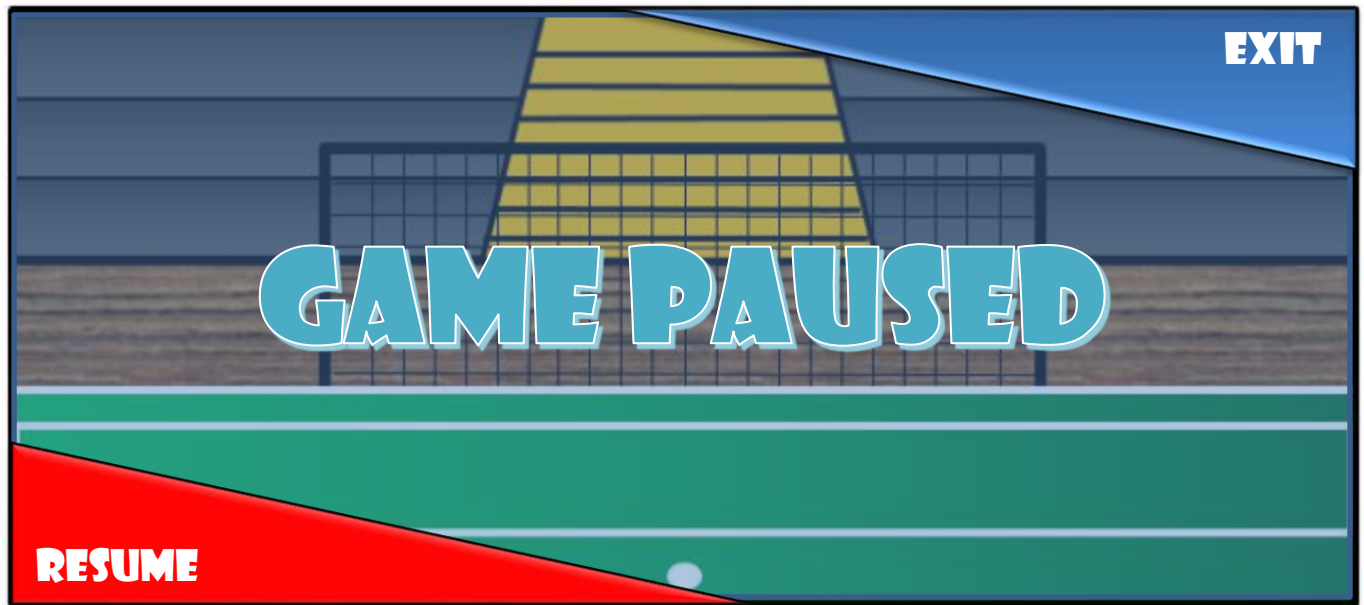
**LEFT MOUSE CLICK
IS USED FOR
NAVIGATION
AROUND THE
MENUS**

GENERAL CONTROLS

**CONTROLS FOR THE
SHOOTER**

**CONTROLS FOR THE
GOALKEEPER**

This page is design for my instruction menu, I have used a key to clearly show the user which controls are for which purpose, I have kept to the 'SHOW CARD GOTHIC' font as it suits the game a fits with the animated cartoon theme. As my target audience are between the ages of 5-14 years I have tried to make the instructions as simple as possible to reduce confusion, Billy said in the interview that the game needed an instruction menu to made a game easy to follow.

PAUSE MENU

This is a screen which gives the user the option to exit the game halfway through or to take a break, I like this design as I have used the transparency tool to give the faded effect on the pitch, all irrelevant detail has been removed such as the scoreboards and the players, they will be reloaded once the game has been resumed. This design has been made using Microsoft Word including the effect used on the text 'game paused' and the 3D effect upon the buttons in the top right and on the bottom left of the screen.

PLAYING STATE-PENALTY TAKER

This is the first shot that is about to be taken within the game, this is as both scoreboards have no red or green circles therefore no shots have been missed, saved or scored. The goalkeeper and football are sprites which have been taken from a sprite sheet on google images, the ball will move with very little difference between its new and original coordinates this will give the illusion that the ball is moving; the same method will be used to make the goalkeeper look like it is moving. The crosshair shows the shooter where their shot will go, however this only appears for the user while they are taking the shot. The power bar to the bottom right of the screen uses the basic components from Fifa 16 and its method used to apply power to the football.



This is what I have decided to use to control the amount of power that is applied to the ball, the purple triangle moves back and forth along the black line that it is placed on, once the space bar is hit the triangle stops upon a coloured bar. Whatever colour is it is placed into the bar underneath the line with the purple triangle on it, that colour represents a power, green gives the power which results in the ball hitting the exact target that the crosshairs are aiming at. If yellow is picked, then the power is increased and will be slightly off target so the crosshairs need to be aimed with that thought taken into account in order to have a chance of scoring the goal and if red is selected then the power is going to be too great to score and the shot will be too powerful to go in the net.

PLAYING STATE-GOALKEEPER

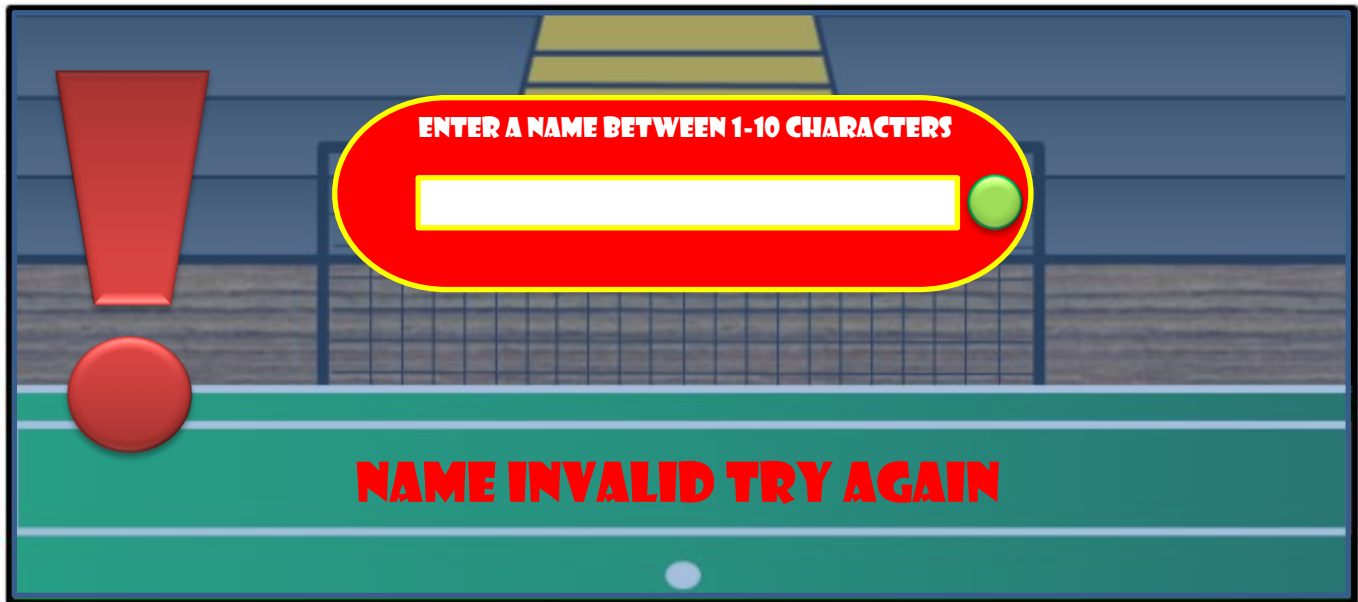


Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

This is what the game looks like when the player is playing as the goal keeper, all the detail involving shot power and crosshairs for the shot have been remove and the user will have to guess where the CPU intends on shooting in order to attempt to save the shot.

INITIALIZING THE game-enter gamer tag



This is the page that will be shown at the start of each game to allow Billy's requirement for customization to be fulfilled, in this example the name enter is not between 1-10 characters therefore an error message has occurred which causes the program to ask for another name attempt. Once the name has been entered the user hits the green button to the right of the bar where the name is entered, if the name is valid then the program continues into game.

SPRITES-FOOTBALL

This is the sprite I used as a football as it is a suitable ball which allows my game to have a cartoon animation theme which Billy required during the interview.





This shows the different sprites that can be used within the game to save the shot:

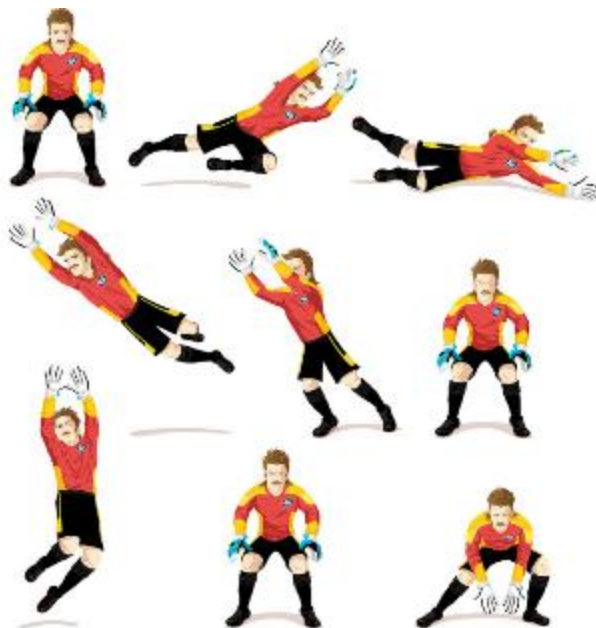
Left arrow causes goalkeeper one,

Right arrow causes goalkeeper two,

Space bar and right arrow simultaneously causes goalkeeper three,

Space bar and left arrow simultaneously causes goalkeeper four,

Space bar causes goalkeeper five.



This is the sprite sheet I used to take the goalkeeper sprites from to create a cartoon animation game.

Players

- **Customizable gamer tag**

- Initialize game
- Load up screen which asks for user to enter gamer tag
- If gamer tag enter is more than 10 characters or less than 1 character then load "Name invalid try again"
- If gamer tag is valid then proceed with loading the game

- **Turns**

Shooter to goalkeeper

- When the player is the shooter
- Check that the player has not had more than five shots
- If they can take the shot then, allow them too
- Else check the outcome of the game and produce the correct end screen
- Wait until shot has been taken
- If shot has been taken then, determine the outcome and update the score board
- Else check if shot has been taken
- Switch turn to goalkeeper

Goalkeeper to shooter

- When player is the goalkeeper
- Check that the CPU hasn't had more than five shots
- If the shot is allowed then proceed with the shot
- Else end game with suitable end page
- When shot has been taken determine outcome and update scoreboard
- Switch turn to shooter

- **Movement**

Shooter

- When player is shooter
- Move purple triangle back and forth along its transect
- When the space bar is hit stop pointer at a colour
- Load the selected colour into the box below the transect
- Apply the determined power to the ball given by the colour selected
- Set a timer of two seconds once the power is applied, once the timer hits zero the ball is kicked

- Before the timer hits zero the player moves the crosshairs by pressing the corresponding keys shown within the instruction menu
- Once the ball is hit set player to goalkeeper

Goalkeeper

- When player is goalkeeper
- Allow the goalkeeper to move with its corresponding keys when pressed
- The keeper has a speed half of the speed of the ball in order for the shooter to have a~ chance at scoring
- If the goal is saved reflect the ball of the area of the goalkeeper that was hit

Controls

- **Pausing the game**
 - When the game is being played
 - If 'P' is hit then load up the pause menu
- **Navigation**
 - Using the mouse curser and the mouse left click navigate through the menus within the game, such as the main menu, return buttons and during the pause menu

Scoring

- **Draw**
 - Game ends in a draw
 - Keep the current backing track going
 - Load the end page which shows a draw
 - Load the options to Restart the game
 - Load the option to Exit back to the main menu
- **Win**
 - Game ends in a win
 - Stop current background track
 - Load a sound track of cheering
 - Load the end page which shows that the game has been won
 - Load the option to restart the game
 - Load the option to exit back to the main menu

- **Loss**
 - Game ends in a loss
 - Stop current background track
 - Load sound track of booing
 - Load the end page that shows that the game was lost
 - Load the option to restart the game
 - Load the option to exit back to the main menu
- **During the game**
 - Game starts up
 - Set the score to zero before any shots have been taken
 - If the player takes a shot and it is scored then make the correct white circle green and increment the score by one
 - If the player doesn't score then turn the correct white circle red and move on to the next shot outcome
 - If the CPU scores/misses then update the score board accordingly and change the colour of the white circle to green if the goal is scored or to red if it is missed
 - Before each shot is taken make sure that the number of shots taken by each player does not exceed five shots each

Gamestate

- **Menu**
 - When menu is loading give option to play game, open instruction menu or to exit the game
 - If 'Play' is hit the initialize the game and change game state to playing
 - If 'Instructions' is selected then load up the instruction menu, and change game state to instruction.
 - If 'Exit' is selected then close the game down
- **Playing**
 - Initialize the game
 - Set scores to zero
 - Load up the sprites and the background
 - Start playing the backing track
 - If 'P' is hit load the pause menu
 - Change game state to paused and proceed with the correct outcome during this game state

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

-When game ends change game state to over

- **Over**

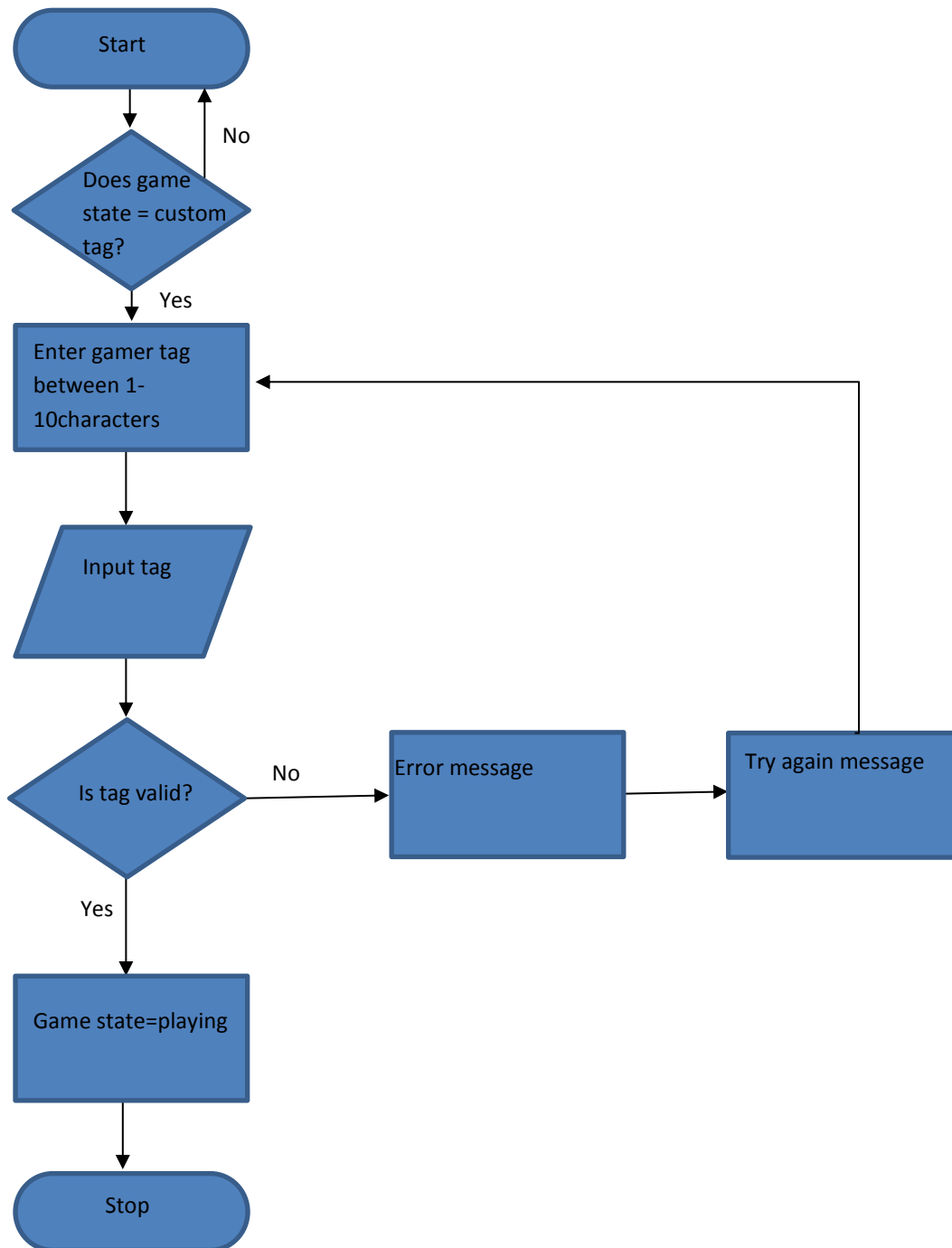
- Load the suitable end page based on the outcome of the score

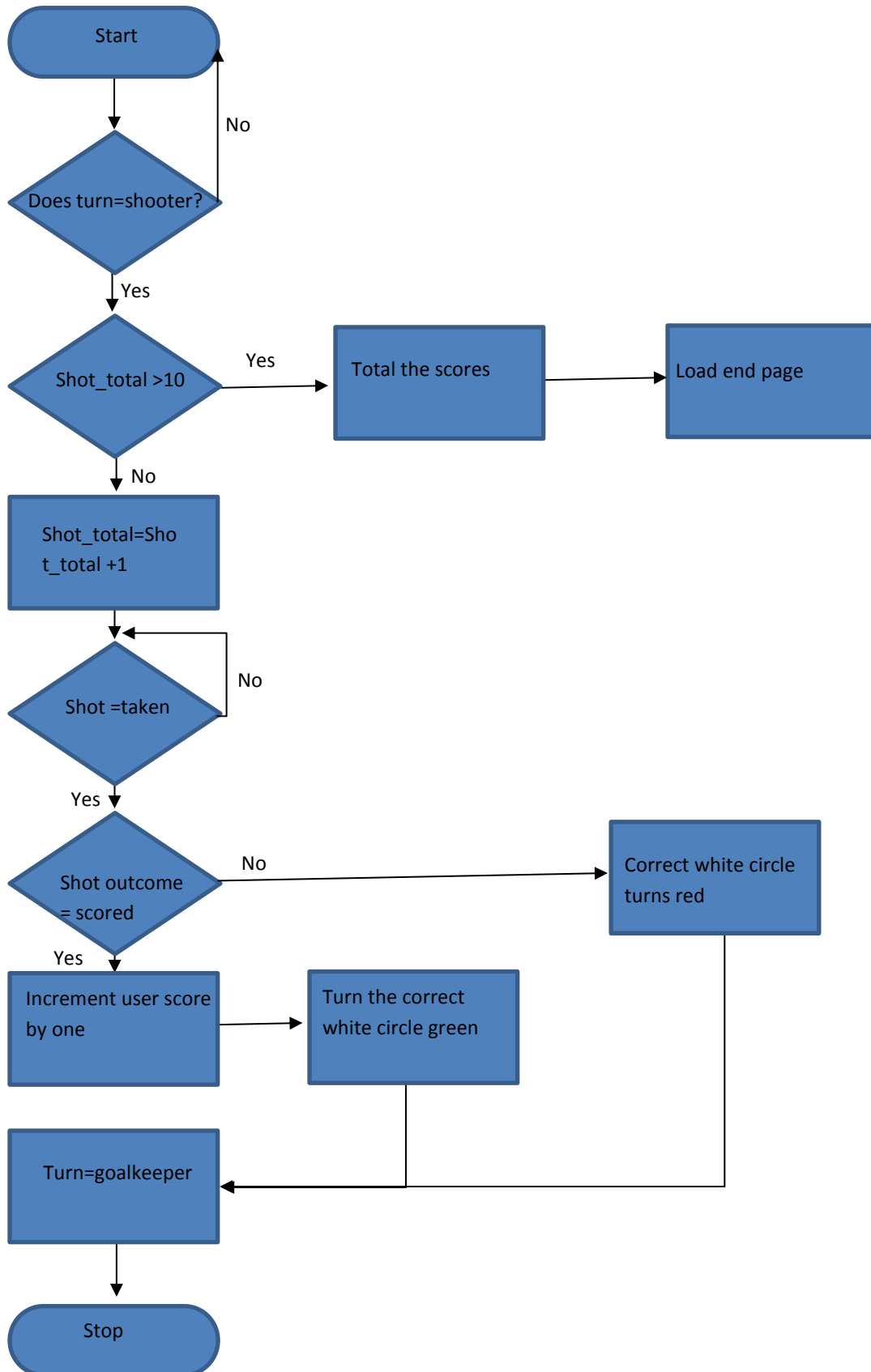
- Load the option to replay the game

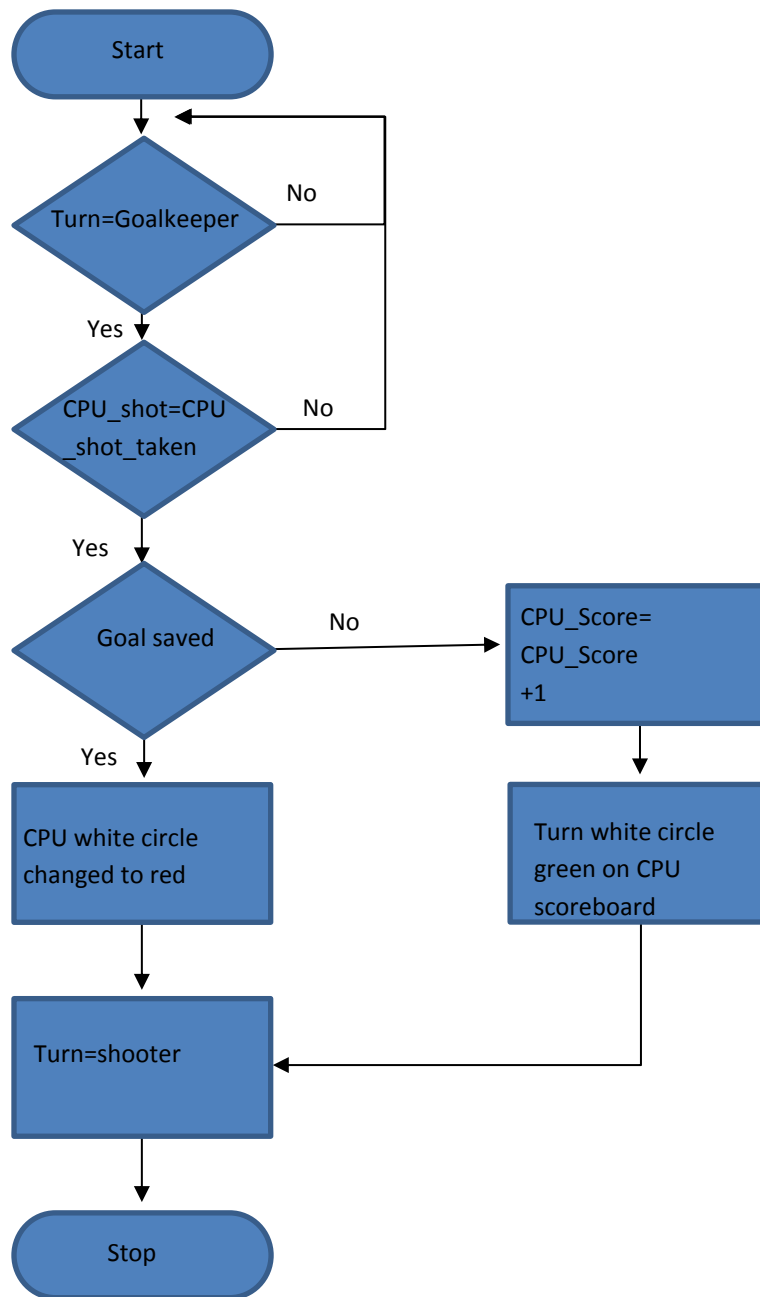
- If the game is restarted then set game state to playing

