Version 1.1

University of Staffordshire

Do You Know Ball?

Technical Design Document

s013671m@student.staffs.ac.uk

SLASKI, Jack

Contents

Project Introduction 2
Project Goals 2
Challenges and Risks 2
Hardware Requirements 2
Systems and Diagrams 3
Gameplay Mechanics 3
Game Mode 3
Game Instance 7
Answer Panel 8
Ball 8
Target 8
UI & Widgets 9
Game Over Screen 9
Question UI 9
Daily Login 9
Penalty Score Indicator 10
Opponent Select 10
Shop 10
Ad Screen 10
Technical Specs 11
Engine Choice 11
Testing 11
Game Mechanics 11
Main Menu 11
In Game 12
Data and Persistence 12
Save Data 12
Coding Standards 13
Programming Standards 13
Style Guide 13
Commenting Rules 14
Code Review Procedures 14
Production Overview 14
Moscow 14

SLASKI, Jack
s013671m@student.staffs.ac.uk

Project Introduction

The idea for this game is to create a mobile football quizzing game titled "Do You Know Ball?". In this game, the player competes in various penalty shootout game modes by answering football-themed trivia questions. The trivia system takes inspiration from "Who Wants To Be A Millionaire?", featuring multiple choice answers and assistive power-ups.

The core concept is to combine the high-stakes tension of a penalty shootout with the mental challenge of football knowledge, resulting in a unique and engaging gameplay loop designed specifically for mobile users. Players progress by answering questions correctly to score goals, with different modes offering varied pacing and difficulty.

Project Goals

The ultimate goal for this project is to create a fully functional, polished, and replayable mobile game experience that is both fun and technically sound. By focusing on clean Blueprint architecture, good mobile UX practices, and thoughtful game design, this project aims to provide a strong example of a complete, self-contained mobile title.

This foundation allows room for future iteration, including expanded question banks, online functionality, or more complex game modes, while remaining performant and readable.

Challenges and Risks

The challenges this project is aiming to solve are as follows:

- Designing an engaging quiz mechanic that feels responsive and rewarding.
- Creating multiple game modes with varied pacing (e.g. Quick Play, Versus, Endless).
- Implementing a scalable trivia system with difficulty tiers.
- Ensuring that the game runs optimally in a mobile environment.
- Creating a responsive UI that fits a range of screen sizes and resolutions.
- Managing game data persistence across sessions, including save systems and high score tracking.
- Implementing a high score leaderboard that tracks score and date.
- Balancing game difficulty to allow progression without frustration.
- Using Blueprint scripting efficiently to maintain clean, readable, and optimised logic.

Hardware Requirements

Recommended Hardware:

Operating System: Windows 10 64-bit version 1909 revision .1350 or higher, or versions 2004 and 20H2 revision .789 or higher

Processor: Quad-core Intel or AMD, 2.5 GHz or faster

Memory: 32 GB RAM

Graphics RAM: 8 GB or more

Graphics Card: DirectX 11 or 12 compatible graphics card with the latest drivers

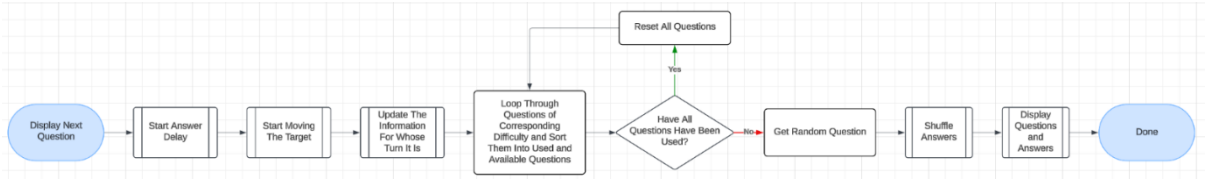
Systems and Diagrams

To plan out each system and mechanic in this project a detailed flowchart has been created.