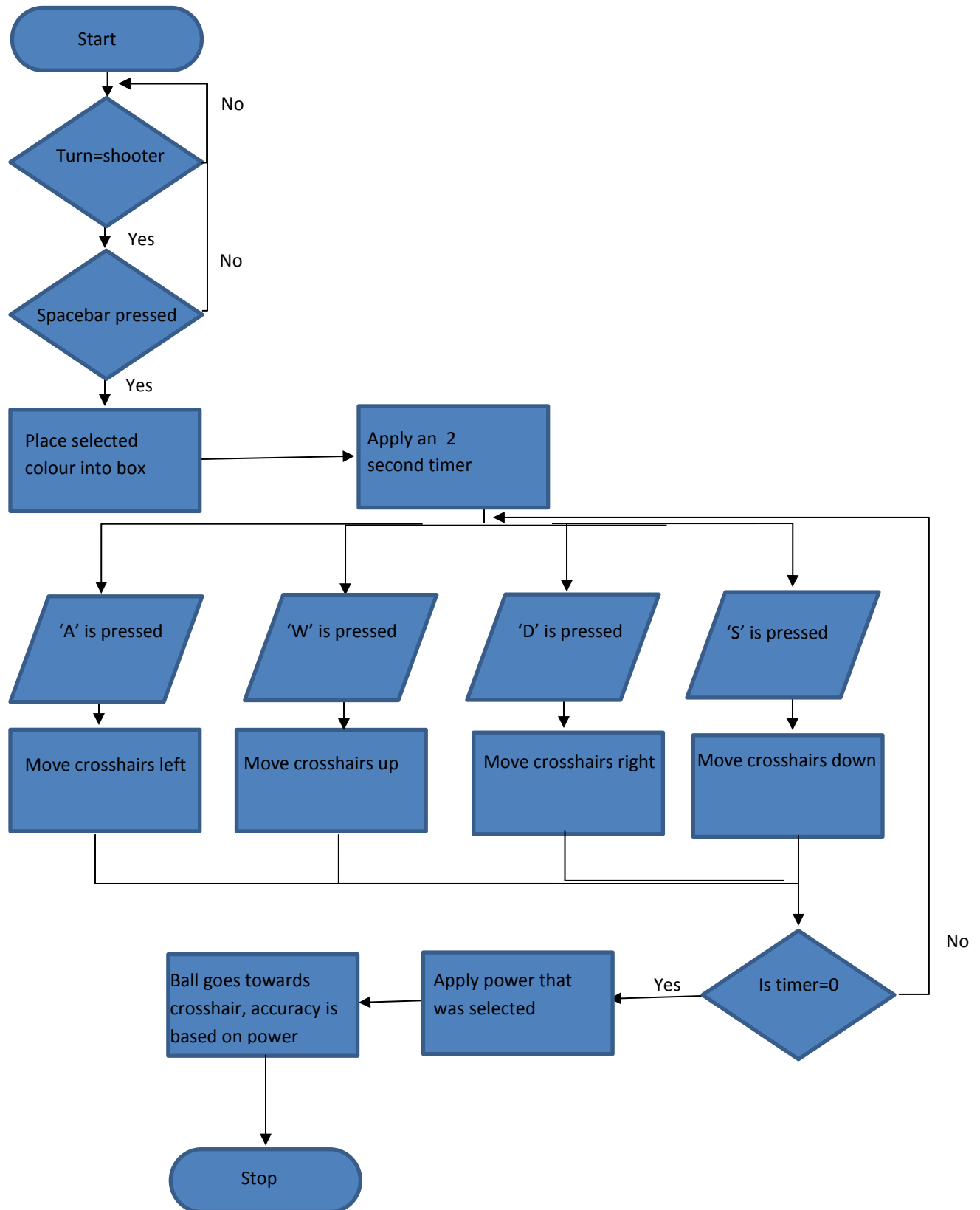
- Load option for the user to exit back to the main menu

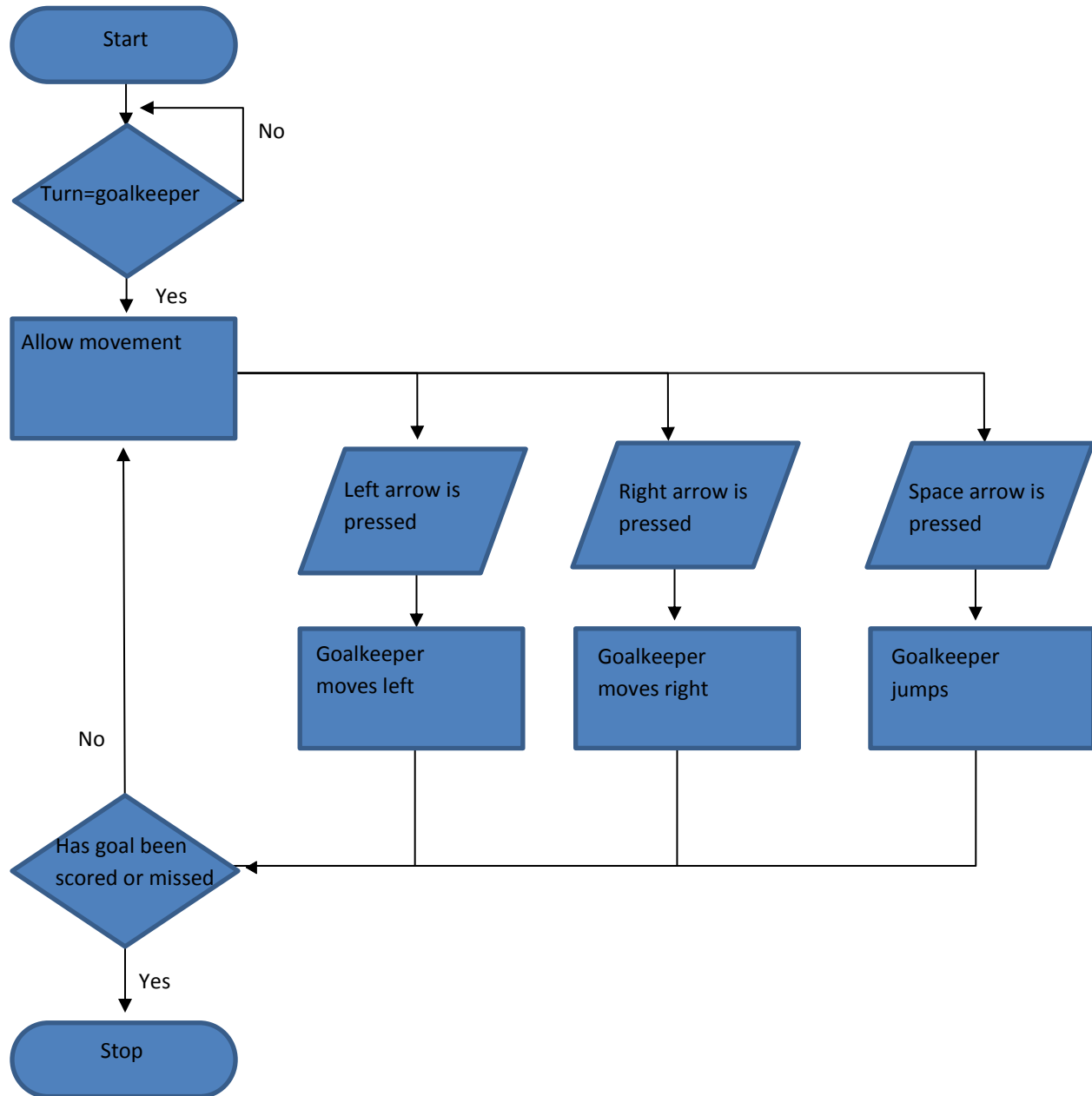
- If this is selected then change game state to menu

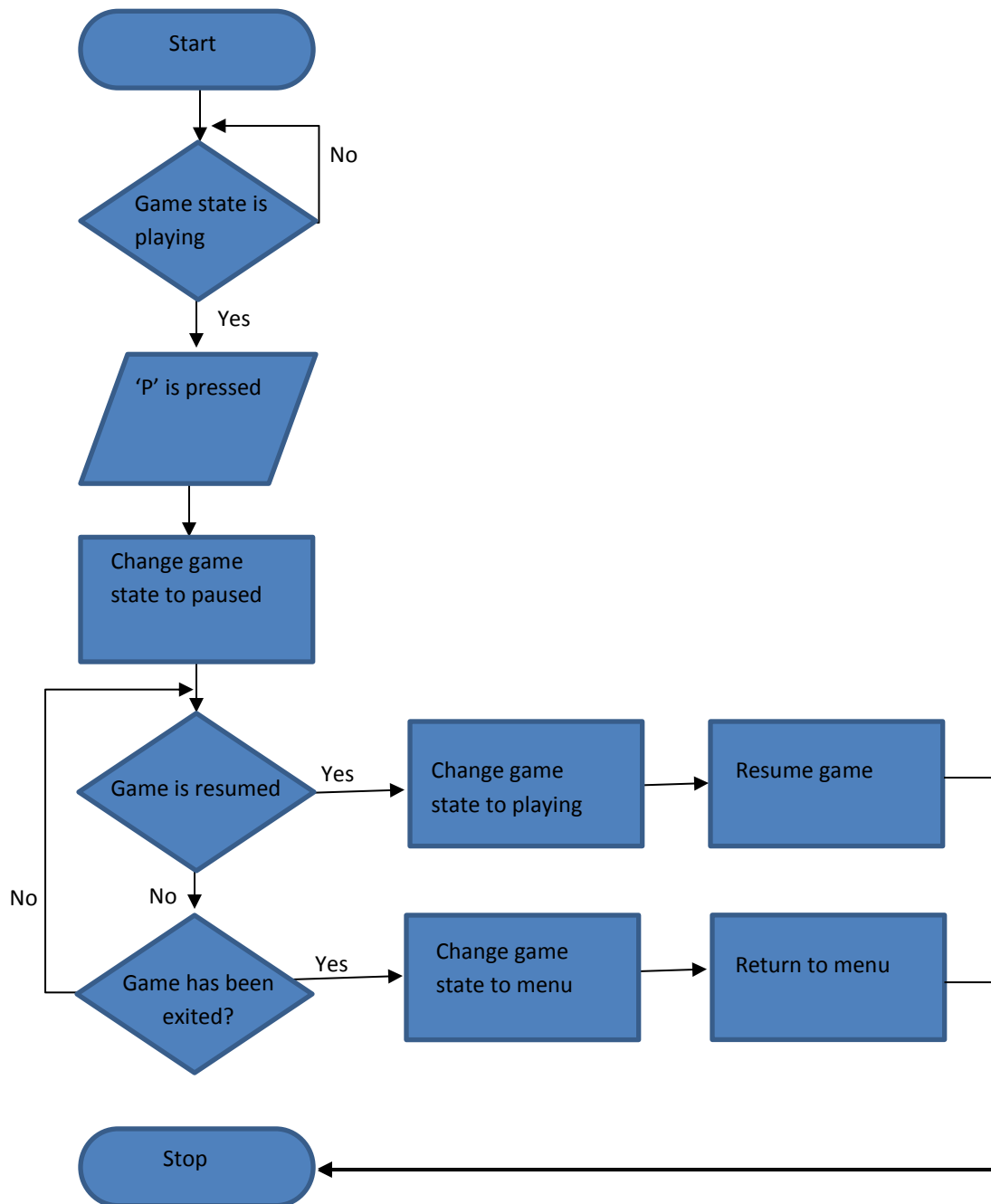
Players***Customizable gamer tag***

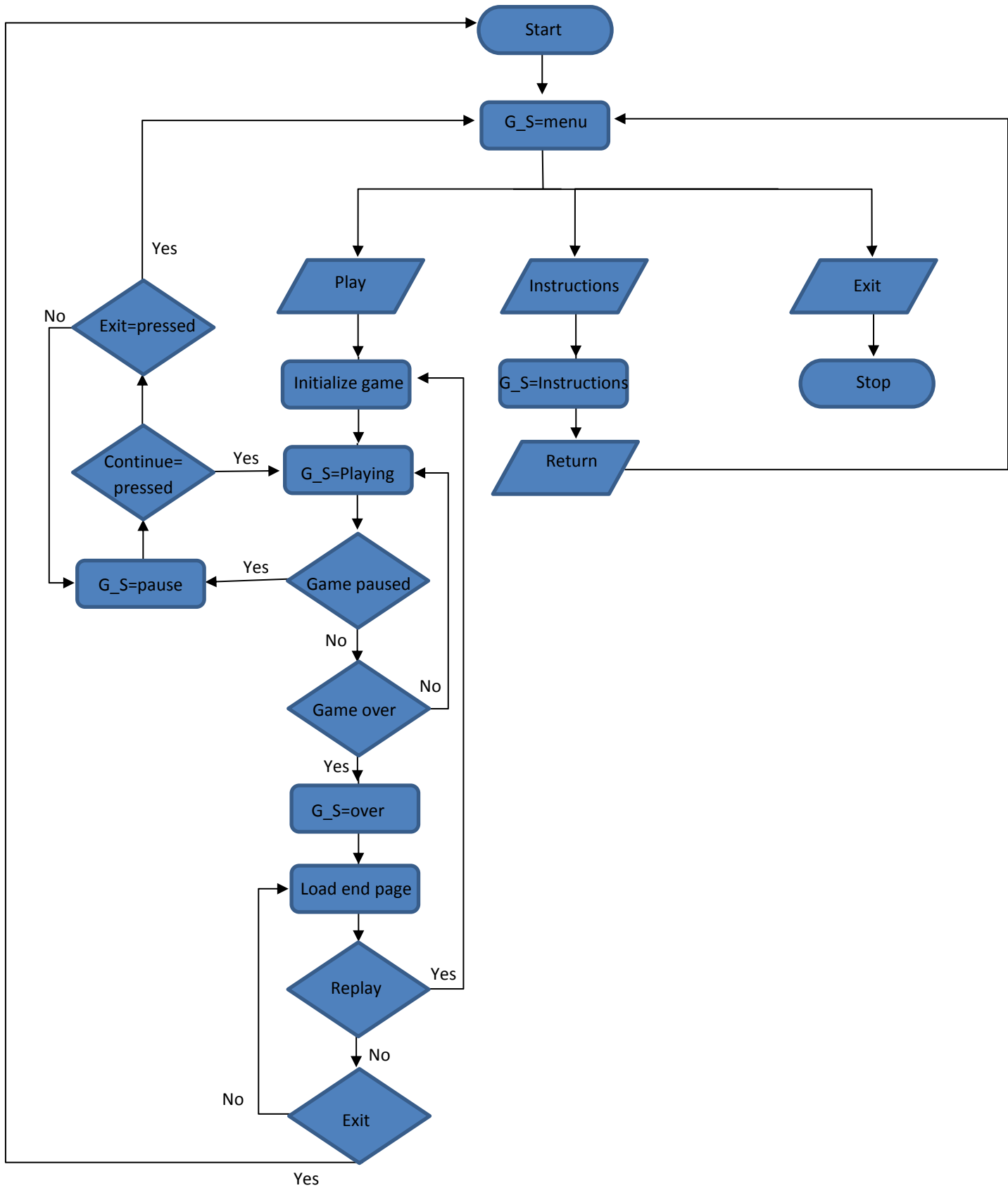
Turns-shooter to goalkeeper

Turns- Goalkeeper to shooter

Movement-shooter

Movement- goalkeeper

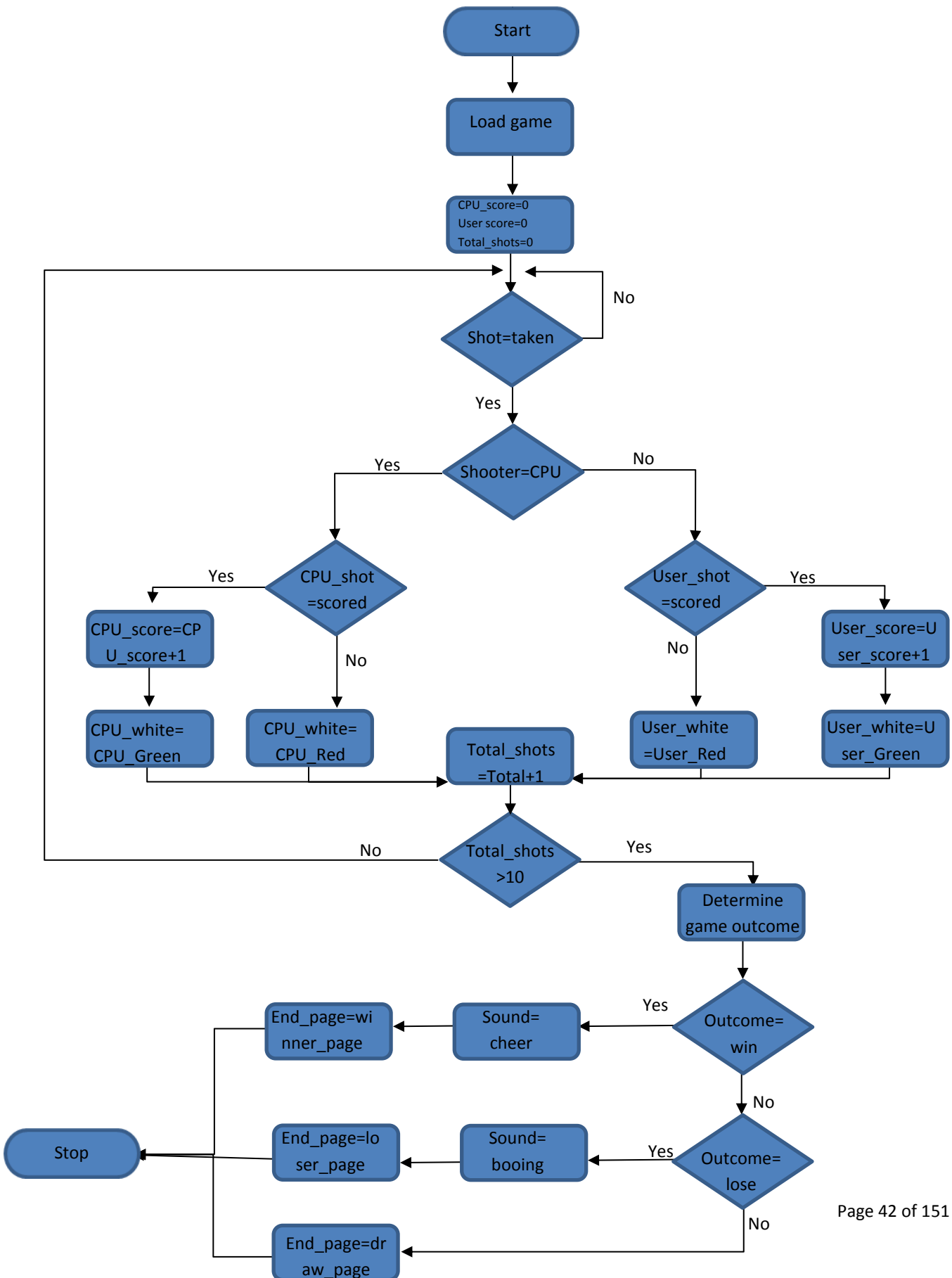
Controls***Pausing the game***

Game states

Candidate Name: XXXXXXXXXX

Candidate Number: XXXXXXXXXX

Scoring



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

TEST DATA FOR DEVELOPMENT**PLAYERS****Movement***Goalkeeper*

Test data	Type
'Left arrow'	Valid
'Right arrow'	Valid
'Spacebar'	Valid/Borderline
'W'	Invalid
'A'	Invalid
'S'	Invalid
'D'	Invalid
'P'	Valid

Shooter

Test data	Type
'W'	Valid
'A'	Valid
'S'	Valid
'D'	Valid
'Spacebar'	Valid/Borderline
'P'	Valid
'R'	Invalid
'3'	Invalid

Turns

Test data	Type
NA	NA

Customizable gamer tag

Test data	Type
Enter tag with 0 characters	Invalid
Enter tag with 7 characters	Valid
Enter tag with 11 characters	Invalid
Enter tag with 1 character	Invalid
Enter tag with 10 characters	Valid

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

CONTROLS

Pause

Test data	Type
'p'	Valid
'3'	Invalid
'{'	Invalid

Navigation

Test data	Type
'LMC'(left mouse click) on play button at the menu	Valid
'RMC'(right mouse click) on play button at the menu	Invalid
'LMC' on the screen where there are no buttons	Invalid

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

GAME STATES

Menu

Test data	Type
Select play(LMC)	Valid
Select instruction(LMC)	Valid
Select exit(LMC)	Valid
Click anywhere there is no button	Invalid
Select play(RMC)	Invalid
Select instruction(RMC)	Invalid
Select exit(RMC)	Invalid

Playing

Test data	Type
'P'	Valid
'O'	Invalid
'Left arrow'	Borderline
'Right arrow'	Borderline
'Spacebar'	Borderline
'W'	Borderline
'A'	Borderline
'S'	Borderline
'D'	Borderline
'P'	Borderline

Over

Test data	Type
Game won	Valid
Game lost	Valid
Game draw	Valid
Game exit	Valid
'Spacebar'	Invalid

Instruction

Test data	Type
Select return	Valid
'P'	Invalid
'A'	Invalid

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

SCORING

Drawing

Test data	Type
Load game	Valid
End game	Valid
Score goal	Valid
Miss goal	Valid
'p'	Invalid
'Spacebar'	Invalid

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

SOUND EFFECTS

Test data	Type
Load game	Valid
Game ends as draw	Valid
Game ends as win	Valid
Game ends as loss	Valid
'p'	Invalid
'Spacebar'	Invalid

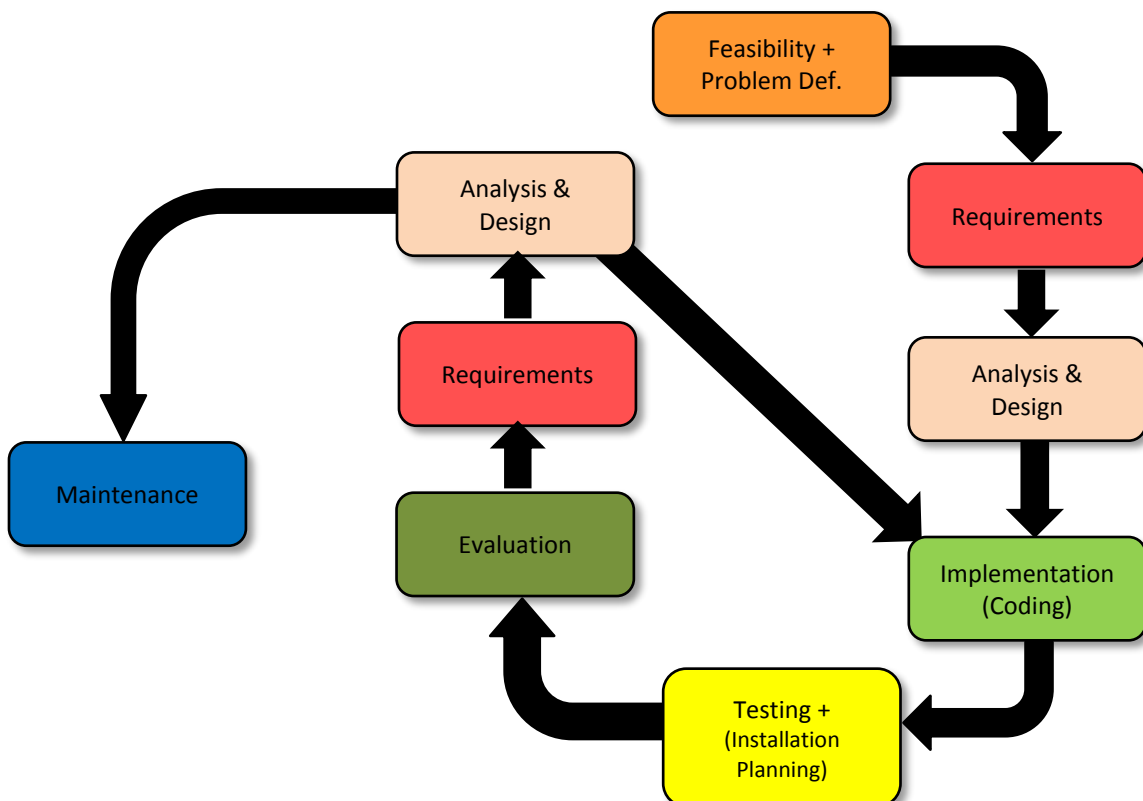
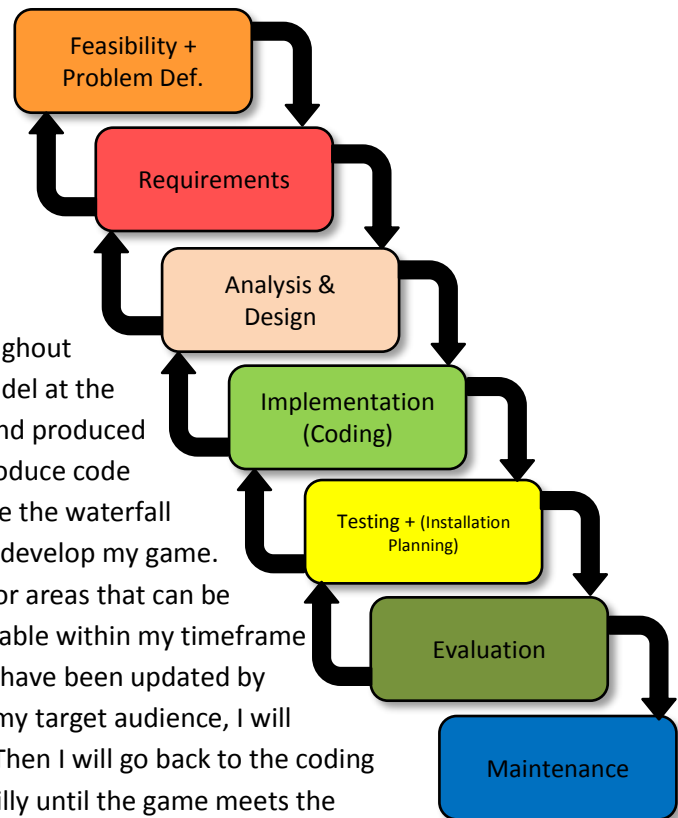
SDLC PROCESS

Waterfall development

I will be using the waterfall methodology for the first areas of my project including feasibility and problem definition, requirements and analysis and design. This shows how I will use the requirements to produce and analysis and design the fit to my target audience.

Agile methodology

This is a mixture of methodologies which I will be using throughout my project, I used the traditional method of the waterfall model at the start where I proposed a problem found the requirements and produced an analysis and a design. Later on within the project I will produce code which I will then be tested and evaluated, this is where I leave the waterfall methodology and start to perform iterations to improve and develop my game. After the first evaluation I check the requirements to check for areas that can be improved or removed if that certain requirement is unachievable within my timeframe and or with my programming ability. Once the requirements have been updated by me and Billy Evans, which is the person that I have acting as my target audience, I will update the analysis and design to fit the new requirements. Then I will go back to the coding stage where I repeat the stages taking improvements from Billy until the game meets the requirements or the time frame runs out, once this happens I will focus on game maintenance.



TEST DATA FOR BETA TESTING

Testing players and controls within the game

Test number	What is being tested?	Input/action	Expected outcome
A	Movement	The movement keys are pressed	The goalkeeper or shoot reacts with the corresponding movement
1	Goalkeepers movements	VALID: Right arrow pressed twice, while player is goalkeeper INVALID: 'D' is pressed twice, while player is goalkeeper	VALID: Goalkeeper moves two places to the right INVALID: The goalkeeper won't move
2	Goalkeepers movements	VALID: Left arrow pressed twice, while player is goalkeeper INVALID: 'A' is pressed twice, while player is goalkeeper	VALID: Goalkeeper moves two places to the left INVALID: The goalkeeper won't move
3	Goalkeepers movements	VALID: Spacebar is pressed once, while player is goalkeeper INVALID: 'W' is pressed twice, while player is goalkeeper	VALID: Goalkeeper jumps up to reach the ball for a chosen time INVALID: The goalkeeper won't move
4	Applying power to the ball before taking a shot	VALID: While player is the shooter press spacebar to stop the purple slide from moving and to select the power INVALID: While player is the shooter and selecting press ENTER	VALID: The purple pointer stops on a colour that colour is then placed in the box below the slide, that colour later determines the power INVALID: No affect from pressing enter

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

5	Moving the crosshairs for the shot	<p>VALID: While the cross hairs are on the screen, A is pressed once</p> <p>INVALID: While the cross hairs are on the screen, 'left arrow' is pressed once</p>	<p>VALID: The crosshairs move left one place</p> <p>INVALID: The crosshairs won't move</p>
6	Moving the crosshairs for the shot	<p>VALID: While the cross hairs are on the screen, W is pressed once</p> <p>INVALID: While the cross hairs are on the screen, 'up arrow' is pressed once</p>	<p>VALID: The crosshairs move upwards one place</p> <p>INVALID: The crosshairs won't move</p>
7	Moving the crosshairs for the shot	<p>VALID: While the cross hairs are on the screen, S is pressed once</p> <p>INVALID: While the cross hairs are on the screen, 'down arrow' is pressed once</p>	<p>VALID: The crosshairs move downwards one place</p> <p>INVALID: The crosshairs won't move</p>
8	Moving the crosshairs for the shot	<p>VALID: While the cross hairs are on the screen, D is pressed once</p> <p>INVALID: While the cross hairs are on the screen, 'right arrow' is pressed once</p>	<p>VALID: The crosshairs move right one place</p> <p>INVALID: The crosshairs won't move</p>
B	Turns	The game will run through from start to finish	The turns will switch between goalkeeper and shooter after every go
9	Once the goalkeeper finishes there go they will then be switched to a shooter	<p>VALID: The goal has been saved/missed</p> <p>BORDERLINE: press 'P'</p>	<p>VALID: The goalkeeper then becomes the shooter</p> <p>BORDERLINE: The game will pause</p>

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

10	Once the shooter has taken there shot and the score has been updated the player will then become the goalkeeper	VALID: The shot has been taken and the score has been updated BORDERLINE: press 'P'	VALID: The shooter become goalkeeper BORDERLINE: The game will pause
C	The player will be able to customize their gamer tag	The user input the gamer tag	If the gamer tag is valid the game will load with the new gamer tag
11	Whether the game will detect an incorrect gamer tag	VALID: Enter a gamer tag with more than ten characters	VALID: There will be an error message and the user must enter a new gamer tag
12	Whether the game will continue when a correct gamer tag is entered	VALID: Enter a gamer tag which between 1-10 characters BORDERLINE: Enter a tag with one character BORDERLINE: Enter a tag with ten characters	VALID: If correct the user will proceed to the game BORDERLINE: The game will proceed BORDERLINE: The game will proceed
D	Whether the controls corresponding with the desired outcome	The keys with a function will be pressed	The task that the corresponding key does will be carried out
13	The pause feature	VALID: During the game the key 'P' will be pressed INVALID: During the game press the 'O' key	VALID: The game will load the paused game state INVALID: The game will continue playing
14	The navigation around menus	VALID: The left mouse click will be pressed on all the buttons within the game (includes all menus, such as the pause menu, main menu, return and rematch buttons)	VALID: The button that is selection will proceed to the correct screen

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

		INVALID: The left mouse click is used were there are no buttons	INVALID: There would be any change within in the game as a result of the click
--	--	--	---

Testing game states

Test number	What is being tested?	Input/ actions taken	Expected outcome
A	Each game state changes at the right time	Actions which cause the game state to change	Game state will change to the correct form
15	Transition from game state of menu to the playing game state	VALID: The 'play' button is clicked at the main menu INVALID: At the main menu press 'ENTER'	VALID: The game state will change to playing and the game will initiate INVALID: No change should occur to the game state
16	Transition from game state of menu to the instruction game state	VALID: The 'instruction' button is clicked at the main menu	VALID: The game state will change to instruction
17	Closing the game down	VALID: The 'exit' button is selected at the main menu	VALID: The game will close
18	The transition to the game state paused	VALID: 'P' is pressed while the game is playing	VALID: The game will pause
19	Transition to 'over' game state	VALID: Complete the game INVALID: Press 'P' and then click 'exit'	VALID: The game state will change from playing to over INVALID: The game returns to the menu game state

Testing the scoring mechanics

Test number	What is being tested?	Input/ action taken	Expected outcome
A	Testing the correct end page is displayed	The game will end in different scenarios	The corresponding end screen will be shown
20	The end screen shows a draw when a draw occurs	VALID: The game will end in a draw INVALID: Press 'P' and then click 'exit'	VALID: The end load screen will show the draw display INVALID: The game returns to the menu game state
21	The end screen shows a winning page if the game is won	VALID: The game will end in a win INVALID: Press 'P' and then click 'exit'	VALID: The end page displays a winning page INVALID: The game returns to the menu game state
22	The end screen shows a losing page if the game is lost	VALID: The game will be lost INVALID: Press 'P' and then click 'exit'	VALID: The end page will be the losing page INVALID: The game returns to the menu game state
B	The initial score board	Loading the game	Correct values within the scoreboard
23	The score board is set to null when the game starts	VALID: Load the game BORERLINE: Load the instruction menu	VALID: The scoring boards will be empty BORFERLINE: The instruction menu we be displayed
24	The maximum number of shots	VALID: Runs the game through to the end	VALID: The game will end once five scores each have been taken
C	Updating the score board	A shot will be taken	The score board will update accordingly
25	The score board increases when a shot is taken	VALID: A goal will be scored	VALID: The correct score board will turn the corresponding

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

			white circle green and increment the score by one
26	The score board remains the same while a shot is missed	VALID: A goal is saved or missed	VALID: The correct score board will turn the corresponding white circle red and the score is kept the same

Testing sound effect

Test number	What is being tested?	Input/action taken	Expected outcome
A	The correct sound track plays at the correct time	The game will enter different stages	The corresponding sound effect/sound track plays with the action taken
27	The backing track within the game	VALID: Run the game to the end INVALID: Press 'P' and then click 'exit'	VALID: While game state is playing there will be a backing track for the game INVALID: The main menu is loaded and all soundtracks/effects are stopped
28	There are suitable sound affects when the game ends	VALID: The game ends as a win INVALID: Press 'P' and then click 'exit'	VALID: There is cheering on the winning page INVALID: The main menu is loaded and all soundtracks/effects are stopped
29	There are suitable sound affects when the game ends	VALID: The game ends as a draw INVALID: Press 'P' and then click 'exit'	VALID: The backing track will continue to play over the draw end page INVALID: The main menu is loaded and all soundtracks/effects are stopped

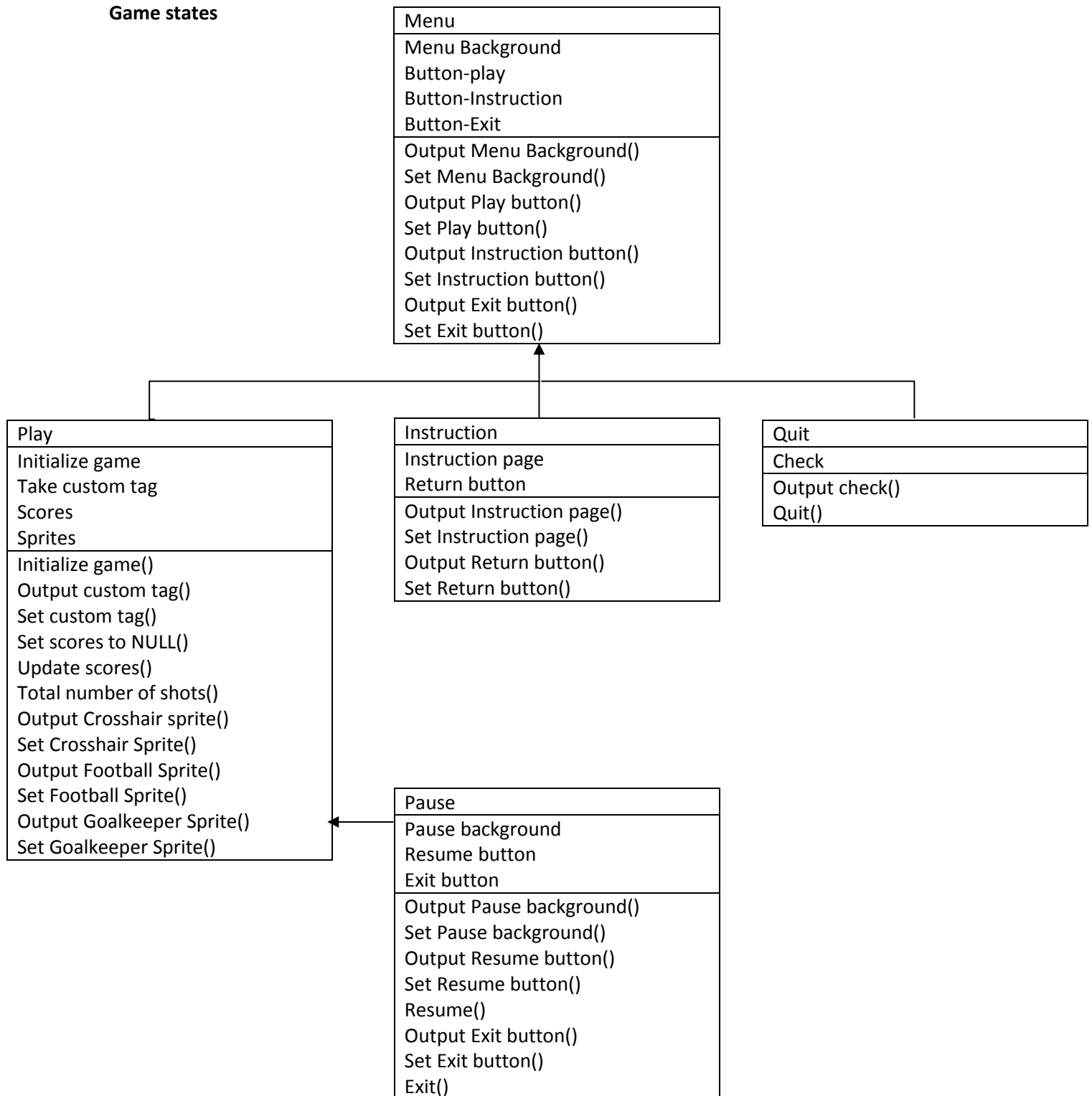
Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

30	There are suitable sound affects when the game ends	VALID: The game ends in a loss INVALID: Press 'P' and then click 'exit'	VALID: There will be booing on the losing page INVALID: The main menu is loaded and all soundtracks/effects are stopped
----	---	--	--

CLASS DIAGRAM

Game states



KEY VARIABLES AND DATA STRUCTURES

NOTE: ALL VARIABLES ARE LOCAL USLESS STATED OTHERWISE

Method	Stored label	Data type	Actual meaning	Justification
Game state	Custom_tag	NA	Game is at the stage where a gamer tag is enter	The game state will produce a variable called 'Tag' which holds the custom gamer tag
Variable	Tag	Char	The gamer tag which the user enters	This variable holds gamer tag which will be used as the played identification
Game state	Playing	NA	The program is at the stage where the game is taking place	This holds the code that will determine the outcome of the game

Customizable gamer tag

The algorithm provides a base for the customizable gamer tag which provides the code for the game state of Custom_tag, this game state will be called at the end of the game initialization, and it also changes the game state to playing at the end of the code. This code is called once each time at the start of the game as it has no further impact upon the game.

Turns

Method	Stored label	Data type	Actual meaning	Justification
Procedure	Turn	NA	Turn acts as a way of asking who is shooting or being the goalkeeper	This code determines and switches the players turn between goalkeeper and shooter
Variable	Shooter		This is what is used identify the shooters turn	When turn is shooter then shooter movement and scoring code is used
Variable	Goalkeeper		This is used to identify the goalkeepers turn	The allows the code for the goalkeeper to be used
Variable (global)	Total_shot	Integer	This is the total number of shots taken so far	This is used to prevent the game from exceeding a maximum of five shots each
Variable	Shot		This is a variable used while turn = shooter	This will check if the shot has been taken or not
Variable	CPU_shot		This is a variable used while turn =goalkeeper	This will check if the CPU has taken a shot the yet

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Variable	Taken		This states that the shot has been taken	Once shot has been taken the variable shot will be set to taken which allows the program to proceed
Variable	CPU_taken	Boolean	This states that the CPU's shot has been taken	Once shot has been taken the variable shot will be set to taken which allows the program to proceed
Variable	Shot_outcome	Boolean	This reads the shots outcome	This is used to say what happened during the shot
Variable	Scored		This states that the user scored	This then opens the code within the scoring procedure which allows the score to get updated
variable	CPU_scored		This states that the CPU score	This then opens the code within the scoring procedure which allows the score to get updated
Variable	Missed		This states that the user missed	This then opens the code within the scoring procedure which allows the score to get updated
Variable	CPU_missed		This states that the CPU missed	This then opens the code within the scoring procedure which allows the score to get updated

This algorithm is used as a template for the code which will be used to switch turns between being the shooter and the goalkeeper within the game.

Method	Stored label	Data type	Actual meaning	Justification
Variable	shooter		The player is currently shooting	Allows the game to run the correct code for the movement of the crosshairs, and power selection
Variable	Selected power	Integer	The power selected from the power bar	This power then affects how accurate the shot is
Variable	Timer	Integer	A timer	Allows a countdown after power has been applied to the ball
Field sprite	Crosshair	NA	The crosshairs	This shows where the ball will be aimed for the shot
Variable	Crosshair_direction	Char	The direction of the crosshair	This will be used to detect which keys are pressed and the key will be used to move the crosshair
Field sprite	Goalkeeper	NA	The player is currently the goalkeeper	Allows the game to run the correct code for the movement of the goalkeeper

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Variable	Shot_taken	Boolean	Has the shot been taken	Checks if the shot has been taken so the game can proceed
----------	------------	---------	-------------------------	---

Movement

The movement algorithms are used to detect inputs which allow the goalkeeper or crosshair to move, the output of the code will be the shot parameter that will be changed to taken which opens up the code for the scoring.

Pausing

Method	Stored label	Data type	Actual meaning	Justification
Game state	Playing	NA	The program is at the stage where the game is taking place	This holds the code that will determine the outcome of the game
Game state	Paused	NA	The program is at the stage where the game is paused	This game state holds the code for the game while it is paused, this code contains a resume and exit button
Procedure	Resumed	NA	The game is resumed	This changes the game state back to playing
Procedure	exit	NA	The game is exited	This changes the game state back to menu
Game state	Menu	NA	The program is at the stage where the main menu is loaded	This game state holds the code for the main menu where the game can be played, the instructions can be loaded or the game can be quit

The pause algorithm is a basic but useful piece of code as it provides an exit all the way throughout the game while it is playing and is a feature that is in every successful game, such as the game I researched during my analysis which was Fifa 16. However, Fifa has a more complex pause menu where you are able to change the games setting, view the games statistics and change the controls ect...

Scoring

Method	Stored label	Data type	Actual meaning	Justification
Variable	CPU_score	Integer	Score for CPU	This holds the score for the CPU
Variable	User_score	Integer	Score for user	This holds the score for the user

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Variable	Total_shots	Integer	Total shots taken	This totals the number of shots taken so that each player only has five shots each
Sound file	Backing_track	NA	Sound track	The backing track will play over the game to give it more of an inviting atmosphere
Variable	Shot	Boolean	This is a variable used while turn = shooter	This will check if the shot has been taken or not
Variable	Shot_taken		Has the shot been taken	Checks if the shot has been taken so the game can proceed
Procedure	Turn	NA	Turn acts as a way of asking who is shooting or being the goalkeeper	This code determines and switches the players turn between goalkeeper and shooter
Variable	CPU_shot		The shot of the CPU	Asks if the shot has been taken or not
Variable	User_shot		The shot of the user	Asks if the shot has been taken or not
Field Sprite	White_circle	NA	White circle on the score board	Represents a shot that hasn't yet been taken
Field Sprite	Green_circle	NA	Green circle on the score board	Represents a shot that has been scored
Field Sprite	Red_circle	NA	Red circle on the score board	Represents a shot that hasn't been scored
Procedure	Win	NA	Game is won	The end page is set the winning display within this procedure
Procedure	Lose	NA	Game is lost	The end page is set the losing display within this procedure
Procedure	Draw	NA	Game is draw	The end page is set the draw display within this procedure
Sound file	cheering	NA	Sound effect of booing	The cheering sound effect will be played over the winning end page to provide a sense of achievement for the target audience
Sound file	Booing	NA	Sound effect of cheering	The booing will be played over the losing end page to make the game more interesting

The scoring algorithm will be one that checks the outcome of the score and based on whether the shot was scored or not and will update the scores, total scores and scoreboard with the correct colour accordingly.

PROPOSAL SIGN OFF

Scoring Board Summary

What is successful	Improvements to be made
<ul style="list-style-type: none"> Aesthetically pleasing Scoreboard changing colour based on the outcome of the shot Custom tag is next to the scoreboard 	NA

End Page Screen Proposal Summary

What is successful	Improvements to be made
<i>Winning page</i> <ul style="list-style-type: none"> The bright colours within the page The idea of cheering when the page loads The buttons to replay or to exit 	<ul style="list-style-type: none"> Display the games score within the end page
<i>Losing page</i> <ul style="list-style-type: none"> The booing when the game is lost The buttons to replay or to exit The page isn't as colourful once the game is lost 	<ul style="list-style-type: none"> Display the games score within the end page
<i>Drawing page</i> <ul style="list-style-type: none"> The buttons to replay or to exit 	<ul style="list-style-type: none"> Not as much going on within the page compared to the other end pages Display the games score within the end page

Main Menu Screen Proposal Summary

What is successful	Improvements to be made
<ul style="list-style-type: none"> The design where the buttons are hanging The layout as you can still see the stadium while at the main menu 	<ul style="list-style-type: none"> The 'instruction' needs to be changed as it looks to cramped

Instructions Screen Proposal Summary

What is successful	Improvements to be made
<ul style="list-style-type: none"> The instructions are clear and easy to follow 	<ul style="list-style-type: none"> Make the page look neater by using a cartoon key board and mouse

Initializing Game Screen Proposal Summary

What is successful	Improvements to be made
--------------------	-------------------------

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

<ul style="list-style-type: none">• The design is good and provides clear instructions	<ul style="list-style-type: none">• Use brighter colours to make the game look more exciting
--	--

Pause Screen Proposal Summary

What is successful	Improvements to be made
<ul style="list-style-type: none">• The layout of the pause menu• The stadium is still visible and stays with the football theme	<ul style="list-style-type: none">• Show the score at the pause menu to make the game look more professional

Shooters perspective Screen Proposal Summary

What is successful	Improvements to be made
<ul style="list-style-type: none">• The power bar is easy to use and under stand• The crosshairs are clear• The goalkeeper is proportional to the goal	<ul style="list-style-type: none">• Make the power selector bar

Shooters perspective Screen Proposal Summary

What is successful	Improvements to be made
<ul style="list-style-type: none">• All irrelevant features are removed such as the crosshairs and the power bar	<ul style="list-style-type: none">• Change to goalkeepers kit so that it isn't the same as the CPU's

Signed

C. DEVELOPING THE CODED SOLUTION ("THE DEVELOPMENT STORY")**04/01/2017 ITERATIVE DEVELOPMENT**

The software development of my game follows an iterative process where after each module has been produced I will provide a prototype in which I will hold an interview with my client Billy Evans where he will see the development and will be able to make comments and improvements for the next iteration where I will produce a new prototype. I will develop modules based on the systems diagram produced within the design as this keeps the development logical and easier to follow which makes good practice for software development. In reality this process will repeat until the client runs out of money for the software development, the software has been completed or it has reached a publishable standard. However, I will only perform these iterations until I run out of time or if the game has been completed.

CODE ANALYSIS

In this area of development I am analysing games that have been made in monkey x, this will help me achieve a greater understanding of the programming language so I am able to produce a successful penalty shooter game for those between the ages of five and fourteen.

04/01/2017 ARCADE SHOOTER

```
'Libraries and globals
Import mojo
Global Game:Game_app
```

Importing an library

Declaring a global variable

```
'Main program starts here
Function Main ()
    Game = New Game_app
End
```

This is the actually program that is run for the game

```
'Main game class
Class Game_app Extends App
    Field menu:Image
    Field player:Rocket
    Field enemy_collection:List<Alien>
    Field missile_collection:List<Laser>

    Global GameState:String = "MENU"
    Global shoot:Sound
```

This is shows the attributes being declared within the class

Variable menu holds a picture

Creates a variable for a rocket

Creates an alien variable

Creates a laser variable

Holds a game state

```
Method OnCreate ()
    'All the game initialisation goes in here:
    SetUpdateRate 60
    menu = LoadImage ("menu.png")
    'Object initialisation goes in here:
    player = New Rocket
    enemy_collection = New List<Alien>
    missile_collection = New List<Laser>

    'Spawn first enemy
    enemy_collection.AddLast(New Alien(200,150))
    enemy_collection.AddLast(New Alien(400,200))

    shoot = LoadSound("shoot.wav")
    PlayMusic("music.ogg",1)
End
```

Initial object declarations

Set the frame rate to 60fps

Menu is loaded with the picture

Player is now the rocket

New list for alien

New list for Missiles

Spawns in two types of alien

Sound effects for shooting

Backing track


```

Method OnUpdate ()
'All the game logic goes in here:
Select GameState
Case "MENU"
    If KeyHit (KEY_SPACE) Then GameState="PLAYING"
Case "PLAYING"
    'Handle inputs
    If KeyHit (KEY_ESCAPE) Then GameState="MENU"
    If KeyDown(KEY_LEFT) Then player.Move(-10)
    If KeyDown(KEY_RIGHT) Then player.Move(10)
    If KeyHit(KEY_F) Then SetMusicVolume(0.1)
    If KeyHit(KEY_G) Then SetMusicVolume(1)
    If KeyHit(KEY_M) Then
        If MusicState = 1 Then
            PauseMusic
        Else
            ResumeMusic
        End
    End
End

```

Movement and logic

Used a select case with game states to give functions to keys

Games starts at the menu state

-10, is related to the movement speeds

```

If KeyHit(KEY_SPACE) Then
    missile_collection.AddLast(New Laser(player.x,player.y))
    PlaySound(shoot)
End
'Handle missile movement
For Local missile:=EachIn missile_collection
    missile.Move(5)
    If missile.y<10 Then missile_collection.Remove missile
Next
'Collision detection
For Local missile:=EachIn missile_collection
    For Local enemy:=EachIn enemy_collection
        If Intersects(missile.x,missile.y,10,11,enemy.x,enemy.y,100,50) Then
            enemy_collection.Remove enemy
            missile_collection.Remove missile
        End
    Next
Next
End

```

If the case is playing and space bar is hit then a missile is created

Missile movement

Missile detection, missiles dimensions are (10, 21), alien dimensions are (100, 50), and if they collide the alien isn't redrawn

```

Method OnRender ()
'All the graphics drawing goes in here:
Select GameState
Case "MENU"
    DrawImage menu, 0,0
Case "PLAYING"
    Cls 0, 0, 0
    DrawImage player.sprite, player.x, player.y
    For Local enemy:=EachIn enemy_collection
        DrawImage enemy.sprite, enemy.x, enemy.y
    Next
    For Local missile:=EachIn missile_collection
        DrawImage missile.sprite, missile.x, missile.y
    Next
End
End

```

This is where screen drawing occurs

Draw picture of menu when game state is menu

If game is playing the draw colours shown

Draws new position to the screen

Loop to draw every alien to the screen

Loop to draw every missile on the list

```
'Class definitions for game objects:
```

```
Class Rocket
```

```
Field sprite:Image = loadImage ("player.png")
```

```
Field x:Float = 100
```

```
Field y:Float = 420
```

```
Method Move(x distance:Int)
```

```
  x+=x distance
```

```
  If x<0 Then x=0
```

```
  If x>590 Then x=590
```

```
End
```

```
End
```

```
Class Alien
```

```
Field sprite:Image = loadImage ("alien.png")
```

```
Field x:Float
```

```
Field y:Float
```

```
Method New(x_spawn:Int, y_spawn:Int)
```

```
  x = x_spawn
```

```
  y = y_spawn
```

```
End
```

```
End
```

```
Class Laser
```

```
Field sprite:Image = loadImage ("missile.png")
```

```
Field x:Float
```

```
Field y:Float
```

```
Method New(x_spawn:Int, y_spawn:Int)
```

```
  x = x_spawn+20
```

```
  y = y_spawn
```

```
End
```

```
Method Move(y_distance:Int)
```

```
  y-=y_distance
```

```
End
```

```
End
```

Class for rocket

Rocket image

Starting position

Horizontal movement

Class for alien

Alien image

Spawning aliens

Class for laser

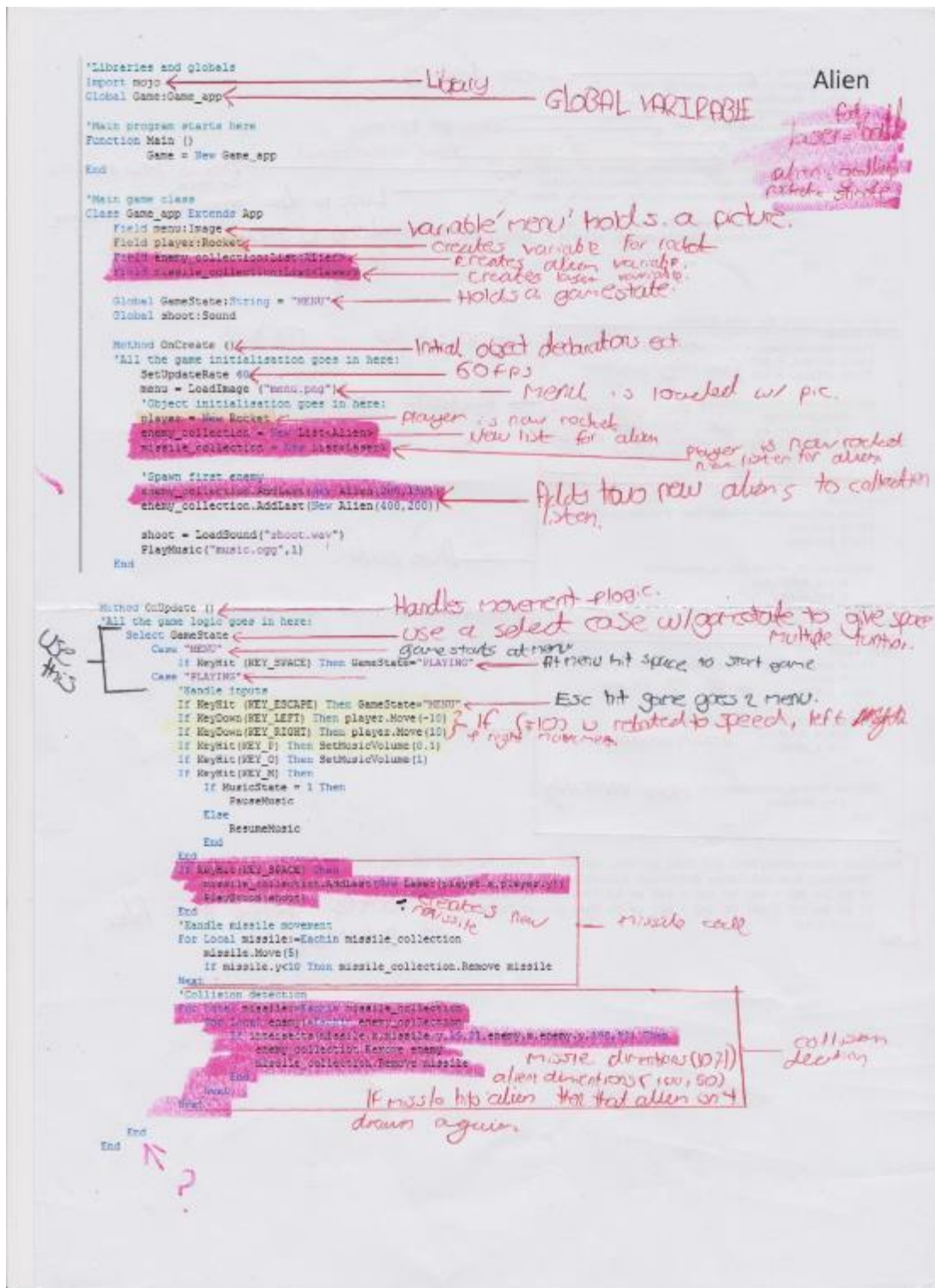
Laser image

Starting position

Vertical movement

```
Function intersects(x1:Float, y1:Float, w1:Float, h1:Float, x2:Float, y2:Float, w2:Float, h2:Float) Boolean
  'Determine how collides 2 rectangles
  If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False
  If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False
  Return True
End
```

If boxes overlap then false is returned



```

Method OnRender ()
    'All the graphics drawing goes in here:
    Select GameState
        Case "MENU"
            DrawImage menu, 0,0
        Case "PLAYING"
            cls 0, 0, 0
            DrawImage player.sprite, player.x, player.y
            For Local enemy:=EachIn enemy_collection
                DrawImage enemy.sprite, enemy.x, enemy.y
            Next
            For Local missile:=EachIn missile_collection
                DrawImage missile.sprite, missile.x, missile.y
            Next
        End
    End
End

'Class definitions for game objects:
Class Rocket
    Field sprite:Image = LoadImage ("player.png")
    Field x:Float = 300
    Field y:Float = 420
    Method Move(x_distance:Int)
        x=x_distance
        If x<0 Then x=0
        If x>590 Then x=590
    End
End

Class Alien
    Field sprite:Image = LoadImage ("alien.png")
    Field x:Float
    Field y:Float
    Method New(x_spawn:Int, y_spawn:Int)
        x = x_spawn
        y = y_spawn
    End
End

Class Laser
    Field sprite:Image = LoadImage ("missile.png")
    Field x:Float
    Field y:Float
    Method New(x_spawn:Int, y_spawn:Int)
        x = x_spawn+22
        y = y_spawn
    End
    Method Move(y_distance:Int)
        y=y_distance
    End
End

Function intersects:Bool (x1:Int, y1:Int, w1:Int, h1:Int, x2:Int, y2:Int, w2:Int, h2:Int)
    'Bounding box collision detection algorithm
    If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False
    If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False
    Return True
End

```

Screen drawing

Draw pic to menu

IP playing draw colour repped.

Draws new position to screen.

Loop to draw every alien to screen.

Loop to move through every missile in the list.

Class for a rocket

Rockets code.

starting position

The right values are added to the origin to give new horizontal movement

Alien code.

Laser code.

sprite on laser

spawn position

move vertically

If boxes overlap then false is returned.

08/01/2017 DECLARING GLOBAL VARIABLES CLASSES AND METHODS

1 Import mojo

1-This piece of code is used to import mojo which is a library and allows functions to run within the program.

2 Class Mygame Extends App

```
3 Method Oncreate()  
    SetupdateRate(60)
```

2-This code creates the class which holds the attributes and methods, the class extends the app class which comes with mojo, which is called 'App'.

```
End
```

```
4 Method Onupdate()
```

3-This method is called once as soon as an instance of this class has been created and it holds all of the game initialization such as frame rate, loading images and sound files. The setupdateRate is the number of times that Onupdate is called per second, it's the frame rate of the game which I have set to 60fps.

```
End
```

```
5 Method Onrender()
```

4-The Onupdate method is called as many times as you set it to and this is where the inputs will be handled and where the movement for the ball, shooter, goalkeeper and power bar will be coded.

```
End
```

```
End
```

6 Function Main()

```
7 New Mygame
```

5-Onrender is where all the drawing commands go, so when the football looks as if it is moving its actually being redraw at different coordinates extremely quickly.

```
End
```

6-This is the main function that holds an instance of the class created that extends the app.

7-This is the instance of the class that was created and will allow the game to run through the class.

```

Import mojo
Global Game:PenaltyGame
Class PenaltyGame Extends App
    Field menubackground:Image

    Method OnCreate()
        SetUpdateRate(60)

        menubackground = LoadImage("menubackground.png")

        'Image.Default = 0
    End

    Method OnUpdate()
    End

    Method OnRender()
        Cls(0,0,0)

        DrawImage(menubackground, 0, 0)

    End

End

Function Main()

    Game = New PenaltyGame

End

```

The menu will be the first screen that is displayed when the game is loaded therefore no conditions need to be met to render the menu. To create the menu, I used Microsoft Word to produce a screen design then I copied the design into paint, this meant I was able to manipulate the image as I could save it as a .png and resize the image in order to fit the screen without disfiguring and reducing the menu's quality.

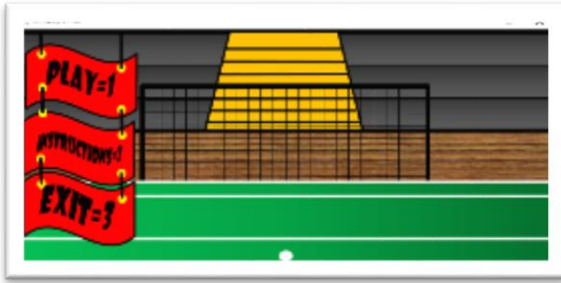
I created an attribute called menubackground and declared it as an image, this meant I was able to load the image within the OnCreate method. Here I called the image from the penaltyshoot.data file where all the images are kept where I will be able to call them. Finally, in the OnRender method I drew the image to the coordinates of 0,0 which meant the top left of the image was drawn to the top left of the window.

After the code I wrote was completed was run this was the result, this start allows me to have a base for writing code for the rest of the game, where I can work through the different game states until the game is fully functional.

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the menu load	The game is loaded	To cause the game to load	The menu loaded as predicted	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]



This was how the screen looked once the game was loaded up

```
Global GameState:String = "Menu"
Global tag:Bool = False
Method OnCreate()
    SetUpdateRate(60)

    menubackground = LoadImage("menubackground.png")

End

Method OnUpdate()
    Select GameState
        Case "Menu"
            If KeyHit (KEY_1) Then GameState= "Customize_tag"
            If KeyHit (KEY_2) Then GameState= "Instructions"
            If KeyHit (KEY_3) Then GameState= "Exit"
        End
    End
End

Method OnRender()
    Select GameState
        Case "Menu"
            DrawImage(menubackground, 0, 0)
        Case "Customize_tag"
            Cls 225,0,0
        Case "Instructions"
            Cls 0,225,0
        Case "Exit"
            Cls 0,0,225
        End
    End
End

Function Main()

    Game = New PenaltyGame

End
```



To program different areas of my penalty game I am using game states which determine which stage the game is in so specific code can then be run to make my game functional.

The first highlighted line of code shows the global variable which I declared and set to "menu", this means every time the game is run the gamestate is automatically set to "menu" which allows the code for the menu to be run. The data type was set to string although it could have been set to char as the label "menu" is only a way of easily identifying the required code for the main menu.

The highlighted code within the OnUpdate method is a case statement where the GameState is "menu" where the keys 1,2 or 3 are waiting to be pressed. If 1 is pressed then the gamestate is changed to customize_tag which is the stage prior to the penalty game itself, if 2 is selected then the instructions are pulled up as the code within the "instructions" gamestate cause that action to happen. Finally, if 3 is hit then the gamestate is changed to "exit" where the game will close down.

The selected code within the OnRender method uses a select case like the one within the OnUpdate method, however this code within this method allows for graphics to be written to the screen. If the case is "menu" then the screen design I made for the menu is rendered, in addition if the case is "customize_tag" then the screen will turn red, if the case is "instructions" then the screen would be given the colour green, finally if the gamestate is changed to "exit" then the colour blue is rendered. The different coloured screens allowed for me to test whether the game was functioning properly.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the Gamestate change correctly	Key 1	This is the key that will start the game	The screen turns red as the gamestate changes to customize_tag	NA
Does the Gamestate change correctly	Key 2	This key loads the instruction menu	The screen turns green where the gamestate changes to instructions	NA
Does the Gamestate change correctly	Key 3	This key will cause the game to exit	The screen changes to blue as the gamestate is changed to exit	NA

```

Import mojo
Global Game:PenaltyGame
Class PenaltyGame Extends App
    Field menubackground:Image
    Field gamebackground:Image

    Global GameState:String = "Menu"
    Global tag:Bool = False
    Method OnCreate()
        'frame rate
        SetUpdateRate(60)
        'menu background
        menubackground = LoadImage("menubackground.png")
        gamebackground = LoadImage("gamebackground.png")
    End

    Method OnUpdate()
        Select GameState
            Case "Menu"
                If KeyHit (KEY_1) Then GameState= "Playing"
                If KeyHit (KEY_2) Then GameState= "Instructions"
                If KeyHit (KEY_3) Then GameState= "Exit"
        End
    End

    Method OnRender()
        Select GameState
            Case "Menu"
                DrawImage(menubackground, 0, 0)
            Case "Customize_tag"
                Cls 225,0,0
                If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
                If Not tag = False Then GameState = "Playing"
            Case "Instructions"
                Cls 0,225,0
            Case "Exit"
                Cls 0,0,225
            Case "Playing"
                DrawImage(gamebackground, 0, 0)
        End
    End

End
End

```

The image I am importing is one I created using Microsoft Word and will be the background for my game while penalties are being taken.

Firstly I created an attribute which declared that the "gamebackground" is an image, this meant I could load "gamebackground" and draw it to the screen.

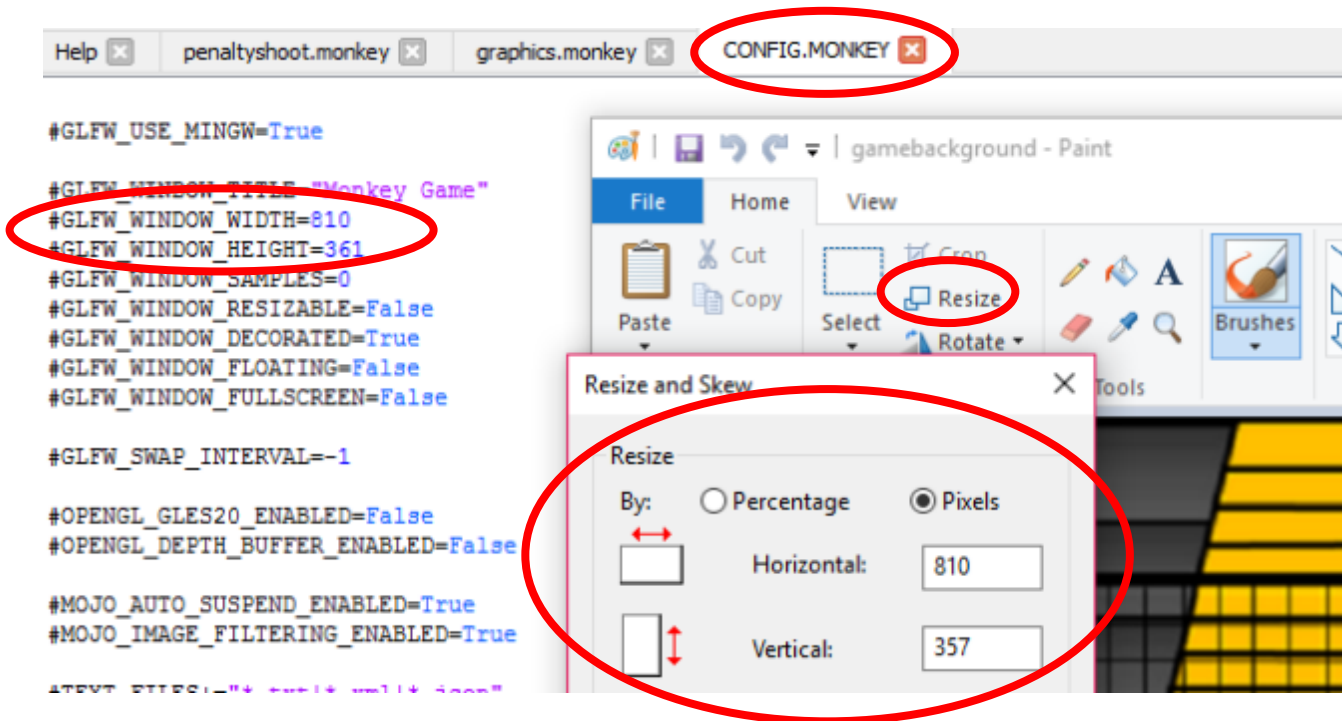
The second highlight shows that I am loading the image within the Oncreate method this allows me to draw the image to the screen later on in the Onrender method.

The third and final step for this process involved me writing the code to draw the image to the screen within the "playing" case, this ensured that when the is under way that "gamebackground" is rendered.



This shows what the game current looks like once "play" is selected

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the background load when the gamestate playing is selected	Key 1	This key results in the playing gamestate to run	The background loaded as expected	NA



This screenshot shows what I did to make sure the console application correctly fitted with the gamebackground, firstly I took a screenshot of the background I designed then I pasted it into paint where I cropped the image down to where only the background remained. After that I opened the CONFIG.MONKEY tab which was in the .buildv84f folder, here you can change the characteristics of the console application. Finally, I set the dimensions within monkey x equal to the dimensions show within paint, however the window and background weren't properly aligned so I changed this dimensions until they did.

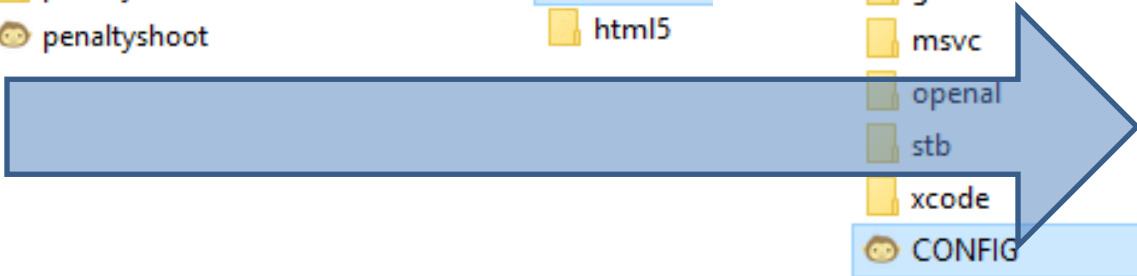
Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

penaltyshoot.buildv84f
penaltyshoot.data
penaltyshoot

glfw
glfw3
html5

gcc_winnt
glfw3
msvc
opengl
stb
xcode
CONFIG
main.cpp
main.h
main.mm



This is testing for development where the testing uses data that is valid, invalid and borderline, i have made some changes to the tests for ther

Menu

Test data (Input)	Type	Outcome
Select play	Valid	Game loads
Select instruction	Valid	Green screen renders
Select exit	Valid	Game closes
Click anywhere there is no button	Invalid	No action
Select play(LMC)	Invalid	No action
Select instruction(LMC)	Invalid	No action
Select exit(LMC)	Invalid	No action

Playing shooting

Test data	Type	Outcome
'P'	Valid	Feature not implemented
'O'	Invalid	No action
'Left arrow'	Borderline	No action
'Right arrow'	Borderline	No action
'Spacebar'	Valid	Shot is taken
'W'	Valid	Crosshair moves up
'A'	Valid	Crosshair moves left
'S'	Valid	Crosshair moves down
'D'	Valid	Crosshair moves right

Playing goalkeeper

Test data	Type	Outcome
'P'	Valid	Feature not implemented
'O'	Invalid	No action
'Left arrow'	Valid	Goalkeeper moves left
'Right arrow'	Valid	Goalkeeper moves right
'Spacebar'	Borderline	No action
'W'	Borderline	No action
'A'	Borderline	No action
'S'	Borderline	No action
'D'	Borderline	No action
'P'	Borderline	No action

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

These game states were not fully completed within my game so testing cannot be achieved.

Over

Test data	Type
Game won	Valid
Game lost	Valid
Game draw	Valid
Game exit	Valid
'Spacebar'	Invalid

Instruction

Test data	Type
Select return	Valid
'P'	Invalid
'A'	Invalid

This is an interview with the client Billy Evans where we discuss the prototype of the game setup.

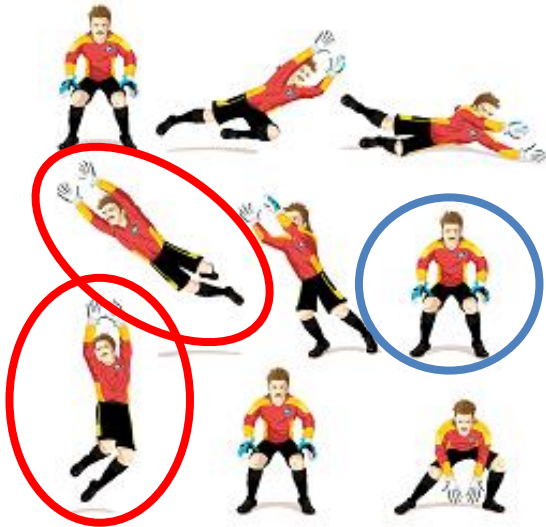
Requirement	Justification
The main menu will have three options of PLAY/INSTRUCTIONS/QUIT	These are the essential options as if the user is unable to play the game there is no point of creating it, these options are a clear way of directing the user to selecting what they wish to proceed with.
Background image of a goal within a stadium	Although my game isn't realistic as other games such as fifa 16 due to the cartoon graphics and other the movement effects etc, the stadium means that I can have an element of realism with the game. This is also the background which Billy picked within my interview as well.
'Left mouse click' is used to navigate through the menu's and return buttons	This is a simple and effective way of navigating through menus of the game.

Me: I was able to fulfil most of the requirements within this area of the development as I have designed and implemented the main menu which gives options to 'PLAY', load 'INSTRUCTIONS' or 'EXIT'. In addition, I have also used the design I created for the game while it is being played. However I decided it would be more suitable for the navigation to be performed by pressing keys due to my programming ability as I am unfamiliar with the implementation of buttons for Monkey X and I did not want to get behind on the development right at the start of it.

Billy: I am happy to see the development has begun to progress and it is making a strong start, I agree with the fact that keys should be used instead of buttons if it suits your programming style better, I look forward to the next prototype.

MOVEMENT

19/01/2017 EDITING SPRITES, CREATING A CLASS FOR THE GOALKEEPER AND GOALKEEPER MOVEMENT



This is the sprite sheet I used within my game, originally, I planned to change between different sprites of the goalkeeper dependent on whether the goalkeeper jumps vertically upwards, dives left (jump and left) or dives right (jump and right). However, I will only be able to implement this if I have enough time once the game is functional this is as I am unable to tell if I will run into any issues while coding the rest of the game. Because of this I picked a sprite that is suitable for all scenario of shots.

```

Import mojo
Global Game:PenaltyGame
Class PenaltyGame Extends App
  Field menubackground:Image
  Field gamebackground:Image

  Field saver:goalkeeper

  Global GameState:String = "Menu"
  Global tag:Bool = False

  Method OnCreate()
    'frame rate
    SetUpdateRate(60)
    'menu background
    menubackground = LoadImage("menubackground.png")
    gamebackground = LoadImage("gamebackground.png")

    'object initialisation
    saver=New goalkeeper

End

```

Here I created an attribute for my sprite and named it saver (goalkeeper), then I declared saver to goalkeeper which is the class I created for the sprite.

Then in the Oncreate method I called the goalkeeper class to make a new save/sprite, this is the process for instantiation.


```

Method OnUpdate()
    Select GameState
        Case "Menu"
            If KeyHit (KEY_1) Then GameState= "Playing"
            If KeyHit (KEY_2) Then GameState= "Instructions"
            If KeyHit (KEY_3) Then GameState= "Exit"
        Case "Playing"
            If KeyHit (KEY_LEFT) Then saver.Move(-10)
            If KeyHit (KEY_RIGHT) Then saver.Move(10)
    End
End

Method OnRender()
    Select GameState
        Case "Menu"
            DrawImage (menubackground, 0, 0)
        Case "Customize_tag"
            Cls 225,0,0
            If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
            If Not tag = False Then GameState = "Playing"
        Case "Instructions"
            Cls 0,225,0
        Case "Exit"
            Cls 0,0,225
        Case "Playing"
            DrawImage(gamebackground, 0, 0)
            DrawImage saver.sprite, saver.x, saver.y
    End

End

End

'Class for goalkeeper
Class goalkeeper
    Field sprite:Image = LoadImage ("goalkeeper.png")
    Field x:Float = 370
    Field y:Float = 120

    Method Move(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>545 Then x=545
    End
End

Function Main()

    Game = New PenaltyGame

End

```

Within OnUpdate I can program the movement for the goalkeeper, so when the left arrow is pressed the sprite moves left by 10 pixels, if the right arrow is hit then the sprite moves right by ten pixels. I am coding this in the OnUpdate method as this is where all the game logic is written.

The OnRender method holds the code that draws the "saver" to the screen.

I have made a class for the goalkeeper, within this class the attributes are set that load the image of the sprite and the sprites starting positions are set, I set these to the center of the goal with the goalkeeper stood on the goal line. The method move holds the boundaries in which the sprite cannot horizontally move past, I set these boundaries to be the goalposts, the $x+=x_distance$ changes the position of the goalkeeper by adding and subtracting 10 dependent on whether the movement is left or right.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

What is being tested?	Input	Justification of input	Outcome	How to solve
Left movement of the goalkeeper	Left arrow key	This is what I have programmed to change the horizontal coordinates of the goalkeeper	The goalkeeper shunts left a small amount when the key is pressed	NA
Right movement of the goalkeeper	Right arrow key	This is what I have programmed to change the horizontal coordinates of the goalkeeper	The goalkeeper moves right a small amount when the key is hit	NA



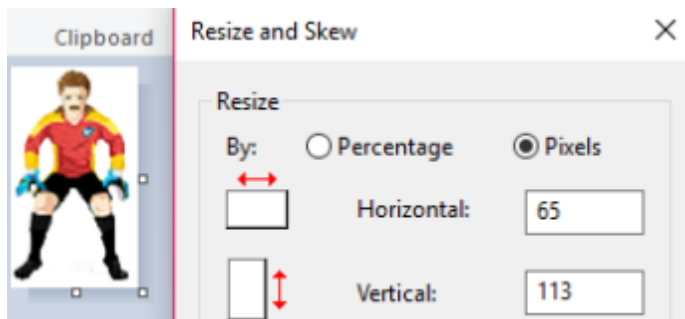
This screenshot shows the current stage I am at with the development of the “playing” gamestate where I have the sprite “saver” moving along the goal line. I have yet to remove the background of the sprite.



Here I used the eraser tool in fireworks to remove the white backing to the goalkeeper, I initially had a problem trying to remove it as I was using paint, eventually I realized that I would have to use different software to achieve this.



Below you can see the stage at which my game is currently at where the goalkeeper can move left and right when the correct controls are pressed.



I also had to resize the image so that the sprite would be to scale against the goal.

What is being tested?	Input	Justification of input	Outcome	How to solve
The removal goalkeeper background	Key 1	Enables the goalkeeper sprite to load	There were a few white patch around the goalkeeper	To solve this I used the wand feature on fireworks and selected and deleted the light grey areas that weren't detected on the original background removal.

```

Import mojo
Global Game:PenaltyGame
Class PenaltyGame Extends App
    Field menubackground:Image
    Field gamebackground:Image
    Field saver:goalkeeper
    Field target:shooter

    Global GameState:String = "Menu"
    Global tag:Bool = False
    Global shooting = True

    Method OnCreate()
        'frame rate
        SetUpdateRate(60)
        'menu background
        menubackground = LoadImage("menubackground.png")
        gamebackground = LoadImage("gamebackground.png")

        'object initialisation
        saver=New goalkeeper
        target=New shooter

    End

    Method OnUpdate()
        Select GameState
            Case "Menu"
                If KeyHit (KEY_1) Then GameState= "Playing_gk"
                If KeyHit (KEY_2) Then GameState= "Instructions"
                If KeyHit (KEY_3) Then GameState= "Exit"
            Case "Playing_gk"
                If KeyHit (KEY_LEFT) Then saver.Move(-10)
                If KeyHit (KEY_RIGHT) Then saver.Move(10)
                If KeyHit (KEY_7) Then shooting = False
                If shooting = False Then GameState= "Playing_s"
            Case "Playing_s"
                If KeyHit (KEY_A) Then target.Move(-10)
                If KeyHit (KEY_D) Then target.Move(10)

        End

    End

    Method OnRender()
        Select GameState
            Case "Menu"
                DrawImage(menubackground, 0, 0)
            Case "Customize_tag"
                Cls 225,0,0
                If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
                If Not tag = False Then GameState = "Playing_gk"
            Case "Instructions"
                Cls 0,225,0
            Case "Exit"
                Cls 0,0,225
            Case "Playing_gk"
                DrawImage(gamebackground, 0, 0)
                DrawImage saver.sprite, saver.x, saver.y
            Case "Playing_s"
                DrawImage (gamebackground, 0,0)
                DrawImage target.sprite, target.x, target.y

        End

    End
End

```

Firstly, I named the sprite "target" and declared it to the class called "shooter".

Then I created a global variable where the shooter is to be "true" this means that the user will be able to shoot at goal while "shooting" is true, on the contrary when "shooting" equals false then the player will be switched to the goalkeeper and will attempt to save the goal.

Within the Oncreate method I used instantiation again where I can run the code for class "shooter".

The yellow highlighted lines show that I have adapted the gamestate "Playing" by creating two new states called "Playing_gk" which will hold the code required while the player is in goal, so it handles the goalkeeper's movement and later in OnRender it will allow the "saver" sprite to be drawn to the screen. Whereas the "Playing_s" gamestate holds the code that is necessary to the player whilst they are shooting at goal, although I haven't implemented all the features of the game these different states are essential to prevent confusion within the development of the game. The code within the "Playing_gk" state shows that I have made a way of manually switching from the goalkeepers to the shooter's perspective, I have done this as a way of testing each sprite moves properly, and have done this by changing states when the key 7 is pressed.

As I explained above the different Gamestates mean I can render different sprites to the screen at different times, in my game I will need to have the target rendered while the player is shooting and then to only have the goalkeeper rendered while the player is the goalkeeper.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

```

'Class for goalkeeper
Class goalkeeper
    Field sprite:Image = LoadImage ("goalkeeper.png")
    Field x:Float = 370
    Field y:Float = 125

    Method Move(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>545 Then x=545
    End
End

```

```

'Class for crosshairs
Class shooter
    Field sprite:Image = LoadImage ("target3.png")
    Field x:Float = 370
    Field y:Float = 125

    Method Move(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>555 Then x=555
    End
End

```

```

Function Main()

    Game = New PenaltyGame

End

```

For the class “shooter” I have repeated the process from the class “goalkeeper” where the movements are identical, the only difference is that I had to change the boundaries for where the targets could move as they originally exceeded the goalposts. I intended on having the target move vertically as well as horizontally, however I wasn’t able to implement this as I don’t know how to increase the coordinates specifically to the y axis even though I was able to for the x axis.

In order, to solve this I tried to set target. Move to a specific axis but an error kept popping up, as shown below.

What is being tested?	Input	Justification of input	Outcome	How to solve
Rendering of the target without it's background	Key 1	Key 1 allows the sprite to be rendered	The target renders as expected	NA
Left horizontal movement of the target	Key A	This is the key that will cause the change in the targets coordinates so it moves left	The target moves left at the expected rate	NA
Right horizontal movement of the target	Key D	This is the key that will cause the change in the targets coordinates so it moves right	The target moves right at the expected rate	NA
Upward vertical movement of the target	Key W	This is the key that will cause the change in the targets	There is no movement when the key is pressed	Created a new screen design where the vertical

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

		coordinates so it moves upwards		movement of the target is not required
Downward vertical movement of the target	Key S	This is the key that will cause the change in the targets coordinates so it moves downwards	There is no movement when the key is pressed	Created a new screen design where the vertical movement of the target is not required


Case "Playing_s"

```
If KeyHit (KEY_A) Then target.Move(-10)
If KeyHit (KEY_D) Then target.Move(10)
```

Original code

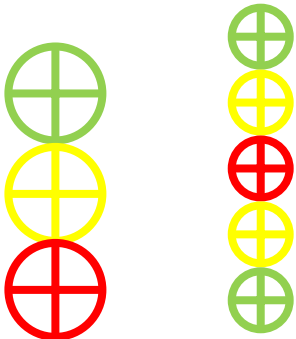
Case "Playing_s"

```
If KeyHit (KEY_A) Then x_target.Move(-10)
If KeyHit (KEY_D) Then x_target.Move(10)
If KeyHit (KEY_W) Then y_target.Move(10)
If KeyHit (KEY_S) Then y_target.Move(-10)
```

 Compile Error


Identifier 'x_target' not found.

Adapted code with error



Target2

Target3

Because of not being able to implement the vertical movement of the target I had to find an alternate way for the target to reach different heights of the goal so I designed a new target which enabled different power levels to be selected, however I realized that this was not the most effective design as the yellow power level which will be harder to select than the red power actually produces a worse shot compared to the red. To fix this I created another target as shown to the left.



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

This design didn't follow the importance of the power which meant that the logic behind the game would show flaws as the red and green targets are better shots whereas the yellow target produces shots that are easier to save.



The design here is more effective than target 2 however, it looks too confusing for a game that is meant to be aimed for 5-14 year olds, I have chosen to go with target 3 as it is more effective than the second one.

What is being tested?	Input	Justification of input	Outcome	How to solve
Left vertical movement of the targets	Key A	This key is programmed to move the targets left	The target moves left	NA
Right vertical movement of the targets	Key D	This key is programmed to move the targets right	The target moves right	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

31/01/2017 SOLUTION TO THE MOVEMENT OF THE TARGET

While looking over my program I noticed that I missed something that allowed me to implement vertical motion for the target.

```

Method OnUpdate()
    Select GameState
        Case "Menu"
            If KeyHit (KEY_1) Then GameState= "Playing_gk"
            If KeyHit (KEY_2) Then GameState= "Instructions"
            If KeyHit (KEY_3) Then GameState= "Exit"
        Case "Playing_gk"
            If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
            If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
            If KeyHit (KEY_7) Then shooting = True
            If shooting = True Then GameState= "Playing_s"
        Case "Playing_s"
            If KeyHit (KEY_A) Then target.Move_x(-10)
            If KeyHit (KEY_D) Then target.Move_x(10)
            If KeyHit (KEY_S) Then target.Move_y(10)
            If KeyHit (KEY_W) Then target.Move_y(-10)
    End Select

'Class for goalkeeper
Class goalkeeper
    Field sprite:Image = LoadImage ("goalkeeper.png")
    Field x:Float = 370
    Field y:Float = 125

    Method Move_x(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>545 Then x=545
    End Method
End Class

'Class for crosshairs
Class shooter
    Field sprite:Image = LoadImage ("target.png")
    Field x:Float = 370
    Field y:Float =125

    Method Move_x(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>560 Then x=560
    End Method

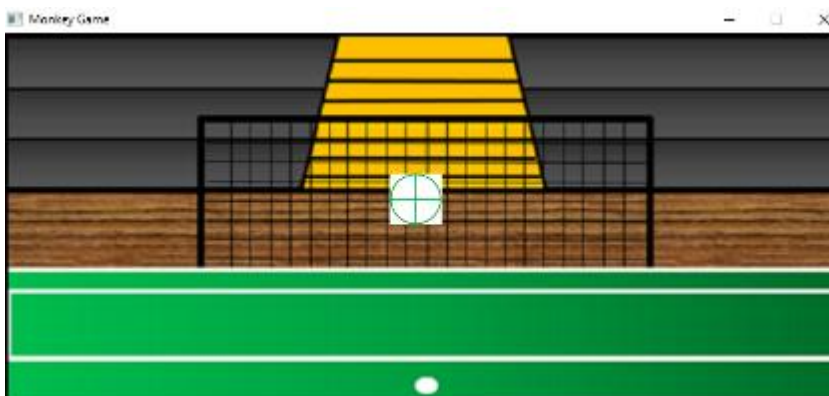
    Method Move_y(y_distance:Int)
        y+=y_distance
        If y<80 Then y=80
        If y>175 Then y=175
    End Method
End Class

```

I noticed that the "saver.Move" was the same as the "Move", originally I thought that it was the word "Move" that allowed for the change in the coordinates, in order to test whether this was the case I changed the word within the OnUpdate method and the goalkeeper class and the game still run properly. After gaining this insight I copied the method which is now called "Move_x" and replaced x with y, then I updated the controls for the target within the gamestate "Playing_s" which now means WASD is used to move the target. Finally, I changed the y axis boundaries via trial and error to prevent the target from leaving the goal.

I decided to use my original design for the target as was approved by Billy Evans as it makes the game look more simplistic and more appealing to children between the ages 5-14.

These screenshots show the two playing Gamestates and where they stand within the development so far



"Playing_s"
gamestate
development



"Playing_gk"
gamestate
development

What is being tested?	Input	Justification of input	Outcome	How to solve
Left horizontal movement of the target	Key A	This is the key that will cause the change in the targets coordinates so it moves left	The target moves left at the expected rate	NA
Right horizontal movement of the target	Key D	This is the key that will cause the change in the targets coordinates so it moves right	The target moves right at the expected rate	NA
Upward vertical movement of the target	Key W	This is the key that will cause the change in the targets coordinates so it moves upwards	The target moves upwards as expected	NA
Downward vertical movement of the target	Key S	This is the key that will cause the change in the targets coordinates so it moves downwards	The target moves downwards as expected	NA

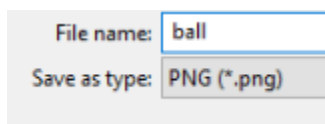
Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

01/02/17 CREATING THE BALL SPRITE AND ITS CLASS



This is the sprite that I am using for my game, here I am cropping the image down using paint where I am also able to resize it to a scale that fits the window proportionally. When saving the image, you need to save it as a .png and in the .data file as that is where the program will only load it from.



```

Import mojo
Global Game:PenaltyGame
Class PenaltyGame Extends App
  Field menubackground:Image
  Field gamebackground:Image
  Field saver:goalkeeper
  Field target:shooter
  Field ball:football

  Global GameState:String = "Menu"
  Global tag:Bool = False
  Global shooting = True

  Method OnCreate()
    'frame rate
    SetUpdateRate(60)
    'menu background
    menubackground = LoadImage("menubackground.png")
    gamebackground = LoadImage("gamebackground.png")

    'object initialisation
    saver=New goalkeeper
    target=New shooter
    ball=New football

End

```

To create the football sprite, I firstly made an attribute called ball which was set to the "football" class, then as before I use instantiation to make an object, in this case it's the ball.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

```

Method OnRender()
  Select GameState
    Case "Menu"
      DrawImage(menubackground, 0, 0)
    Case "Customize_tag"
      Cls 225,0,0
      If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
      If Not tag = False Then GameState = "Playing_gk"
    Case "Instructions"
      Cls 0,225,0
    Case "Exit"
      Cls 0,0,225
    Case "Playing_gk"
      DrawImage(gamebackground, 0, 0)
      DrawImage saver.sprite, saver.x, saver.y
    Case "Playing_s"
      DrawImage (gamebackground, 0,0)
      DrawImage target.sprite, target.x, target.y
      DrawImage ball.sprite, ball.x, ball.y
  End
End

```

Here I drew the sprite "ball" to the screen and spawned it on the penalty spot, this took a lot of testing via trial and error as the coordinates were difficult to locate.

```

'Class for football
Class football
  Field sprite:Image = LoadImage ("ball.png")
  Field x:Float = 375
  Field y:Float = 295
End

```

What is being tested?	Input	Justification of input	Outcome	How to solve
The render of the ball sprite	Key 1	The game loads into the gamestate where the ball sprite should be	The ball sprite is render in the correct gamestate with the right spawn coordinates and image sizing	NA

```

Method OnUpdate()
  Select GameState
    Case "Menu"
      If KeyHit (KEY_1) Then GameState= "Initialise"
      If KeyHit (KEY_2) Then GameState= "Instructions"
      If KeyHit (KEY_3) Then GameState= "Exit"
    Case "Initialise"
      ball.speed = 5
      target.x=370
      target.y=125
      ball.x=375
      ball.y=295
      GameState = "Playing_s"
    Case "Playing_gk"
      If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
      If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
      If KeyHit (KEY_7) Then shooting = True
      If shooting = True Then GameState= "Playing_s"
    Case "Playing_s"

      'moving between shooter and keeper
      If KeyHit (KEY_8) Then shooting = False
      If shooting = False Then GameState= "Playing_gk"

      If KeyHit (KEY_A) Then target.Move_x(-10)
      If KeyHit (KEY_D) Then target.Move_x(10)
      If KeyHit (KEY_S) Then target.Move_y(10)
      If KeyHit (KEY_W) Then target.Move_y(-10)

      If KeyHit (KEY_SPACE)
        While target.x<>ball.x Or target.y<>ball.y
          If ball.x < target.x Then ball.x +=ball.speed
          If ball.x > target.x Then ball.x -=ball.speed
          If ball.y < target.y Then ball.y +=ball.speed
          If ball.y > target.y Then ball.y -=ball.speed
        End
      End
  End

```

```

If KeyHit (KEY_A) Then target.Move_x(-10)
If KeyHit (KEY_D) Then target.Move_x(10)
If KeyHit (KEY_S) Then target.Move_y(10)
If KeyHit (KEY_W) Then target.Move_y(-10)
If KeyHit (KEY_SPACE) Then GameState = "Ball_Move"

Case "Ball_Move"
  If ball.x < target.x Then ball.x +=ball.speed
  If ball.x > target.x Then ball.x -=ball.speed
  If ball.y < target.y Then ball.y +=ball.speed
  If ball.y > target.y Then ball.y -=ball.speed
End
End

```

This code here shows the final outcome that I have for the task of coding the ball to move from the penalty spot to the target when commanded to. Firstly, I created new GameState called "initialize" which is where I had the code for the coordinates of the ball and target, in addition I had the speed of the ball as well although this is not a necessary piece of code. Within the "initialize" GameState I changed the GameState to "Playing_s" so it automatically changes, once the coordinates and speeds have been set.

The block of code I used here is an adapted concept from the CraignDave maze runner game tutorial, this code means I am able to move the ball from the penalty spot to the target. I used a loop in order to repeat the process of moving the ball closer to the target. I used the space bar as a way to shoot the ball as that is what I had set originally to shoot the ball within the analysis and design.

This first thing I tried to run the section of code on command was with this extract of code here, the highlighted code shows that I attempted to solve the problem by creating a GameState. I did this by setting the Space bar to change the Gamestate from "Playing_s" to "Ball_Move" which in theory should run the code and move the ball from the penalty spot to the target. However, an error occurred where there would be an error within the methods, I discovered that it was down to the amount of ends I had at the end of the select statement. Although I had solved this error, the program continued not to function properly as when the game loaded up the ball would not move and the game would glitch.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the ball move towards the penalty spot (first method)	Key Space	The space bar is stated in the requirement specification to shoot the ball towards the target	The game loaded up but the ball would not move and the game would glitch	Find alternative way to move the ball towards the penalty spot
Does the ball move towards the penalty spot (second method)	Key Space	The space bar is stated in the requirement specification to shoot the ball towards the target	The ball does move towards the target however, the ball jumps to the target instead of moving smoothly and realistically	I will return to this issue once I have made the game functional as I don't want to excessively spend time on the movement of the ball at risk of the game not getting completed

Goalkeeper testing

Test data	Outcome
'Left arrow'	Goalkeeper moves left
'Right arrow'	Goalkeeper moves right
'Spacebar'	Goalkeeper remains still
'W'	Goalkeeper remains still
'A'	Goalkeeper remains still
'S'	Goalkeeper remains still
'D'	Goalkeeper remains still
'P'	No action occurs

Although these tests are accurate they won't be reliable at the end of the project as I have yet to implement the pause function (P), and the code so the goalkeeper is able to jump when space bar is pressed. This is down to the time scales for my project as I am unable to predict if I will run into any problems that may pull me behind target, therefore I will leave the 'jump' function until I have a functional game.

Crosshair testing

Test data	Outcome
'W'	Crosshair moves up
'A'	Crosshair moves left
'S'	Crosshair moves down
'D'	Crosshair moves right
'Spacebar'	Crosshair remains still
'P'	Crosshair remains still
'R'	Crosshair remains still
'3'	Crosshair remains still

As mentioned before I have not implemented the paused feature as of yet.

I have conducted an interview with Billy Evans in order to successfully perform iterative software development, this interview is based around the prototype of movement within the game. This interview involves going through the requirement specification and running the prototype, then discussing alterations that need to be made if any.

Requirement	Justification
<ul style="list-style-type: none"> • “ (space bar)- Applies power to the shot • ‘A’ moves the crosshairs for the shot to the left • ‘D’ moves the crosshairs for the shot to the right • ‘S’ moves the crosshairs for the shot to the down • ‘W’ moves the crosshairs for the shot to the up 	<p>The space bar is an effective way of applying power to the ball, as the longer it is held down the more power is given.</p> <p>‘W,A,S,D’ is used inside of the arrows to avoid confusion with in the game.</p>
‘Left arrow’(move left) , ‘Right arrow’(move right) and ‘space’(jump) will control the goal keeper	These are simple and clear controls which mean my audience will be able to play the game without an issues which may prevent their interests in my game.
The goal keeper moves at a speed where it takes two seconds to move from one goal post to the other	This is a reasonable speed as it’s not as fast as the ball which means the person shooting has a very possible chance of scoring.
The ball takes one second to move from the penalty spot to the goal	This is so the shooter has a higher chance of scoring than the goalkeeper has with saving the ball.

Me: After the initial implementation of the movement I discovered that the time frame for the project would be slim as there is a lot to do in not much time as a result of this realisation I have kept the movement features to a functional level.

I have followed most of the requirements including :

- Space bar to shoot the ball the target
- A, to move the crosshairs left
- D, to move the crosshairs right
- S, to move the crosshairs downwards
- W, to move the crosshairs upwards
- Left arrow, to move the goalkeeper left
- Right arrow, to move the goalkeeper right

I was unable to implement the power feature within the game as that was above my ability of programming of monkey X and it would have consumed too much time attempting to find a solution. In addition, I found that the only movement necessary to the goalkeeper is the left and right horizontal movement, due to this I decided not to attempt to include the jumping feature of the goalkeeper until I have a functioning game as I am worried I will not reach the deadline for the game development. The requirements for the speed of the goalkeeper and the speed of the ball where not implemented as I do not have a good enough understanding of Monkey X to implement the speeds of the sprites with the time I have for the development.

Billy: Although not all of the features that were set are not in the game I am not disappointed with the outcome, I

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

agree with not using the jumping feature for the goalkeeper as that would make the game a lot more complicated and harder to play as the goalkeeper would have to predict which way the shot is going and then at what height it will the target. However, I am slightly disappointed that power selection for the shot was unable to be created, if there is time after the game is functional I would like you to focus on making power selection possible.

TURNS

02/03/2017 MOVING BETWEEN TURNS

```

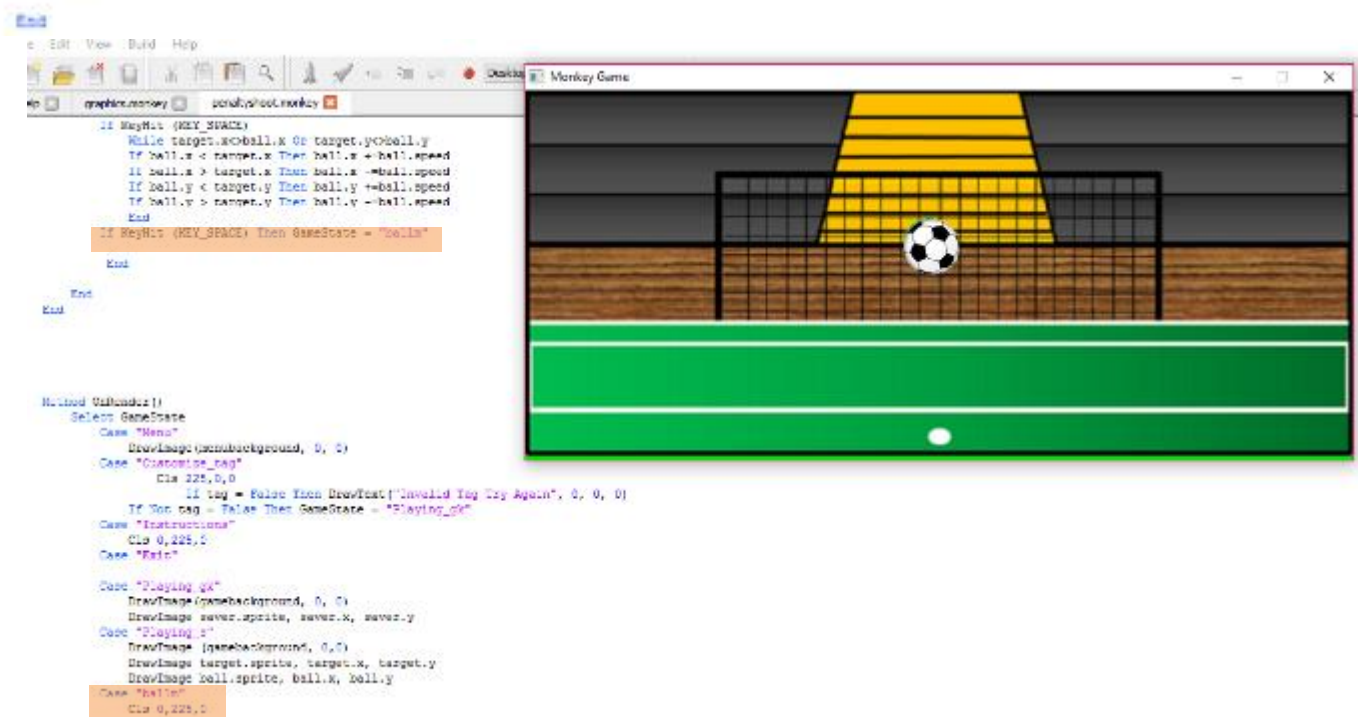
Case "Playing_s"
    'moving between shooter and keeper
    If KeyHit (KEY_S) Then shooting = False
    If shooting = False Then GameState= "Playing_gk"

    If KeyHit (KEY_A) Then target.Move_x(-10)
    If KeyHit (KEY_D) Then target.Move_x(10)
    If KeyHit (KEY_S) Then target.Move_y(10)
    If KeyHit (KEY_W) Then target.Move_y(-10)

    If KeyHit (KEY_SPACE)
        While target.x < ball.x Or target.y < ball.y
            If ball.x < target.x Then ball.x += ball.speed
            If ball.x > target.x Then ball.x -= ball.speed
            If ball.y < target.y Then ball.y += ball.speed
            If ball.y > target.y Then ball.y -= ball.speed
        End
        GameState = "Playing_gk"
    End
End

```

Firstly, I tried to change the gamestate straight after the code that moves the ball from the penalty spot to the target however whenever I would run the game the game would freeze once I hit space bar to take the shot.



For my second attempt to develop a solution for changing turns I set a test to check if the gamestate would change where if the gamestate changes to "ballm" then the screen would be cleared to green, as shown in the screenshot above the layer under the playing background was changed to green. However, the whole screen was not cleared to green which means the test was a fail.

```

target.y=100
ball.x=375
ball.y=295
GameState = "Playing_s"
Case "Playing_gk"
If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
If KeyHit (KEY_M) Then shooting = True
If shooting = True Then GameState= "Playing_s"
Case "Playing_s"

'moving between shooter and keeper
If KeyHit (KEY_N) Then shooting = False
If shooting = False Then GameState= "Playing_gk"

If KeyHit (KEY_A) Then target.Move_x(-10)
If KeyHit (KEY_D) Then target.Move_x(10)
If KeyHit (KEY_S) Then target.Move_y(10)
If KeyHit (KEY_W) Then target.Move_y(-10)

If KeyHit (KEY_SPACE)
While target.x<>ball.x Or target.y<>ball.y
If ball.x < target.x Then ball.x +=ball.speed
If ball.x > target.x Then ball.x -=ball.speed
If ball.y < target.y Then ball.y +=ball.speed
If ball.y > target.y Then ball.y -=ball.speed
End
End
End
End

```

As I was struggling to find a solution I decided to make it so the user manually changes the turns by pressing "n" to switch between shooting and saving, if I have time once I have finished the code I will attempt to make the turns switch automatically which will improve the quality of my game. For the code itself I have programmed it so that the variable "shooting" changes to false and once it is false the gamestate changes to "Playing_gk" where the user becomes the goalkeeper.

What is being tested?	Input	Justification of input	Outcome	How to solve
The gamestate changes once the space bar is pressed	Key space	I used a line of code that runs in sequence to the space bar being pressed	Game would freeze once I hit space bar to take the shot	Try a new method which involves gamestates where the space key changes the gamestate as well as taking the shot
The gamestate changes once the space bar is pressed	Key space	The gamestate should change once the space bar is pressed	The gamestate would change however the screen designs would not change	Create manual turn switching
The gamestate changes when 'n' is pressed	Key n	N is the key I have decided to set as the turn changer as this key is unlikely to be hit by accident which could have a negative effect on the game	The gamestate changes as expected	NA

Once I had the turns switching between goalkeeper and shooter I needed to make a limit for the number of shots taken from the CPU and the user, Billy Evan which is representing my target audience prefers the option where the user and CPU would take 5 shots each.

```

End
Turn = 0

Method OnUpdate()
    Select GameState
        Case "ballm"
            GameState = "Playing_gk"
        Case "Menu"
            If KeyHit (KEY_1) Then GameState= "Initialise"
            If KeyHit (KEY_2) Then GameState= "Instructions"
            If KeyHit (KEY_3) Then GameState= "Exit"
        Case "Exit"
            If GameState = "Exit" Then Error("")
        Case "Initialise"
            ball.speed = 5
            target.x=370
            target.y=125
            ball.x=375
            ball.y=295
            GameState = "Playing_s"
        Case "Playing_gk"
            If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
            If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
            Turn = Turn + 1
            If Turn <> 5 Then GameState = "Playing_s"
            If KeyHit (KEY_M) Then shooting = True
            If shooting = True Then GameState= "Playing_s"
        Case "Playing_s"
            'moving between shooter and keeper
            If KeyHit (KEY_N) Then shooting = False
            If shooting = False Then GameState= "Playing_gk"

            If KeyHit (KEY_A) Then target.Move_x(-10)
            If KeyHit (KEY_D) Then target.Move_x(10)
            If KeyHit (KEY_S) Then target.Move_y(10)
            If KeyHit (KEY_W) Then target.Move_y(-10)
    End Select
End

```

To make the desired 5 shots each possible I created a variable called "turns" as an Integer and set it to zero within the OnCreate method. Once the user was within the code for being the goalkeeper I incremented the variable "turn" by 1, then the line of code after that contained an IF statement where if turns was not equal to 5 then move to the code for the user to take a shot at goal. However, this solution was not effective as the game would continue regardless of the number of turns.

What is being tested?	Input	Justification of input	Outcome	How to solve
The limit to the number of turns within the game	Key n	This key allows manual turn changing	There was no change in the game as the turns could be switched an infinite number of times	Introduce Boolean variables which in theory will cap the number of turns taken

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

In order to solve this issue I tried another approach where is said if variable “turns” is more than 4 then “Next_turn” is False, and then I used a line, which said If Next_turn is False Gamestate is “end” in theory this should count the number of times the user plays as the goalkeeper and once it hits 5 the game should end. I had set the “end” gamestate to render a black screen, however the game would render the black screen as soon as the “n” was pressed.

```

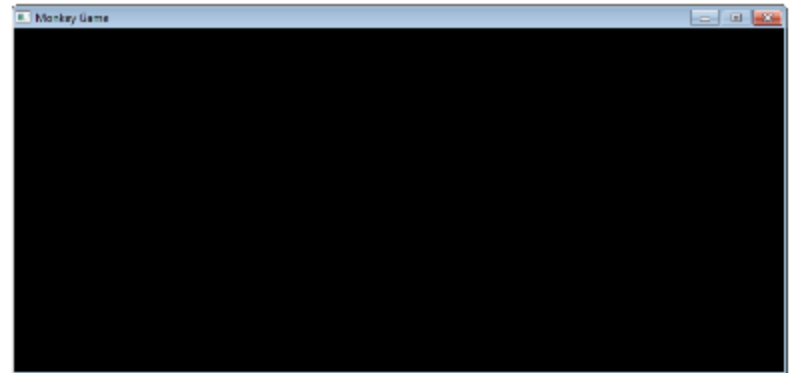
Case "Initialise"
    ball.speed = 5
    target.x=370
    target.y=125
    ball.x=375
    ball.y=295
    GameState = "Playing_s"
Case "Playing_gk"
    If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
    If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
    Turn = Turn + 1
    If KeyHit (KEY_N) Then shooting = True
    If Turn > 4 Then Next_turn = False
    If Next_turn = False Then GameState= "End"
    If shooting = True Then GameState= "Playing_s"
Case "Playing_s"

    'moving between shooter and keeper
    If KeyHit (KEY_N) Then shooting = False
    If shooting = False Then GameState= "Playing_gk"

    If KeyHit (KEY_A) Then target.Move_x(-10)
    If KeyHit (KEY_D) Then target.Move_x(10)
    If KeyHit (KEY_S) Then target.Move_y(10)
    If KeyHit (KEY_W) Then target.Move_y(-10)

    If KeyHit (KEY_SPACE)
        While target.x<>ball.x Or target.y<>ball.y
            If ball.x < target.x Then ball.x +=ball.speed
            If ball.x > target.x Then ball.x -=ball.speed
            If ball.y < target.y Then ball.y +=ball.speed
            If ball.y > target.y Then ball.y -=ball.speed
        End
    End
End

```



What is being tested?	Input	Justification of input	Outcome	How to solve
The limit to the number of turns within the game	Key n	This key allows manual turn changing	The screen turned black after n was pressed once	Read through algorithms produced in the design and try to find a solution

After multiple attempts to make the game end after 5 turns I decided to look through the other algorithms I made prior to the development, the scoring algorithm also has a way to limit the number of goes integrated into the algorithm. As a result of this I decided to move on to scoring and solve the number of turns within that algorithm.

During an evaluation meeting with my teacher I showed Mr Sargent the issue I was having with the turns switching and how after I pressed n once the screen changed black. My teacher noticed that the logic within the code was

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

correct so in theory the code should perform as expected, however I had forgot a main mechanics of programming where the OnUpdate method refreshes 60 times a second in my game as I set the update rate to 60.

```
'Game setup.....
Case "Initialise"
    'turns
    turn = 5
    'Ball setup
    ball.speed = 5
    ball.x=375
    ball.y=295
    'Target setup
    target.x=370
    target.y=125
    'Scoring setup
    CPU_Score=0
    User_Score=0

    'Start the game
    GameState = "Playing_s"

'player is goalkeeper.....
Case "Playing_gk"
    'moving goalkeeper
    If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
    If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
    'moving between keeper and shooter
    If KeyHit (KEY_N) Then shooting = True
    If shooting = True Then GameState= "Playing_s"
    'turns
    If KeyHit(KEY_N) Then turn = turn - 1
    If turn = 0 Then GameState = "End"
    'scoring
    If Saved = True Then Cpu_shot=False

    CPU_Score = CPU_Score
    If Saved = False Then Cpu_shot=True

    CPU_Score = CPU_Score + 1
```

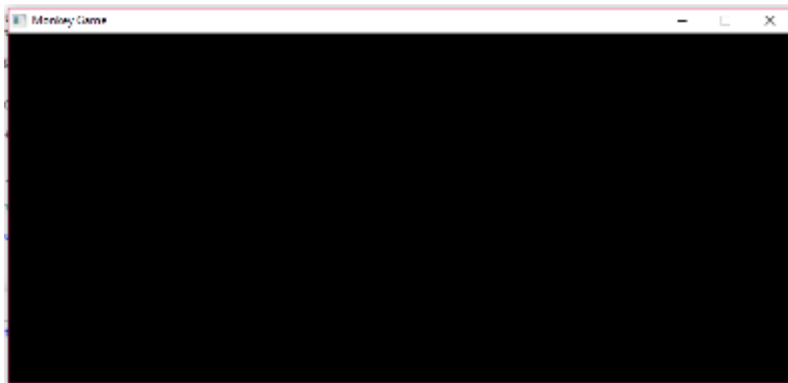
This is the working code which allows me to limit the number of turns taken, after noticing the turns where incrementing based on the number of times the OnUpdate method refreshed. To get around this issue I decided to increment the number of turns based on when "n" is pressed which allows the turn to switch. In the end, I set the variable turn equals 5 and then every time "n" is pressed while the user is playing as the goalkeeper it decrements by 1, and when turns is zero then the gamestate changes it "end" which is currently and black screen.

What is being tested?	Input	Justification of input	Outcome	How to solve
The screen turns black when the correct number of turns have been taken	Key n	This key allows manual turn changing	The turns change so the user can take 5 shot and attempt to save 5 shots before the screen is cleared	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

The picture on the top shows the screen after “n” has been pressed once and the bottom one shows the result of pressing “n” the correct number of times to end the game.



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

04/03/2017 TESTING TURNS

Test data/Input	Type	Outcome
Press n	Valid	Turn changes
Press P	Invalid	No action
Press W	Borderline	If user is shooting the crosshairs move up
Press left arrow	Borderline	If user is goalkeeper the goalkeeper moves left
Press n until five turns of shooting and goalkeeper have passed	Valid	Screens goes black

For this interview, I will be discussing improvements and positives within the prototype for turns in my game with the client Billy Evans.

Requirement	Justification
The time for each player to have a turn shooting should be around fifteen seconds as long as they haven't paused the game	The game shouldn't have a slow atmosphere as it will appeal less to my target market, they would prefer a fast paced lively game.

Me: After reevaluating this requirement it seemed unnecessary as the timer is complex to program and would require excessive time to perfect whereas as changing turns once shot has been taken seemed much more reasonable and would appeal more to the users. During the implementation of the changing of turns I hit a wall where I was unable to crack the code for automatic turn switching therefore I used manual turn switching instead as I had run out of time for the development of this prototype. On the contrary I was able to introduce the feature of limited turns within the game which makes the next prototype of scoring easier to complete, as it will be easier to total the scores.

Billy: I am disappointed about the manual turn switching, however it is better than a game that doesn't allow for proper turns and it is understandable why it may be more effective to make it manual instead of automatic.

After discussion we altered the requirement so that it is more realistic for my programming ability.

Requirement	Justification
The turns will be changed from shooter to goalkeeper etc. every time 'n' is pressed	This allows for ease of changing the turn for the user while playing the game
Each player will have a maximum of five shots at goal and five attempts to be the goalkeeper	This is a suitable number of turns that was agreed within the interview prior to the development of the game

SCORING

06/03/2017 SCORING VARIABLES

```
Global GameState:String = "Menu"  
Global tag:Bool = False  
Global shooting = True  
Global CPU_Score:Int  
Global User_Score:Int  
Global Total_Score:Int  
Global shot_taken:Bool  
Global Shot_taken:Bool  
Global Shooter:Bool  
Global User_shot:Bool  
Global Cpu_shot:Bool |
```

Firstly I created global variables to be used within the scoring algorithm, there are two data types I am using for the scoring and that is integer and Boolean. The integer's variables hold the scores of the CPU and the user, the "Total_Score" variable will hold the number of turns that have passed, incrementing by 1 each time the player is the goalkeeper. The variables that are Boolean will determine whose turn it is and whether the goal was scored or missed.

Shot_taken, is a variable that will start the scoring code if true if not then the game will continue to wait for a shot to be taken.

Shooter, this one is used to tell which player needs to have the score updated, if shooter is true then the player has taken the shot at goal and if false then the CPU score needs to be updated.

User_shot and CPU_shot allow the program to tell whether the goal has been scored or not, if either of these variables are false then shot was missed and the score remains unchanged whereas if true is returned then the goal was scored and the score needed to be incremented.

```

Method OnRender()
    Select GameState
        Case "Menu"
            DrawImage(menubackground, 0, 0)
        Case "Customise_tag"
            Cls 225,0,0
            If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
            If Not tag = False Then GameState = "Playing_ok"
        Case "Instructions"
            Cls 0,225,0
        Case "Exit"

        Case "Playing_ok"
            DrawImage(gamebackground, 0, 0)
            DrawImage server.sprite, server.x, server.y
            DrawImage (scoreboard,350,0)
            If Cpu_shot = True Then DrawImage (reddot,477,16)
            If Cpu_shot = False Then DrawImage (greendot,478,16)

        Case "Playing_n"
            DrawImage (gamebackground, 0,0)
            DrawImage target.sprite, target.x, target.y
            DrawImage ball.sprite, ball.x, ball.y
            DrawImage (scoreboard, 350,0)
            If Cpu_shot = True Then DrawImage (reddot,477,16)
            If Cpu_shot = False Then DrawImage (greendot,478,16)

        Case "End"
            Cls 0,0,0
    End
    
```

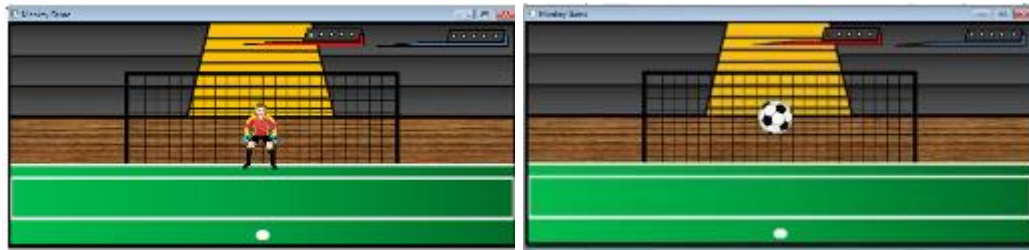
In order to make the scoreboard I had intended to design in my game I need to create a variable to hold the different coloured dots to represent a goal scored or missed. The code highlighted is for the scoreboard of the CPU, if the goal is missed then CPU_shot is false and that results in the red dot being rendered, however if CPU_shot is true then the dot rendered is green. The code in the two different playing gamestates for scoring is identical this makes sure that the scoreboard still loads the correct dots for the different areas of the game.

What is being tested?	Input	Justification of input	Outcome	How to solve
When CPU_shot is false the red dot is rendered	Key n	As the variable CPU_shot has already been set to FALSE the game only needs the shot to be taken	Once the shot is taken the scoreboard renders a red dot	NA
When CPU_shot is true the green dot is rendered	Key n	As the variable CPU_shot has already been set to TRUE the game only needs the shot to be taken	Once the shot is taken the scoreboard renders a green dot	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

This shows the results from the tests where the scoreboard has a red and green dot rendered.



After using the variables such as CPU_shot and User_shot I noticed I will need to get scoring to function automatically to find a range of outcomes to test whether a red dot is loaded or a green dot is loaded correctly based on the outcome of the shot taken. To do this I need set the CPU_shot and User_shot to be either TRUE or FALSE and to make that happen automatically I will use collisions, so the next stage of my development will be working on and test collision.

```

Method OnRender()
Select GameState
Case "Menu"
DrawImage (menubackground, 0, 0)
Case "Customize_tag"
Cls 225,0,0
If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
If Not tag = False Then GameState = "Playing_gk"
Case "Instructions"
Cls 0,225,0
Case "Exit"

Case "Playing_gk"
DrawImage (gamebackground, 0, 0)
DrawImage saver.sprite, saver.x, saver.y
DrawImage ball.sprite, ball.x, ball.y
DrawImage (scoreboard,350,0)
DrawImage (whitedot1, 477,16)
If Cpu_shot = False Then DrawImage (reddot,477,16)
If Cpu_shot = True Then DrawImage (greendot,478,16)

Case "Playing_s"
DrawImage (gamebackground, 0,0)
DrawImage saver.sprite, saver.x, saver.y
DrawImage target.sprite, target.x, target.y
DrawImage ball.sprite, ball.x, ball.y
DrawImage (scoreboard, 350,0)
DrawImage (whitedot1, 477,16)
If Cpu_shot = False Then whitedot1 = reddot
DrawImage (reddot,477,16)
If Cpu_shot = True Then DrawImage (greendot,478,16)
Case "End"
Cls 0,0,0

```

The highlighted code here shows that I have drawn in a new ball in for when the user is the goalkeeper and a goalkeeper in for when the user is the shooter

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the goalkeeper load when the user is taking the shot	Key 1	This keys is needed to load the "playing_s" gamestate	The goalkeeper has loaded	NA
Does the ball load when the user is the goalkeeper	Key 1 and Key n	This keys are needed to load the "playing_gk" gamestate	The ball has loaded	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]



This shows the test results as the goalkeeper and ball sprites have both loaded in correctly



Now I have the CPU sprites loaded into the game I need to give them movement which I will do by using a random number generator. A random number will be generated within the ranges of 1 and 3, if the number is 1 then the goalkeeper moves left, if 2 the goalkeeper remains within the centre of the goal and finally if 3 is selected then the goalkeeper moves right. This code will run once the user presses spacebar so the goalkeeper will move while the shot is being taken.

```
'moving between shooter and keeper
  If KeyHit (KEY_N) Then shooting = False
  If shooting = False Then GameState= "Playing_gk"
'moving target
  If KeyHit (KEY_A) Then target.Move_x(-10)
  If KeyHit (KEY_D) Then target.Move_x(10)
  If KeyHit (KEY_S) Then target.Move_y(10)
  If KeyHit (KEY_W) Then target.Move_y(-10)
'moving ball to target
  If KeyHit (KEY_SPACE)
    'CPU keeper movement
    Local Seed
    GK_move = Rnd(1,3)
    Select GK_move
      Case GK_move = 1
        saver.Move_x(-100)
      Case GK_move = 2
        saver.Move_x(100)
      Case GK_move = 3
        saver.Move_x(0)
    End
```

As shown within the highlighted region I have created a local seed which should mean that a different number is randomly generated each time the code runs though the "Playing_s" gamestate, then I made the actual generator where a random number should be picked between 1 and 3. Once the number has been selected I stored the random number within the variable "GK_move", then I used a case statement where the direction can be determined based on the outcome of the number.

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the goalkeeper move when spacebar is pressed	Key space	As the space bar is used to take the shot I have used the space bar to also produce the random direction for the goalkeeper as this means the goalkeeper moves as the shot is taken	The goalkeeper does move when spacebar is hit however, the goalkeeper only moves once the ball hits the goal	In order to solve this problem, I will place the code for the goalkeeper to move before the code that allows the ball to move towards the target
Direction of the goalkeeper over multiple shots	Key space	This means the goalkeeper will move simultaneously	The goalkeeper only moves to the left whenever	I believe that the issue here is that the number being generated is

Candidate Name: [REDACTED]	Candidate Number: [REDACTED]			
		with the ball so its direction cannot be predicted	spacebar is pressed	constantly the same, therefore I will need to fix the seed function so new number can be selected

This shows the screen after the spacebar is pressed where the goalkeeper only moves left. If I have time after the initial development I will increase the number of outcomes where the goalkeeper can move, whether it is all the way left or half left as show in the picture.



Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

20/03/2017 COLLISIONS BETWEEN THE GOALKEEPER AND THE BALL

To making scoring possible within my game I will have to setup collisions between the football and the goalkeeper so if the ball hits goalkeeper the shot has been saved and the score stays the same and a red dot is loaded into the scoreboard, whereas if there is no collision then the goal has been scored so the score is incremented and a green dot is rendered into the scoreboard.

I was unsure about to even start the code to setup the collision but I did know that I would have to put a box or a tile over the goalkeeper and the ball so either the two tiles collide then a TRUE value would be returned. In order to tackle this problem I reached out someone that has expertise in Monkey X called David Hillyard. David showed me the code of how to check for a collision between the tiles.

Dear Mr hillyard

I am having a problem setting up a collision within monkey x for my coursework,

I am trying to make it so that when my football sprite collides with my goalkeeper it produces a true or false value, as this will allow me to know whether or not the goal has been scored.

I think I will have to set up a tile collision where there is a tile over my goalkeeper but I'm not certain on how to make that possible.

Please could you advise my on how to go about this problem, thank you for your time.

Yours faithfully

Aaron Walkley



David Hillyard <david.hillyard@gmail.com>

Thu 23/03, 21:03

99AWA0604



Action Items

Aaron,

It sounds like you need to use bounding box collision detection. Just check if the coordinates of one sprite overlap another.

x and y are top left co-ordinates, w is width and h is height in pixels.

```
Function intersects:Bool (x1:Int, y1:Int, w1:Int, h1:Int, x2:Int, y2:Int, w2:Int, h2:Int)
```

```
If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False
```

```
If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False
```

```
Return True
```

```
End
```



```

Function intersects:Bool (x1:Int, y1:Int, w1:Int, h1:Int, x2:Int, y2:Int, w2:Int, h2:Int)
If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False
If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False
Return True
End

```

This function checks the position of the two tiles and whether they have overlapped, if they have then the function returns the value TRUE and if they have not collided then FALSE is returned.

```

'Method needed for tile.....
Method tile_hit:Int (x:Int,y:Int)
'collision
Local left_tile:Int=x /32
Local right_tile:Int=(x+31)/32
Local top_tile:Int=y/32
Local bottom_tile:Int=(y+31)/32

If left_tile < 0 Then left_tile = 0
If right_tile > 19 Then right_tile = 19
If top_tile < 0 Then top_tile = 0
If bottom_tile > 14 Then bottom_tile = 14

Local collision_result:Int = 0
For Local i:Int = left_tile To right_tile
    For Local j:Int = top_tile To bottom_tile
        If tiles[i][j] = wall Then collision_result = wall
    Next
Next
Return collision_result
End

```

```

'checks collision
If intersects(target.x,target.y,32,32,ball.x,ball.y,32,32) Then GameState = "End"

```

This IF statement is within the OnUpdate method and calls the function of “intersects” which checks if a collision has occurred. The highlighted code shows how I will test whether the collision is returning then correct Boolean value, where if TRUE is returned from the “intersects” function then the game will render a black screen.

What is being tested?	Input	Justification of input	Outcome	How to solve
-----------------------	-------	------------------------	---------	--------------

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Does the GameState change when the ball hits the goalkeeper	A shot at the goalkeeper	A shot needs to be taken at the goal	The screen turns black where the GameState has been changed	NA
Does the GameState change when the ball does not hit the goalkeeper	A shot that does not come into contact with the goalkeeper	To test the collision doesn't occur when the ball misses the goalkeeper	The screen turns black where the GameState has been changed	I believe the issue with this is that the tiles around the goalkeeper and ball are too large which results in the tiles colliding every time a shot is taken

I have made progress for the collisions within my game, as I was going to remind myself where I had got I to I noticed that I had placed a tile around the wrong sprite as shown below.

```
'checks collision
If intersects(target.x,target.y,32,32,ball.x,ball.y,32,32) Then GameState = "End"
```

```
'checks collision
If intersects(saver.x,saver.y,10,10,ball.x,ball.y,10,10) Then GameState = "End"
```

As shown above I made the mistake of using the target instead of the goalkeeper, this meant every time a shot was taken the screen would turn black as a collision has been detected, hence the problem I faced with the screen turning black every time.

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the GameState change when the ball hits the goalkeeper	A shot at the goalkeeper	A shot needs to be taken at the goal	The screen turns black where the GameState has been changed	NA
Does the GameState change when the ball does not hit the goalkeeper	A shot that does not come into contact with the goalkeeper	To test the collision doesn't occur when the ball misses the goalkeeper	The screen does not turn black and the game continues as planned	NA

However, I have found another problem with the collisions.

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the gamestate change when the ball hits the goalkeeper after it has moved	A shot at goal when the goalkeeper has moved either left or right	This will allow me to test the collision where the goalkeeper is moved, this will be the most common type of collision as the user will want to move the goalkeeper and the CPU goalkeeper is likely to move with its random direction	The screen does not turn black and there is no change in gamestate	I am going to go over the whole collision function and read through the Craigndave barebone tutorials and see if they were able to implement collisions with a different approach

After struggling with finding a way to solve this I spoke to Dan Needham another student developing a game which involves collisions, after discussing how with Dan, he helped me to implement the collision I needed.

Firstly, outside any method I created a global variable where the result of the collision are stored, this variable will be used when determining the outcome of the shot.

```
'Collision  
Global collision_test:bool
```

This line of code checks for an overlap of the saver sprite and the ball sprite and returns the Boolean value of true if there is an overlap, within this line the function called intersect is called.

```
'checks collision  
If intersects(saver.x,saver.y,65,113,ball.x,ball.y,55,55) Then collision_test = true
```

This function takes the dimensions set in the code above and if those coordinates overlap then true is returned.

```
Function intersects:Bool (x1:Int, y1:Int, w1:Int, h1:Int, x2:Int, y2:Int, w2:Int, h2:Int)  
If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False  
If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False  
Return True  
End
```

This piece of code allows me to test whether the collisions are working or not as when the collision is happening true is returned and the text "collision" appears on the screen.

```
'collision  
If collision_test = True Then  
    DrawText("collision", 10, 10)
```

What is being tested?	Input	Justification of input	Outcome	How to solve
Does the collision text appear when the ball hits the goalkeeper once it has moved	Ball is shot at goal and hits the goalkeeper	This tests the collisions between the goalkeeper and the ball	The text appears	NA
Does a collision happen when the ball does not hit the goalkeeper	Ball is shot at the goal and does not hit the goalkeeper	This tests whether the collisions are functioning properly	The text does not appear	NA

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]



This shows the collision working as in the top left of the screen the text is shown which means the collision is true



This shows that there is no collision after the shot has been taken at goal as there is no text on the screen

This is the interview held with Billy Evans where we evaluate the progress in the development of the scoring prototype.

Requirement	Justification
When a goal is scored the relevant white circle will turn green and if the goal is missed or saved the circle will become red	This is the method I plan to use for the score board as it removes the number system of points and uses a more appealing visual score board which will be more enticing for my audience.
When the game ends the end screen is displayed with cheering and 'PLAYER X WINS' if they win, and booing sound effects with 'BETTER LUCK NEXT TIME' if they lose. Finally, if the player draws with the CPU the banning 'IT'S A DRAW' appears and the backing track which will have been playing throughout the game will continue to play.	This shows achievement for the player once they win, and they are likely to play again to get feeling of achievement once again, on the contrary if they lose the phrase 'BETTER LUCK NEXT TIME' leads them to replay the game in order to attempt the win.
There will be a maximum score limit of five	This score limit is one that Billy chose and it is a very reasonable one as there are not so many that the game drags on and not so little that the game ends too soon.

Me: I found these requirements to be effective and suitable for my game, however time was my greatest issue upon the development of this prototype, as a result of the lack of time available during this module's implementation I was unable to complete the scoring prototype.

I had started to make progress within the requirement where the different coloured dots would appear to symbolise the outcome of the shot. For the other requirement, I was not able to start to program the end screen feature as the deadline had passed before I had reached its development. However, I was able to complete the maximum score of 5 by implementing a set number of turns within the turns development as a result of this the user cannot exceed the maximum number of turns.

As the module was not completed we were unable to discuss the outcome of the scoring prototype for the penalty shootout game.

ANNOTATIONS 31/03/2017

During my development, I enjoyed writing the code and when I did I would go off on a tangent and produce code which was functional however, I was not spending the time inserting annotations within the program, this meant if I wrote a long section of code at night and went back to it the next day I would find myself lost within my own code. Although I understood the majority of my code it was an inconvenience as I would have to spend time reading through the code and rendering the game to find where I was up to. For an algorithm to be effective it requires suitable annotation so that other users if they were to take over the coding would know how far the development has come and it also helps with the process of code maintainence.

To show I am able to write effective code I have now annotated my code to make the algorithms understandable for anyone else, the sections of code below are taken from the start of the code where I have attributes used within the game and below that a piece from the OnRender method.

```
'importing libraries
import mojo
import brl
'creating a variable for the game itself
Global Game:PenaltyGame

'class for the game*****
Class PenaltyGame Extends App

'Attributes
    'Declare images within game
    Field menubackground:Image
    Field gamebackground:Image
    Field scoreboard:Image
    Field reddot:Image
    Field greendot:Image
    Field whitedot1:Image
    'Declare sprites
    Field saver:goalkeeper
    Field target:shooter
    Field ball:football
```

```

'Renders graphics-----
'This code is responsible for loading images to the screen
Method OnRender()
    'gamestate selection
    Select GameState

        'Draw menu
        Case "Menu"
            DrawImage (menubackground, 0, 0)

        'Draw customize tag screen
        Case "Customize_tag"
            Cls 225,0,0
            If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
            If Not tag = False Then GameState = "Playing_gk"

        'Draw instruction menu
        Case "Instructions"
            Cls 0,225,0

        'Draw Goalkeeper images and sprites
        Case "Playing_gk"
            'Background
            DrawImage (gamebackground, 0, 0)
            'Goalkeeper
            DrawImage saver.sprite, saver.x, saver.y
            'Ball
            DrawImage ball.sprite, ball.x, ball.y
            'Scoring
            DrawImage (scoreboard,350,0)
            'drawing white, green and red dots to the screen
            DrawImage (whitedot1, 477,16)
            If Cpu_shot = False Then DrawImage (reddot,477,16)
            If Cpu_shot = True Then DrawImage (greendot,478,16)

        'Draw Shooting images and sprites
        Case "Playing_s"
            'Background
            DrawImage (gamebackground, 0,0)
            'Goalkeeper
            DrawImage saver.sprite, saver.x, saver.y
            'Target
            DrawImage target.sprite, target.x, target.y
            'Ball
            DrawImage ball.sprite, ball.x, ball.y
            'Scoring (testing scoring idea)
            DrawImage (scoreboard, 350,0)
            DrawImage (whitedot1, 477,16)
            If Cpu_shot = False Then whitedot1 = reddot
            DrawImage (reddot,477,16)
            If Cpu_shot = True Then DrawImage (greendot,478,16)

            If collision_test = True Then
                DrawText("collision!!!", 10, 10)
            End

        'Draw end screens
        Case "End"
            Cls 0,0,0

```


After looking through my development I realised that I had explained the purpose of all the attributes but I had left out the variables within the game, I will now explain the role of each variable which I have used, as in the real world if someone was to take over your code then you need them to be aware of the functionality of the variables.

Modules	Name of variable	Justification	Data type	Justification
Gamestate	Gamestate	This allows the gamestate to be easily changed so different areas of code can be access for different areas of the game	String	The gamestate are named after sections of the game
Custom tag	-	-	-	-
Turns	Shooting	I used this variable as a way to determine what turn the user should have and also as a mechanic in order to change the turn	Boolean	If shooting is true then the gamestate is changed to or remains at the shooting code, if it is false then the game will run the goalkeeper code
Turns	Turn	This is used within the capped number of turns feature where it acts a countdown in effect	Integer	The variable is initially set at a specific number then each time the goalkeeper turn passes the variable is decremented by one until the value of Turn equals zero
Turns	Next_turn	This variable was not needed in the end result	Boolean	-
Turns	addturn	This variable was not needed in the end result	Boolean	-
Scoring	CPU_Score	This variable should have held the current CPU score during the game	Integer	If the CPU scored then the value of CPU_Score would be incremented by one, as a result of this functionality the datatype needed to be integer
Scoring	User_Score	This variable should have held the current user score during the game	Integer	If the user scored then the value of User_Score would be incremented by one, as a result of this functionality the datatype needed to be integer
Scoring	Shooter	This in theory acts like the variable 'shooting' where it is a way to determine which turn the user is currently in	Boolean	When 'Shooter' is true the user is taking the shot at goal and if the variable is false then the CPU is taking the shot, this can only be done with the Boolean datatype

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Scoring	User_shot	This variable is a way to determine the outcome of the shot at goal for the user	Boolean	The data type is Boolean as true and false are used to determine if the shot has been scored, if the variable returns true then the user has scored and if false then they have missed
Scoring	Cpu_shot	This variable is a way to determine the outcome of the shot at goal for the CPU	Boolean	The data type is Boolean as true and false are used to determine if the shot has been scored, if the variable returns true then the CPU has scored and if false then it has missed
Scoring	Saved	The variable here is associated with the collision between the ball and the goalkeeper, this means it will say whether the goal has been scored or not for either the user or the CPU	Boolean	If the collision returns true then the goal has been saved and if the collision returns false then the goal has been scored, this function can once again only be performed using a Boolean data type
CPU movement	Seed	I tried to generate a seed so new random numbers would be selected each time the CPU goalkeeper needed to move, however this attempt did not work	-	-
CPU movement	GK_move	This variable would have allowed use of a case statement where the direction of the goalkeeper can be determined based on the outcome of the number	Integer	This variable uses an integer data type as it is used within a Select Case where a random number is generated and then number is set to GK_move
Collision	collision_test	This variable hold the outcome of the collision	Boolean	Boolean is needed for this variable as if there is a collision then collision_test comes back as true and if there is no collision then it comes back as false

D. EVALUATION**TESTING****MOVEMENT**

These tests are used to test the movement within the game and will run valid, invalid and extreme data against the algorithm.

Test number	What is being tested?	Input/action	Expected outcome	Actual outcome	Has requirement been reach?	Video reference
1	Goalkeepers movements	VALID: Right arrow pressed twice, while player is goalkeeper	VALID: Goalkeeper moves two places to the right	VALID: The goalkeeper moves right twice	YES	1.1
		INVALID: 'D' is pressed twice, while player is goalkeeper	INVALID: The goalkeeper won't move	INVALID: The goalkeeper doesn't move	YES	1.2
		Move the goalkeeper all the way to its boundaries	The goalkeeper should stay within the goal	The goalkeeper stays within the goal	YES	1.3
2	Goalkeepers movements	VALID: Left arrow pressed twice, while player is goalkeeper	VALID: Goalkeeper moves two places to the left	VALID: Goalkeeper moves two places to the left	YES	2.1
		INVALID: 'A' is pressed twice, while player is goalkeeper	INVALID: The goalkeeper won't move	INVALID: The goalkeeper doesn't move	YES	2.2

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

3	Goalkeepers movements	VALID: Spacebar is pressed once, while player is goalkeeper	VALID: Goalkeeper jumps up to reach the ball for a chosen time	I did not implement this feature	NO	-
		INVALID: 'W' is pressed twice, while player is goalkeeper	INVALID: The goalkeeper won't move	The goalkeeper does not move	YES	3.1
4	Applying power to the ball before taking a shot	VALID: While player is the shooter press spacebar to stop the purple slide from moving and to select the power	VALID: The purple pointer stops on a colour that colour is then placed in the box below the slide, that colour later determines the power	I did not implement this feature	NO	-
		INVALID: While player is the shooter and selecting press ENTER	INVALID: No affect from pressing enter	INVALID: No change occurs from pressing enter	YES	4.1
5	Moving the crosshairs for the shot	VALID: While the cross hairs are on the screen, A is pressed once	VALID: The crosshairs move left one place	VALID: The crosshairs move left one place	YES	5.1
		INVALID: While the cross hairs	INVALID: The crosshairs won't move	INVALID: The crosshairs did not move	YES	5.2

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

		are on the screen, 5 is pressed once				
		BORDERLINE: While the cross hairs are on the screen, 'left arrow' is pressed once	BORDERLINE: The crosshairs won't move	BORDERLINE: The crosshairs did not move	YES	5.3
		Move the target to its boundaries	The target should stay within the goal	The target stays within the goal	YES	5.4
6	Moving the crosshairs for the shot	VALID: While the cross hairs are on the screen, W is pressed once	VALID: The crosshairs move upwards one place	VALID: The crosshairs moved upwards one place	YES	6.1
		INVALID: While the cross hairs are on the screen, 'up arrow' is pressed once	INVALID: The crosshairs won't move	INVALID: The crosshairs would not move	YES	6.2
7	Moving the crosshairs for the shot	VALID: While the cross hairs are on the screen, S is pressed once	VALID: The crosshairs move downwards one place	VALID: The crosshairs moved downwards one place	YES	7.1
		INVALID: While the cross hairs are on the screen, 'down arrow'	INVALID: The crosshairs won't move	INVALID: The crosshairs didn't move	YES	7.2

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

		is pressed once				
8	Moving the crosshairs for the shot	VALID: While the cross hairs are on the screen, D is pressed once	VALID: The crosshairs move right one place	VALID: The crosshairs moved right one place	YES	8.1
		INVALID: While the cross hairs are on the screen, 9 is pressed once	INVALID: The crosshairs won't move	INVALID: The crosshairs did not move	YES	8.2
		BORDERLINE: While the cross hairs are on the screen, 'right arrow' is pressed once	BORDERLINE: The crosshairs won't move	BORDERLINE: The crosshairs did not move	YES	8.3
9	Movement from the penalty to the target	VALID: The space bar is pressed	VALID: The ball moves towards and ends up on the target	VALID: The ball jumps directly to and ends on the target	NO	9.1
		BORDERLINE: When the user is the goalkeeper, space bar is pressed	BORDERLINE: There will be no movement of the ball	BORDERLINE: There was no movement of the ball	YES	9.2

Test number	What is being tested?	Input/action	Expected outcome	Actual outcome	Has requirement been reached?	Video reference
10	Once the goalkeeper finishes there go they will then be switched to a shooter	VALID: The goal has been saved/missed	VALID: The goalkeeper then becomes the shooter	VALID: CPU cannot shoot so this is unable to be tested	NO	-
		BORDERLINE: press 'P'	BORDERLINE: The game will pause	The pause feature has not been implemented	NO	-
11	Once the shooter has taken there shot and the score has been updated the player will then become the goalkeeper	VALID: The shot has been taken and the score has been updated	VALID: The shooter become goalkeeper	VALID: The user stays in the same turn	NO	11.1
		BORDERLINE: press 'P'	BORDERLINE: The game will pause	The pause feature has not been implemented	NO	-
12	The player changes between goalkeeper and shooter	VALID: The turn is manually changed when N is pressed	VALID: The user will change turns and will be able to control the	VALID: While playing as the goalkeeper the user can move the goalkeeper only and when	YES	12.1

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

			goalkeeper on one turns and control the target and be able to shoot the ball in the other	they are the shooter the target can be moved and the shot can be taken at goal		
		INVALID: The keys W,A,S,D, left arrow and right arrow are pressed	INVALID: The cross hairs or the goalkeeper will move depending on what turn the user is in	INVALID: While the user is shooting the crosshairs move and when the user is saving the shot the goalkeeper moves	YES	12.2
13	There is a maximum number of five turns of each turn	VALID: N is pressed ten time	VALID: A black screen is loaded when the maximum number of turns has been taken	VALID: The screen turns black on the tenth-time n is pressed	YES	13.1
		BORDERLINE: N is pressed eleven times	BORDERLINE: The screen will remain black	BORDERLINE: The screen remains black	YES	13.2
		INVALID: W is pressed eleven times	INVALID: The crosshairs move upwards eleven times	INVALID: The crosshairs move upwards	YES	13.3

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

CUSTOM TAG AND SOUND EFFECTS

These features were not implemented within my game as I had run out of time for the development of my game.

NAVIGATION

Test number	What is being tested?	Input/action	Expected outcome	Actual outcome	Has the requirement been reached?	Video reference
14	The pause feature	VALID: During the game the key 'P' will be pressed	VALID: The game will load the paused game state	The pause feature was not implemented	NO	-
		INVALID: During the game press the 'O' key	INVALID: The game will continue playing	The pause feature was not implemented	NO	-
15	The navigation around menus	VALID: The left mouse click will be pressed on all the buttons within the game (includes all menus, such as the pause menu, main menu, return and rematch buttons)	VALID: The button that is selection will proceed to the correct screen	VALID: No effect occurs	NO	15.1
		INVALID: The left mouse click is used were there are no buttons	INVALID: There would not be any change within in the game as a result of the click	INVALID: There was no change within the game	YES	15.2

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

		VALID: The use of keys to navigate the menu where 1,2 and 3 are pressed while the menu is loaded	VALID: The game loads the correct areas of the game where the game loads if one is pressed, a green screen is loaded if 2 is pressed and finally the game closes if 3 is pressed	VALID: The game loads the correct areas of the game where the game loads when one is pressed, a green screen is loaded when 2 is pressed and finally the game closes when 3 is pressed	YES	15.3.1 15.3.2 15.3.3
		INVALID: At the menu, the keys that control the crosshairs movement are pressed	INVALID: There is no change within the game	INVALID: There is no change in the game	YES	15.4

Test number	What is being tested?	Input/ actions taken	Expected outcome	Actual outcome	Has the requirement been reach?	Video reference
16	Transition from game state of menu to the playing game state	VALID: The 'play' option is selected at the main menu	VALID: The game state will change to playing and the game will initiate	VALID: The game initializes loads	YES	16.1
		INVALID: At the main menu press 'ENTER'	INVALID: No change should occur to the game state	INVALID: There is no change to the game	YES	16.2
17	Transition from game state of menu to the instruction game state	VALID: The 'instruction' button is selected at the main menu	VALID: The game state will change to instruction and the screen will turn green	VALID: A green screen is loaded	YES	17.1
		BORDERLINE: The key 1 is pressed at the main menu	BORDERLINE: The game state will change to playing and the game will initiate	BORDERLINE: The game initializes loads	YES	17.2
18	Closing the game down	VALID: The 'exit' button is selected at the main menu	VALID: The game will close	VALID: The game closes	YES	18.1
		INVALID: The key ESC is pressed	INVALID: There is no change is the game	INVALID: There is no change within the game	YES	18.2

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

19	The transition to the game state paused	VALID: 'P' is pressed while the game is playing	VALID: The game will pause	The pause feature was no implemented	NO	-
20	Transition to 'over' game state	VALID: Complete the game	VALID: The game state will change from playing to over and a black screen will render	VALID: A black screen loads	YES	20.1
		INVALID: Press 'P' and then click 'exit'	INVALID: The game returns to the menu game state	This feature was not implemented	No	-

SCORING

The tests within scoring are not able to be completed as I have not completed the scoring module, I am able to test the collisions within the game however I have not got as far as to refresh the collisions and sprites after each turn as a result if the collision hits once then it will stay there each time the user becomes the shooter.

Test number	What is being tested?	Input/ action taken	Expected outcome	Actual outcome	Has the requirement be reached?	Video reference
21	Collisions between the goalkeeper and the ball	VALID: A shot is taken at the goalkeeper while it has not been moved from its starting position	VALID: The text collision should appear in the top left of the screen	VALID: The string collision appeared	YES	21.1
		INVALID: Move the goalkeeper to the edge of the goal and take a shot at where the goalkeeper spawns	INVALID: The collision text should not render	INVALID: The collision text does not load	YES	21.2
		VALID: Move the goalkeeper from the starting position and shoot the ball at the it	VALID: The collision text should render	VALID: The collision text appears	YES	21.3

USABILITY FEATURES

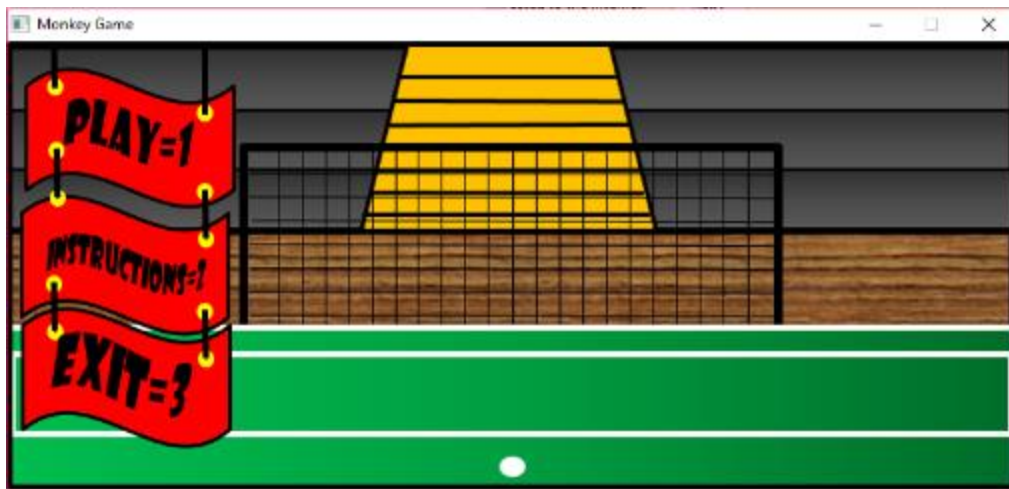
MENU

Me: On the menu I used the keys 1,2 and 3 to select the options play, instructions or exit.

Billy: I like this as it makes the game easy to navigate as there are clear keys that need to be pressed to continue with the game.

Me: I have left the menu screen fairly simplistic as we agreed the game cannot to be too 'busy' as that may put the user off the game.

Billy: This is simple but less is more and this clearly sets a scene for the game early on which.



CONTROLS

Me: As agreed in the game requirements I have set the controls for the target to be WASD which are a well-known set of controls familiarly known for movement.

Billy: This is good as I've played games with these controls before and is a basic need when looking into game controls.

Me: For the controls of the goalkeeper I have used the left and right arrows which we agreed prior to the development.

Billy: These controls are more suitable for the goalkeeper as the WASD controls are used more with full range of movement whereas the goalkeeper can only go left or right.

Me: Although I was not able to implement the power selection for as a feature within the game I used the space bar as the control for shooting which was still in the requirements.

Billy: As mentioned before I would have liked the power selections to be implemented as it would give the game an advantage over other football games but it isn't a great issue.

Me: For the manual turn switching which is a new requirement I have set 'n' to be the control for this and it isn't a key you can accidentally hit which using the controls which would potentially ruin the game, and I thought the n could stand for 'next turn'.

Billy: It is a good idea to use a key that cannot easily be pressed by mistake as if the turns changed before the shot was taken then the turn would count as a miss and jeopardise the scores within the game.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

SCOREBOARD

Me: Although the positioning and size of the scoreboard was not discussed I believe it to be important decision as if too big then it could take up too much of the screen and if too small then the scores could be difficult to read. I have positioned the scoreboard in the top right of the screen as it not blocking anything important there.

Billy: This is a good place for the scoreboard as it is easy to check and is not in your face.

Me: I have started but not finished the coloured dot feature, the dots allow for the user to easily see whether or not the goal has been scored, in addition the scoreboard shows five dots for each player so they know how many shots remain.

Billy: I like this feature as it's more than a simple number and it makes the game more interesting and will appeal more to the audience.



EVALUATION

Requirement	Justification
Game has a 1 st person view from the person taking the penalty.	This gives the game a more realistic effect and I don't have the time to implement another moving character.

This requirement has been met as my game has first person view from the penalty taker, this was achieved by the screen designs being designed from the specified point of view.

Requirement	Justification
Still camera view for the screen	This makes the game easier to play as it means the shots can be more accurate.

This was achieved easily as this automatically happens within the program of Monkey X, therefore no action was taken to implement this feature into the game.

Requirement	Justification
<ul style="list-style-type: none"> • “ ”(space bar)- Applies power to the shot • ‘A’ moves the crosshairs for the shot to the left • ‘D’ moves the crosshairs for the shot to the right • ‘S’ moves the crosshairs for the shot to the down • ‘W’ moves the crosshairs for the shot to the up 	<p>The space bar is an effective way of applying power to the ball, as the longer it is held down the more power is given.</p> <p>‘W,A,S,D’ is used inside of the arrows to avoid confusion with in the game.</p>
‘Left arrow’(move left) , ‘Right arrow’(move right) and ‘space’(jump) will control the goal keeper	These are simple and clear controls which mean my audience will be able to play the game without any issues which may prevent their interests in my game.
The goal keeper moves at a speed where it takes two seconds to move from one goal post to the other	This is a reasonable speed as it's not as fast as the ball which means the person shooting has a very possible chance of scoring.
The ball takes one second to move from the penalty spot to the goal	This is so the shooter has a higher chance of scoring than the goalkeeper has with saving the ball.

After the initial implementation of the movement I discovered that the time frame for the project would be slim as there is a lot to do in not much time as a result of this realisation I have kept the movement features to a functional level.

I have followed most of the requirements including :

- Space bar to shoot the ball the target
- A, to move the crosshairs left
- D, to move the crosshairs right
- S, to move the crosshairs downwards
- W, to move the crosshairs upwards

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

-Left arrow, to move the goalkeeper left

-Right arrow, to move the goalkeeper right

I was unable to implement the power feature within the game as that was above my ability of programming in monkey X and it would have consumed too much time attempting to find a solution. In addition, I found that the only movement necessary to the goalkeeper is the left and right horizontal movement, due to this I decided not to attempt to include the jumping feature of the goalkeeper until I have a functioning game as I am worried I will not reach the deadline for the game development. The requirements for the speed of the goalkeeper and the speed of the ball were not implemented as I do not have a good enough understanding of Monkey X to implement the speeds of the sprites with the time I have for the development.

HOW TO SOLVE

Requirement	Justification
The main menu will have three options of PLAY/INSTRUCTIONS/QUIT	These are the essential options as if the user is unable to play the game there is no point of creating it, these options are a clear way of directing the user to selecting what they wish to proceed with.
Background image of a goal within a stadium	Although my game isn't realistic as other games such as fifa 16 due to the cartoon graphics and other the movement effects etc, the stadium means that I can have an element of realism with the game. This is also the background which Billy picked within my interview as well.
'Left mouse click' is used to navigate through the menu's and return buttons	This is a simple and effective way of navigating through menus of the game.

I was able to fulfil most of the requirements within this area of the development as I have designed and implemented the main menu which gives options to 'PLAY', load 'INSTRUCTIONS' or 'EXIT'. In addition, I have also used the design I created for the game while it is being played. However I decided it would be more suitable for the navigation to be performed by pressing keys due to my programming ability as I am unfamiliar with the implementation of buttons for Monkey X and I did not want to get behind on the development right at the start of it.

HOW TO SOLVE

Requirement	Justification
The time for each player to have a turn shooting should be around fifteen seconds as long as they haven't paused the game	The game shouldn't have a slow atmosphere as it will appeal less to my target market, they would prefer a fast paced lively game.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

After reevaluating this requirement it seemed unnecessary as the timer is complex to program and would require excessive time to perfect whereas as changing turns once shot has been taken seemed much more reasonable and would appeal more to the users. During the implementation of the changing of turns I hit a wall where I was unable to crack the code for automatic turn switching therefore I used manual turn switching instead as I had run out of time for the development of this prototype. On the contrary I was able to introduce the feature of limited turns within the game which makes the next prototype of scoring easier to complete, as it will be easier to total the scores.

Requirement	Justification
When a goal is scored the relevant white circle will turn green and if the goal is missed or saved the circle will become red	This is the method I plan to use for the score board as it removes the number system of points and uses a more appealing visual score board which will be more enticing for my audience.
When the game ends the end screen is displayed with cheering and 'PLAYER X WINS' if they win, and booing sound effects with 'BETTER LUCK NEXT TIME' if they lose. Finally, if the player draws with the CPU the banning 'IT'S A DRAW' appears and the backing track which will have been playing throughout the game will continue to play.	This shows achievement for the player once they win, and they are likely to play again to get feeling of achievement once again, on the contrary if they lose the phrase 'BETTER LUCK NEXT TIME' leads them to replay the game in order to attempt the win.
There will be a maximum score limit of five	This score limit is one that Billy chose and it is a very reasonable one as there are not so many that the game drags on and not so little that the game ends too soon.

I found these requirements to be effective and suitable for my game, however time was my greatest issue upon the development of this prototype, as a result of the lack of time available during this module's implementation I was unable to complete the scoring prototype.

I had started to make progress within the requirement where the different coloured dots would appear to symbolise the outcome of the shot. For the other requirement, I was not able to start to program the end screen feature as the deadline had passed before I had reached its development. However, I was able to complete the maximum score of 5 by implementing a set number of turns within the turns development as a result of this the user cannot exceed the maximum number of turns.

Requirement	Justification
The game will be a single player game where you play the CPU	I won't be able to implement the multiplayer feature due to my ability in programming within the time frame I have for the end product to be made.

As mentioned in the justification of why the game would be a single player, I was unable to implement the multiplayer feature although I believe it would have made an improvement.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

After the implementing stage of project I come to a realisation that multiplayer is easier to program than the single player, this is as the sprites won't need random motion which is difficult to implement within Monkey X.

Requirement	Justification
The instruction menu will have a diagram of the keyboard and mouse which are labelled with their functions	Billy stated in the interview that the instruction menu is an important feature as it reduces confusion, the labelled diagram is better than a list of instructions as it gives a visual aid which will appeal more to my audience.

As I was unable to finish my game by the deadline I was unable to produce an instruction menu as it was not a fundamental to the game whereas areas such as scoring and collisions are, as the instructions are pointless if the game does not function properly.

Requirement	Justification
'P' will pause the game until the game is resumed with a 'left mouse click' on the resume tab	The pause feature is a useful requirement as it allows the player to come back to the game, this means they are more likely to return for another go.

I did not implement the pause feature within the game as it was a bonus feature where if the game was functional then I would implement it, the feature was not necessary as the turns do not time out hence pausing isn't needed.

Requirement	Justification
Graphics will involve cartoon sprites	In the interview, Billy informed that he would prefer to have a cartoon game as it looks more interesting for my target market instead of a reality representation.

This requirement was met as I used sprites and based my screen designs around a cartoon theme, I think the sprites I selected for the game are fitting as they suit the games background and over style.

Requirement	Justification
There will be a suitable backing track	This backing track will reduce silence within the game which could possibly lead to the audience getting bored.

I run out of time during the development process to implement sound tracks however I know how I would create a solution to put sound effects within the game.

HOW

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

Requirement	Justification
Customizable gamer tag	This feature is strongly looked upon by Billy and is also a feature used within Fifa 16 so I've decided to use it in order to improve my game.

Although I did not fully produce this feature in the game I did consider how I would be able to make this possible and it involved creating a text box for the user to type in and I am not experienced enough with Monkey X to implement this sort of feature. If I would have spent more of my time of finding out on how to do this I would have wasted time that could have been spent on movement of the sprites which has more of a priority over the customizable tag.

Requirement	Justification
To be bright and colourful	The game needs to have an enticing vibe to it in order to persuade children between the ages of 5 and 14 to play it, I believe that this can be done by using bright colours to appeal to children.

The outcome of whether the requirement has been met or not will be down to the perspective of the user, however I feel that I have made the game look appealing to people between the ages of 5 and 14 which are my target audience for this game.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

MAINTENANCE

There are limitations within my game as I have not completed the development, however these limitations are based around the modules that I have completed. Maintenance within games overall are made to fix bugs that may be in the games which aren't meant to be there or to make improvements to the original game. This may include introducing any requirements that were not implemented in the original game development.

MAINTENANCE AND LIMITATIONS

The ball takes one second to move from the penalty spot to the goal

If DLC were to be released once the game was released I would attempt to implement this requirement as within the current game the ball moves directly to the target and that was not an original requirement set by my client, however due to the difficulty to produce this feature I was unable to implement it within the period for my development.

'Left mouse click' is used to navigate through the menu's and return buttons

For this I would need to implement buttons into the game however it would improve the game as it means the requirement made in the analysis would be completed.

Background image of a goal within a stadium

This would be maintained by creating new backgrounds over time to keep the game interesting and to keep the user playing the game, other backgrounds may include different pitches or parks based on different counties. Essentially the same can be done to the sprites so they suit the new background.

The game will be a single player game where you play the CPU

Although the requirement specifies single player the game could be improved by implementing a multiplayer feature which would have been easier to develop as this cuts out the need for random movement of the CPU goalkeeper and random shots from the CPU. However, I was unaware of the difficulty that would be presented for the single player prior to the development of the game.

When a goal is scored the relevant white circle will turn green and if the goal is missed or saved the circle will become red

This feature has not been completed for my game so for maintenance I would have this feature implemented.

When the game ends the end screen is displayed with cheering and 'PLAYER X WINS' if they win, and booing sound effects with 'BETTER LUCK NEXT TIME' if they lose. Finally, if the player draws with the CPU the banning 'IT'S A DRAW' appears and the backing track which will have been playing throughout the game will continue to play.

I have not implemented this feature as I had run out of time for the development however to implement this I would create a gamestate for each outcome, for example 'game_win, game_lose and game_draw' then based on the outcome of the scores render the fitting end screen with the correct sound effect. As mentioned earlier about releasing new backgrounds that are changed overtime, the same concept can be applied here as the end screens show the original background.

'P' will pause the game until the game is resumed with a 'left mouse click' on the resume tab

This feature was in the original requirements however it never got implemented, in an update post game release I could introduce the pause feature to make the overall game quality improve. This can be implemented by creating a gamestate called 'pause' which when it is called renders the pause screen; to know when 'pause' needs to be selected a selection statement will go in both gamestates 'playing_gk' and 'playing_s' where if 'P' is pressed the gamestate will be changed to 'pause'.

Customizable gamer tag

The limitation here was that I did not know how to implement a text box that the user could type into to produce their own custom gamer tag, this could be put into the game as an update for an improvement.

There will be a suitable backing track

This would make the game more intriguing as this way the user isn't sat in silence, this was an original requirement but again was not implemented as of the time frame for the development. Over time once this feature has been created different sound tracks and effects can be added to fit the different backgrounds and their themes from different countries.

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]

PROJECT APPENDIXES

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

- Complete Code Listing (ESSENTIAL)
- Interview Transcripts
- Meeting notes
- Observation notes or questionnaires

CODE LISTINGS

```

'importing libraries
Import mojo
Import brl
'creating a variable for the game itself
Global Game:PenaltyGame

'-----
'class for the game
Class PenaltyGame Extends App

'Attributes
'Declare images within game
Field menubackground:Image
Field gamebackground:Image
Field scoreboard:Image
Field reddot:Image
Field greendot:Image
Field whitedot1:Image
'Declare sprites
Field saver:goalkeeper
Field target:shooter
Field ball:football

'Global variables
'Game states
Global GameState:String = "Menu"
'Custom tag
Global tag:Bool = False
'Turns
Global shooting = True
Global turn:Int
Global Next_turn:Bool
Global addturn:Bool
'Scoring
Global CPU_Score:Int
Global User_Score:Int
Global shot_taken:Bool
Global Shooter:Bool
Global User_shot:Bool
Global Cpu_shot:Bool
Global Saved:Bool
'CPU movement
Global Seed
Global GK_move:Int
'Collision
Global collision_test:bool

'-----
'Game initialise
'-----
'This is code is run once as the start of the game, this holds the initialising code at the start of the game
Method OnCreate()
'frame rate
SetUpdateRate(60)
'Load backgrounds and images used
menubackground = LoadImage("menubackground.png")
gamebackground = LoadImage("gamebackground.png")
scoreboard = LoadImage ("scoreboard.png")
reddot = LoadImage ("reddot.png")
greendot = LoadImage ("greendot.png")
whitedot1 = LoadImage ("whitedot1.png")

'object initialisation
saver=New goalkeeper
target=New shooter
ball=New football

End

```

```

'Game logic-----
'This code is run 60 times a second as I have set the UpdateRate to 60 in OnCreate, this code handles the logic within the game
Method OnUpdate()
    'gamestate selection
    Select GameState

        'Main menu options.....
        Case "Menu"
            If KeyHit (KEY_1) Then GameState= "Initialise"
            If KeyHit (KEY_2) Then GameState= "Instructions"
            If KeyHit (KEY_3) Then GameState= "Exit"

        'Closes game.....
        Case "Exit"
            If GameState = "Exit" Then Error("")

        'Game setup.....
        Case "Initialise"
            'turns
            turn = 5
            'Ball setup
            ball.speed = 5
            ball.x=375
            ball.y=295
            'Target setup
            target.x=370
            target.y=125
            'Scoring setup
            CPU_Score=0
            User_Score=0

            'Start the game
            GameState = "Playing_s"

        'player is goalkeeper.....
        Case "Playing_gk"
            'moving goalkeeper
            If KeyHit (KEY_LEFT) Then saver.Move_x(-10)
            If KeyHit (KEY_RIGHT) Then saver.Move_x(10)
            'moving between keeper and shooter
            If KeyHit (KEY_N) Then shooting = True
            If shooting = True Then GameState= "Playing_s"

            'turns
            If KeyHit(KEY_N) Then turn = turn - 1
            If turn = 0 Then GameState = "End"

            'scoring
            If Saved = True Then Cpu_shot=False
            CPU_Score = CPU_Score
            If Saved = False Then Cpu_shot=True
            CPU_Score = CPU_Score + 1

        'player is shooting.....
        Case "Playing_s"
            'moving between shooter and keeper
            If KeyHit (KEY_N) Then shooting = False
            If shooting = False Then GameState= "Playing_gk"
            'moving target
            If KeyHit (KEY_A) Then target.Move_x(-10)
            If KeyHit (KEY_D) Then target.Move_x(10)
            If KeyHit (KEY_S) Then target.Move_y(10)
            If KeyHit (KEY_W) Then target.Move_y(-10)
            'moving ball to target
            If KeyHit (KEY_SPACE)
            'Ball moving to target from spot
            While target.x<>ball.x Or target.y<>ball.y
                If ball.x < target.x Then ball.x +=ball.speed
                If ball.x > target.x Then ball.x -=ball.speed
                If ball.y < target.y Then ball.y +=ball.speed
                If ball.y > target.y Then ball.y -=ball.speed
            End
            'Scoring
            Saved = False
            If Saved = True Then User_shot=False
            User_Score = User_Score
            If Saved = False Then User_Score=True
            User_Score = User_Score + 1
            End

            'checks collision
            If Intersects(saver.x,saver.y,65,113,ball.x,ball.y,55,55) Then collision_test = true

    End
End

```

```

'Renders graphics
'This code is responsible for loading images to the screen
Method OnRender()
'gamestate selection
Select GameState

'Draw menu
Case "Menu"
    DrawImage (menubackground, 0, 0)

'Draw customize tag screen
Case "Customize_tag"
    Cls 225,0,0
    If tag = False Then DrawText("Invalid Tag Try Again", 0, 0, 0)
    If Not tag = False Then GameState = "Playing_gk"

'Draw instruction menu
Case "Instructions"
    Cls 0,225,0

'Draw Goalkeeper images and sprites
Case "Playing_gk"
    'Background
    DrawImage (gamebackground, 0, 0)
    'Goalkeeper
    DrawImage saver.sprite, saver.x, saver.y
    'Ball
    DrawImage ball.sprite, ball.x, ball.y
    'Scoring
    DrawImage (scoreboard,350,0)
    'drawing white, green and red dots to the screen
    DrawImage (whitedot1, 477,16)
    If Cpu_shot = False Then DrawImage (reddot,477,16)
    If Cpu_shot = True Then DrawImage (greendot,478,16)

'Draw Shooting images and sprites
Case "Playing_s"
    'Background
    DrawImage (gamebackground, 0,0)
    'Goalkeeper
    DrawImage saver.sprite, saver.x, saver.y
    'Target
    DrawImage target.sprite, target.x, target.y
    'Ball
    DrawImage ball.sprite, ball.x, ball.y
    'Scoring (testing scoring idea)
    DrawImage (scoreboard, 350,0)
    DrawImage (whitedot1, 477,16)
    If Cpu_shot = False Then whitedot1 = reddot
    DrawImage (reddot,477,16)
    If Cpu_shot = True Then DrawImage (greendot,478,16)
    'collision
    If collision_test = True Then
        DrawText("collision", 10, 10)
    End
End

'Draw end screens
Case "End"
    Cls 0,0,0
End
End
End

```

```

'Class for goalkeeper*****
Class goalkeeper
'
    Field sprite:Image = LoadImage ("goalkeeper.png")
    Field x:Float = 370
    Field y:Float = 125

    Method Move_x(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>545 Then x=545
    End
End

'Class for crosshairs*****
Class shooter
    Field sprite:Image = LoadImage ("target.png")
    Field x:Float = 370
    Field y:Float = 125

    Method Move_x(x_distance:Int)
        x+=x_distance
        If x<190 Then x=190
        If x>560 Then x=560
    End

    Method Move_y(y_distance:Int)
        y+=y_distance
        If y<80 Then y=80
        If y>175 Then y=175
    End
End

'Class for football*****
Class football
    Field sprite:Image = LoadImage ("ball.png")
    Field x:Float = 375
    Field y:Float = 295
    Field speed:Int
    Field speed_x:Int
    Field speed_y:Int
End

'Runs the game
Function Main()

    Game = New PenaltyGame

End

Function intersects:Boolean (x1:Int, y1:Int, w1:Int, h1:Int, x2:Int, y2:Int, w2:Int, h2:Int)
    If x1 >= (x2 + w2) Or (x1 + w1) <= x2 Then Return False
    If y1 >= (y2 + h2) Or (y1 + h1) <= y2 Then Return False
    Return True
End

```

Candidate Name: [REDACTED]

Candidate Number: [REDACTED]