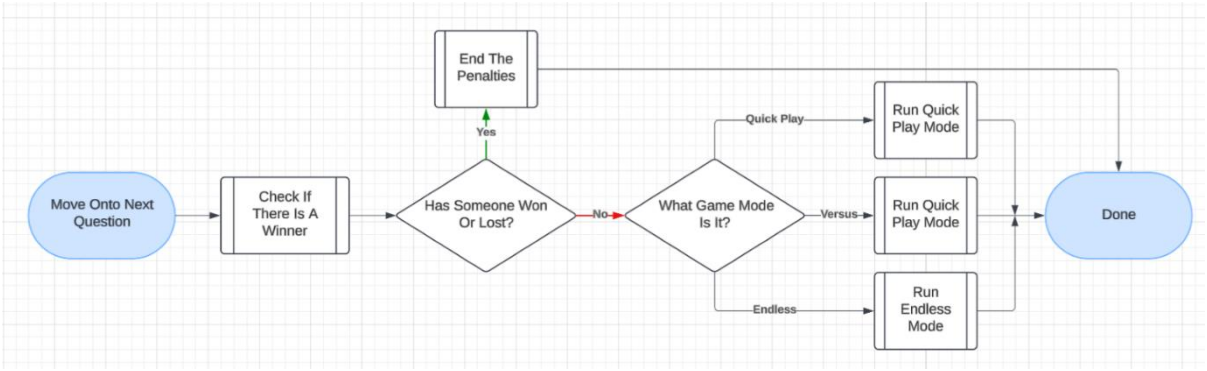
Gameplay Mechanics

Game Mode

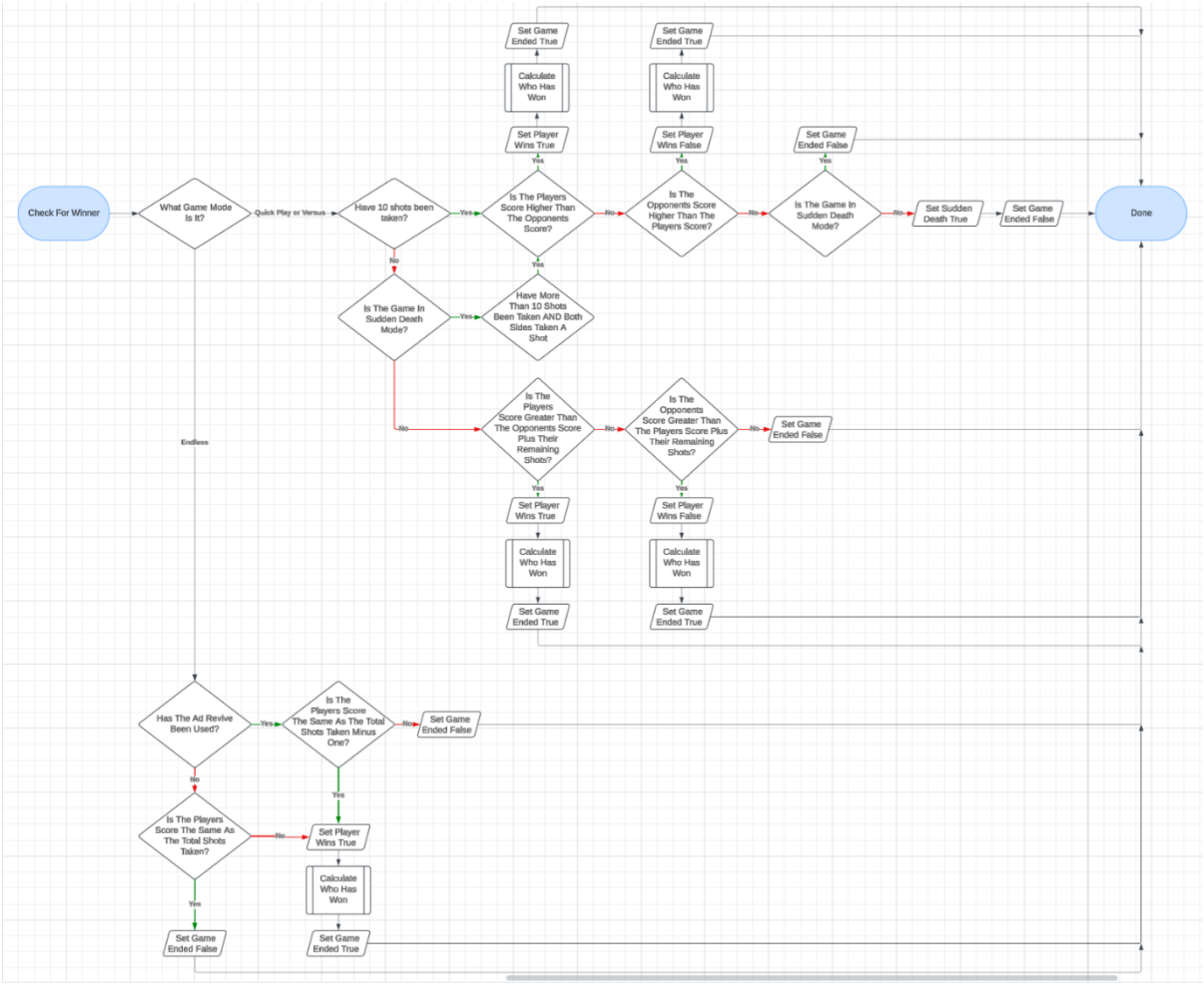
Here is a flowchart showing the logic of sorting through the questions and choosing one to display:



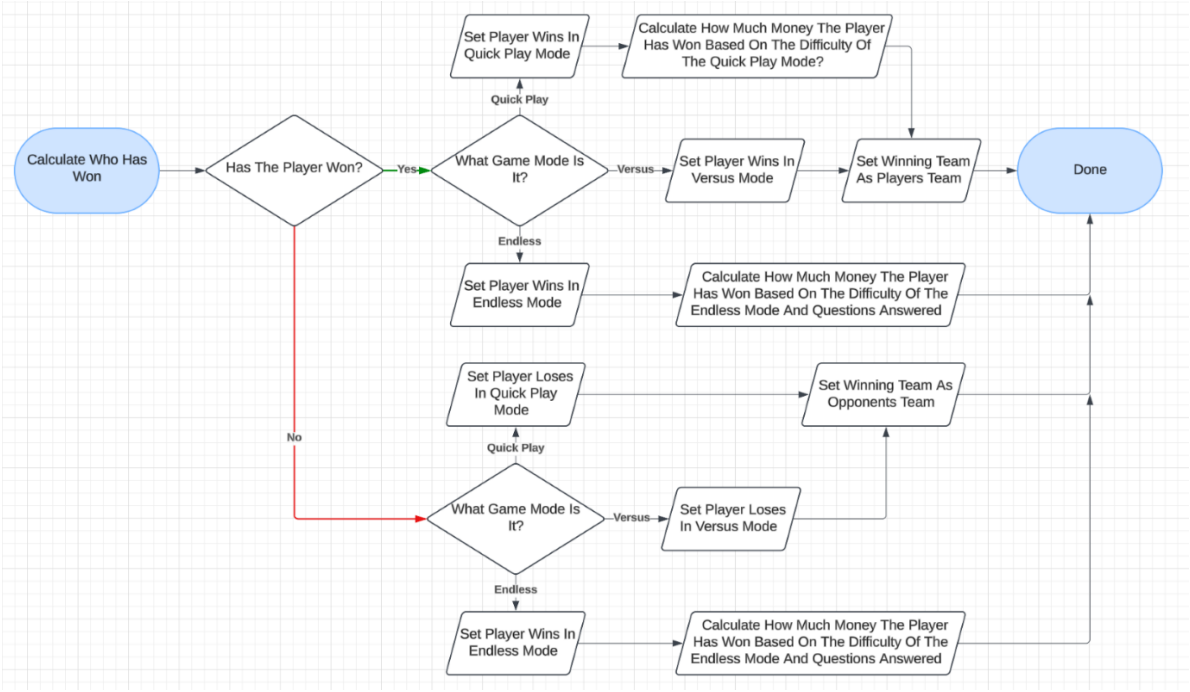
Here is a flowchart showing the logic of moving onto the next question after a penalty has been taken:



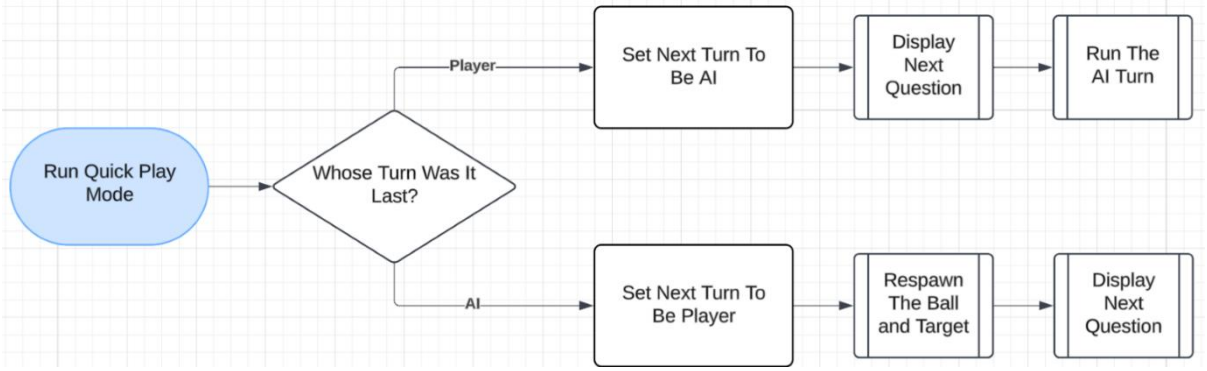
Here is a flowchart showing the logic of checking if the game has been won or lost:



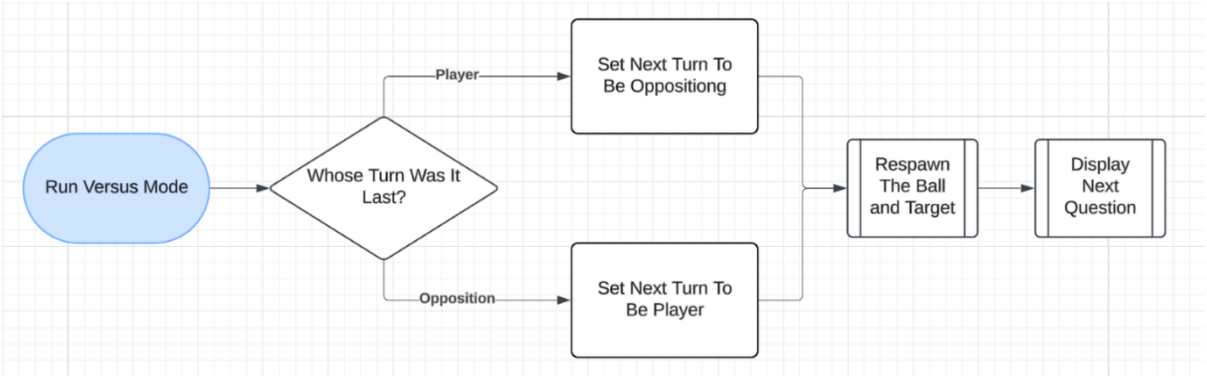
Here is a flowchart showing the logic of calculating who has won and how many coins they have won:



Here is a flowchart showing the logic of running the quick-play game mode:

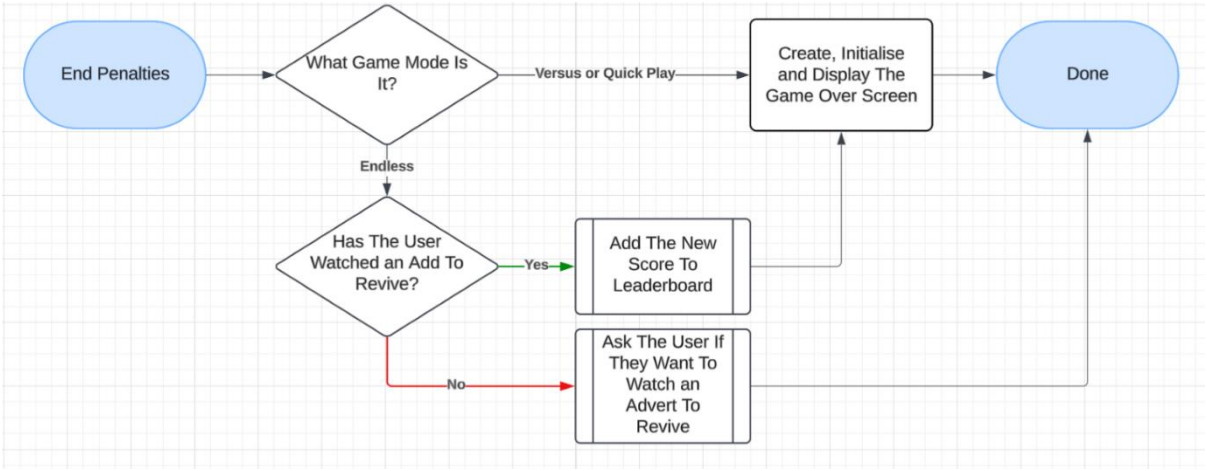


Here is a flowchart showing the logic of running the versus game mode:

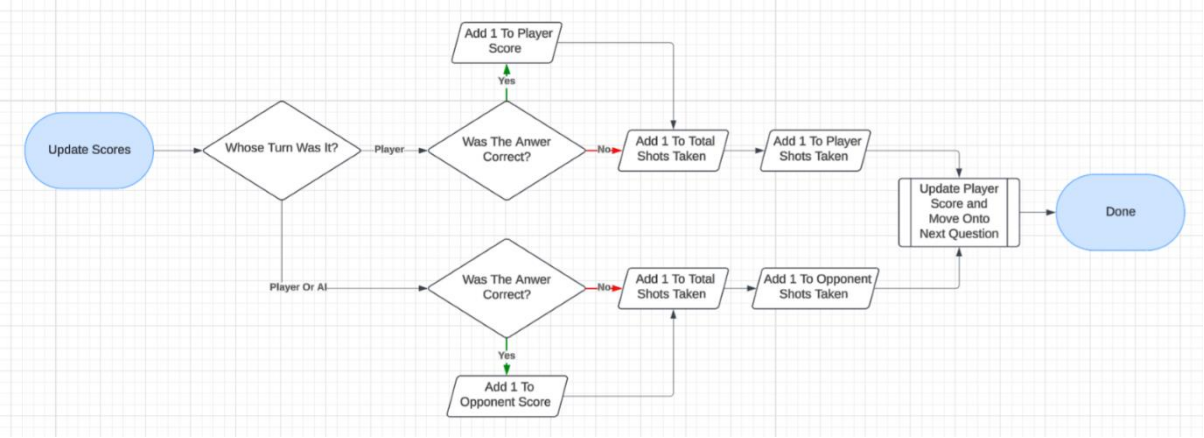


Here is a flowchart showing the logic of running the endless game mode:

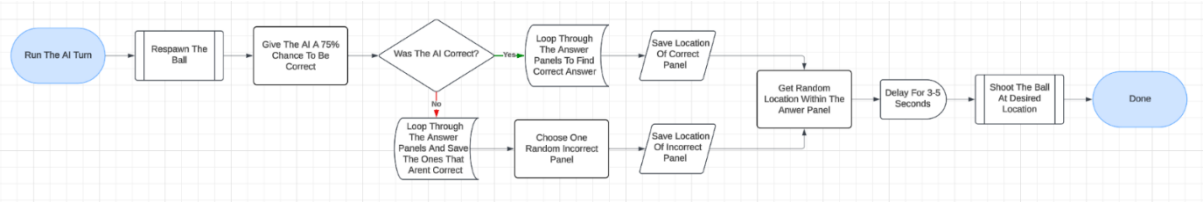
Here is a flowchart showing the logic of ending the penalties:



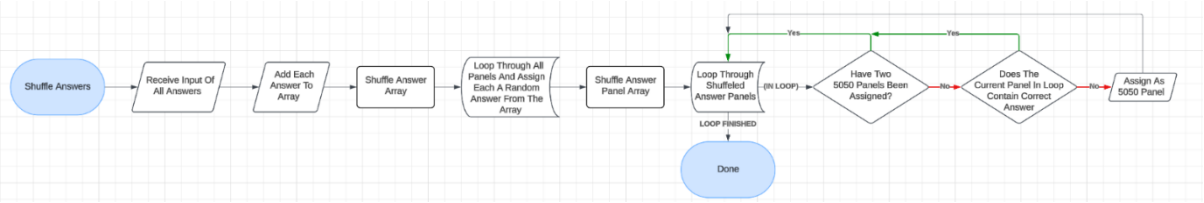
Here is a flowchart showing the logic of updating the scores for the current round:



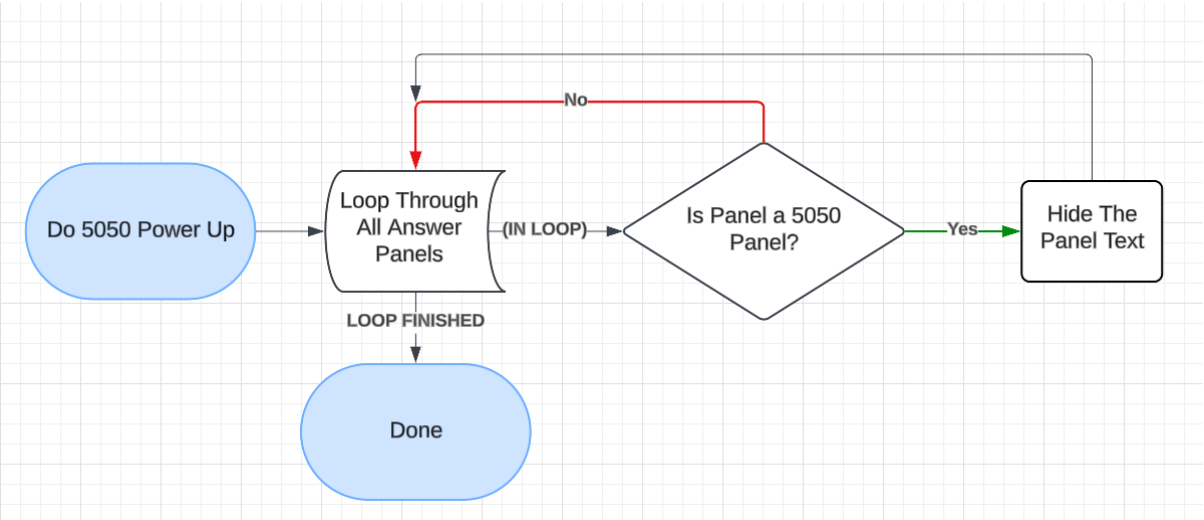
Here is a flowchart showing the logic of the AI turn for the versus or quick-play game mode:



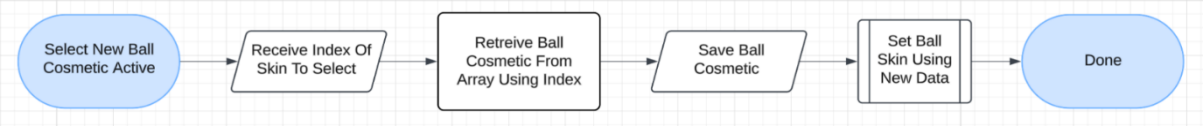
Here is a flowchart showing the logic of shuffling and initialising the answer panels:



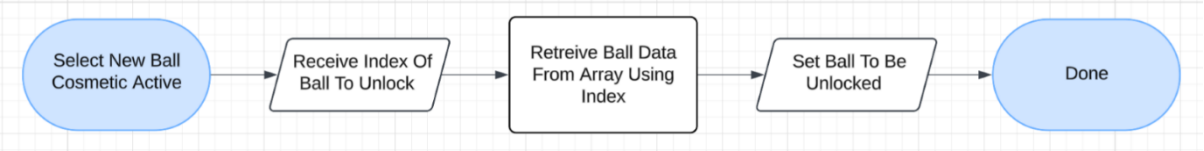
Here is a flowchart showing the logic of doing the 5050 power-up:



Here is a flowchart showing the logic of select a new ball cosmetic in the shop:

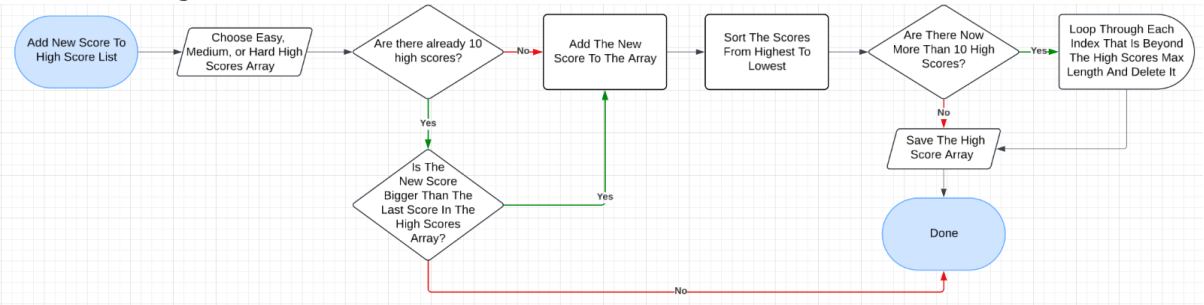


Here is a flowchart showing the logic of unlocking a new ball cosmetic in the shop:

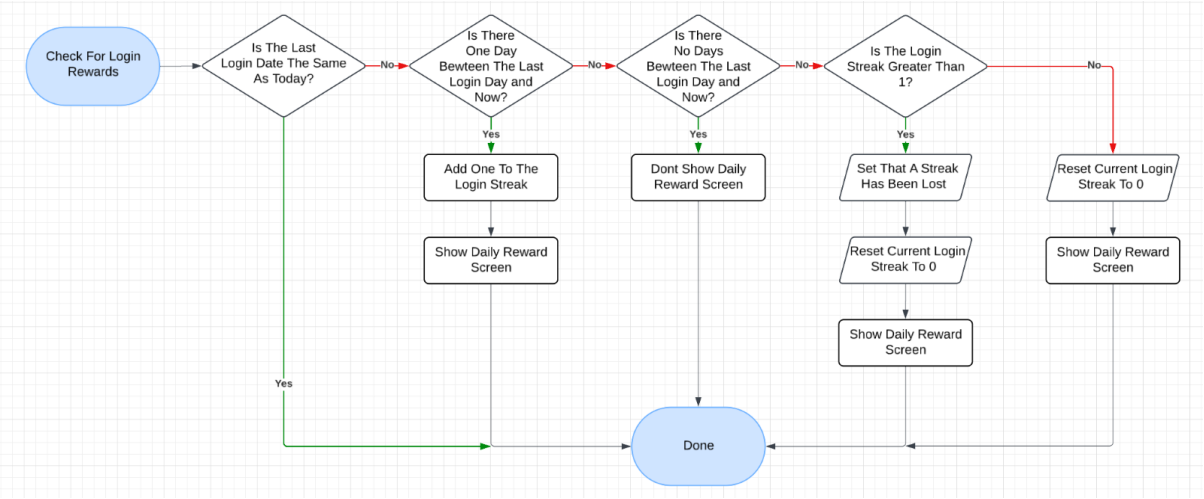


Game Instance

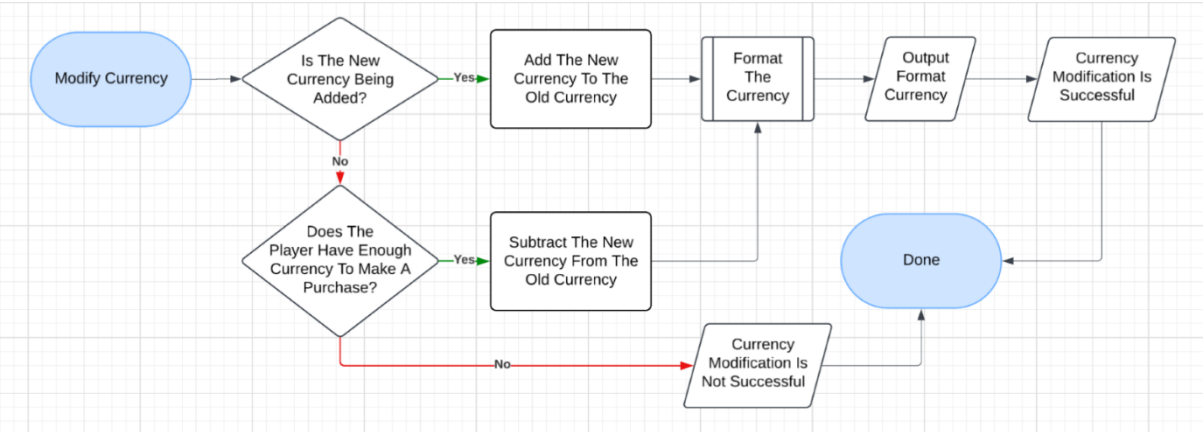
Here is a flowchart showing the logic of a new final score being checked/added to the high scores after a game in endless mode has been finished:



Here is a flowchart showing the logic of checking the current date for the daily login rewards:

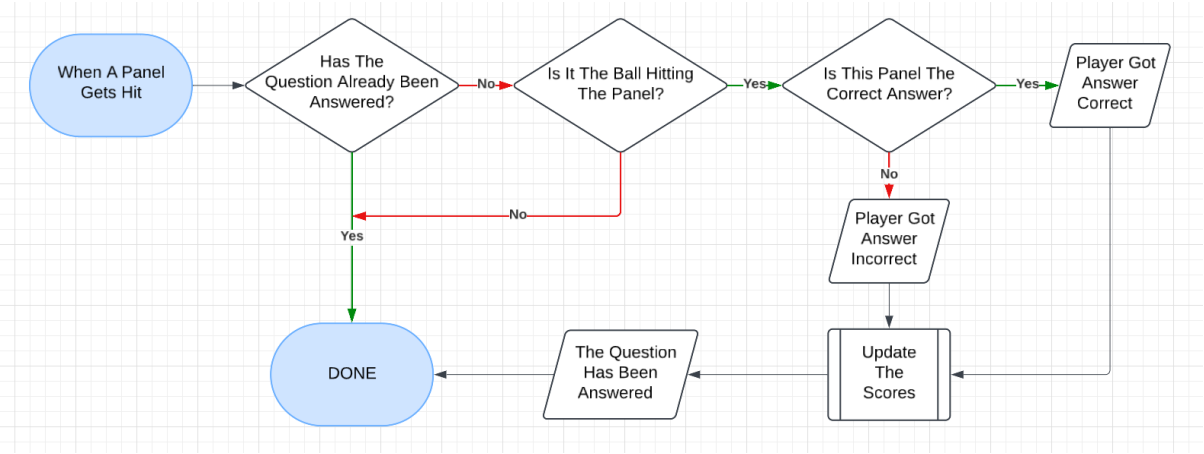


Here is a flowchart showing the logic of the users currency being modified:



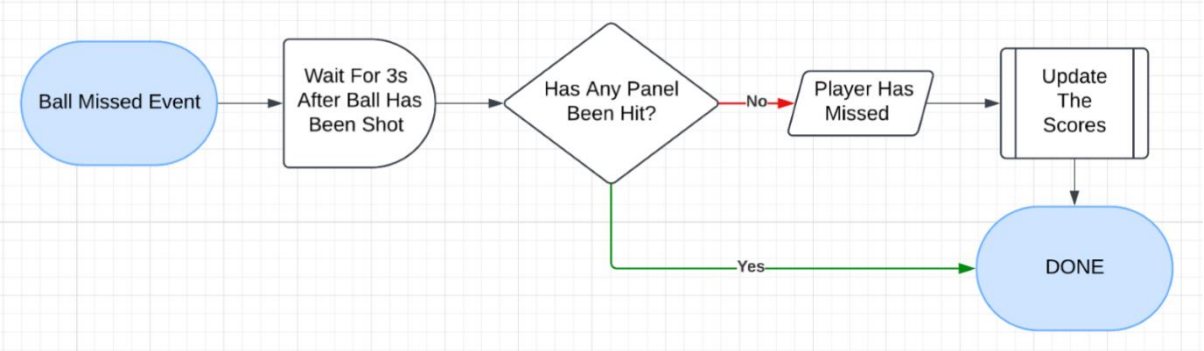
Answer Panel

Here is a flowchart showing the logic of when an answer panel is hit by a ball:



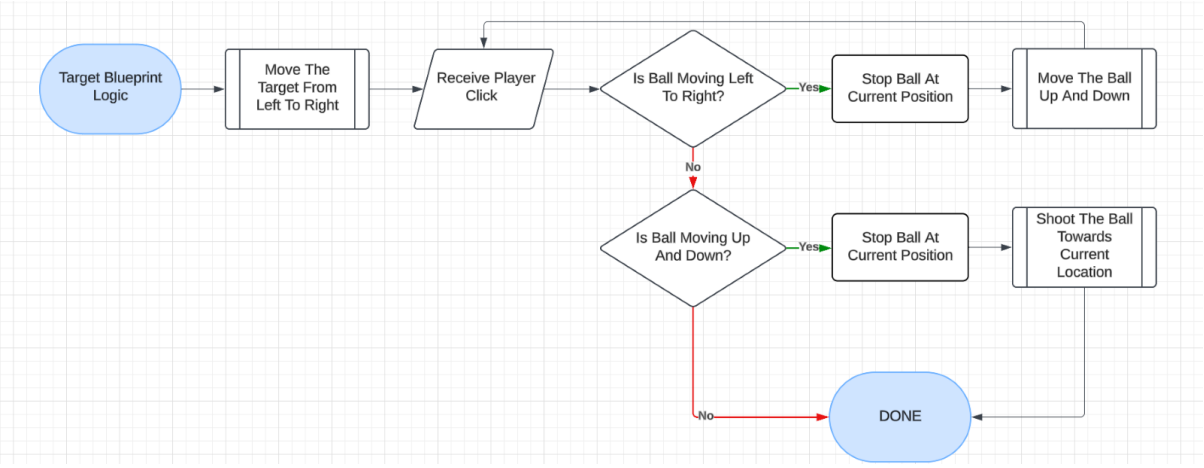
Ball

Here is a flowchart showing the logic of a ball missing an answer panel:



Target

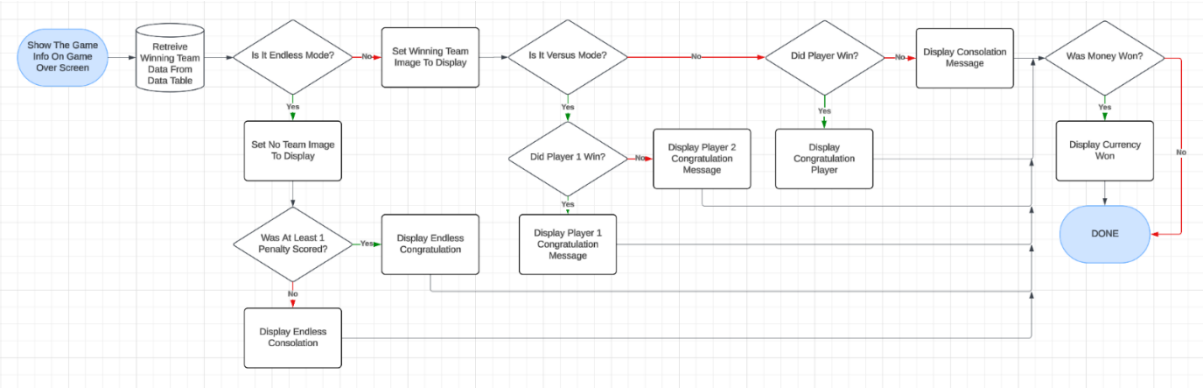
Here is a flowchart showing the logic of the target moving to answer a question:



UI & Widgets

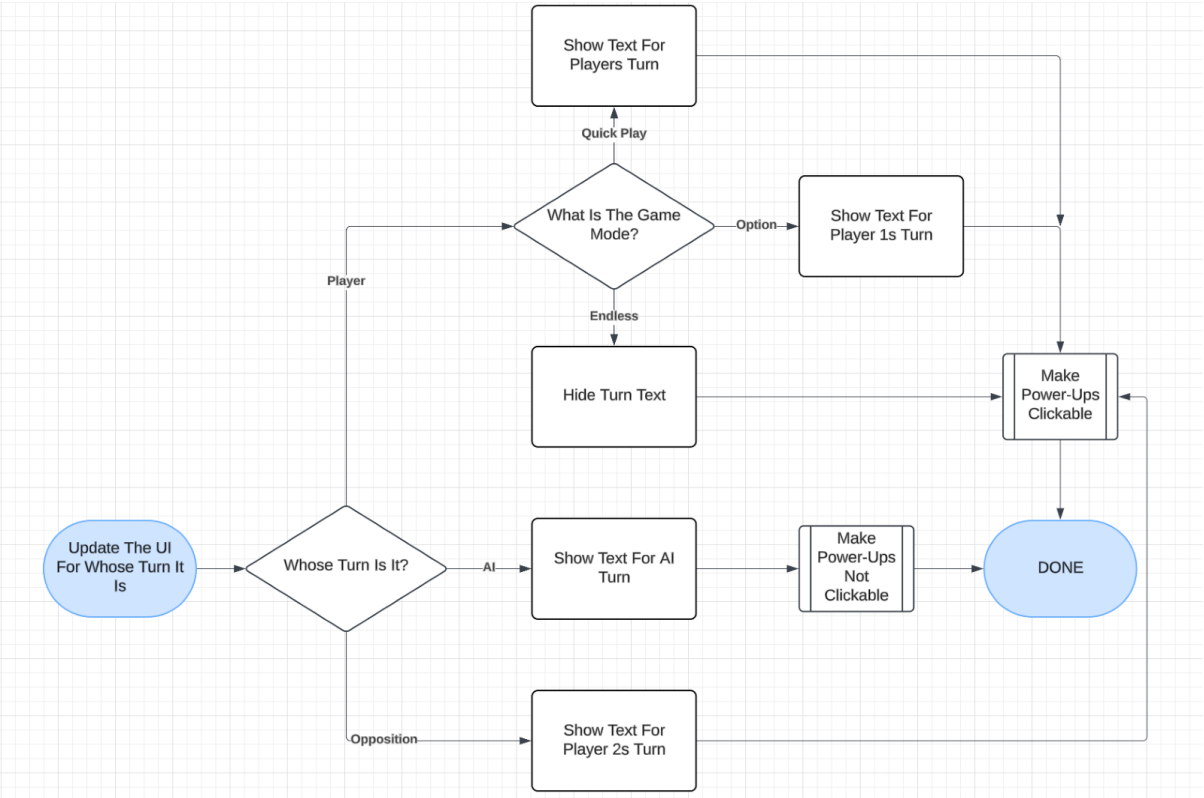
Game Over Screen

Here is a flowchart showing the logic of updating the game over screen:



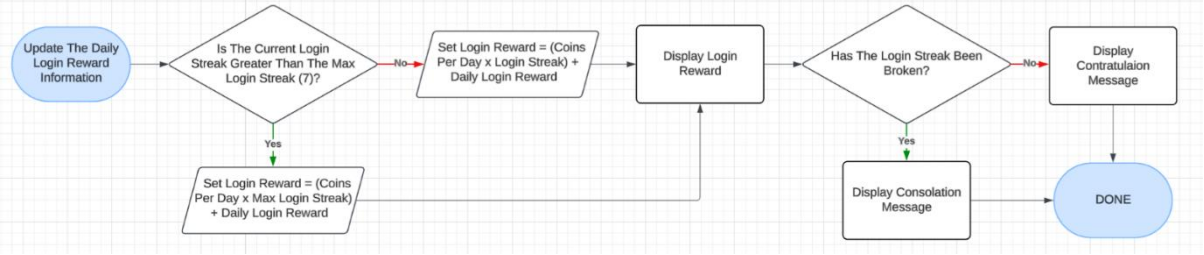
Question UI

Here is a flowchart showing the logic of updating the UI to display whose turn it is:



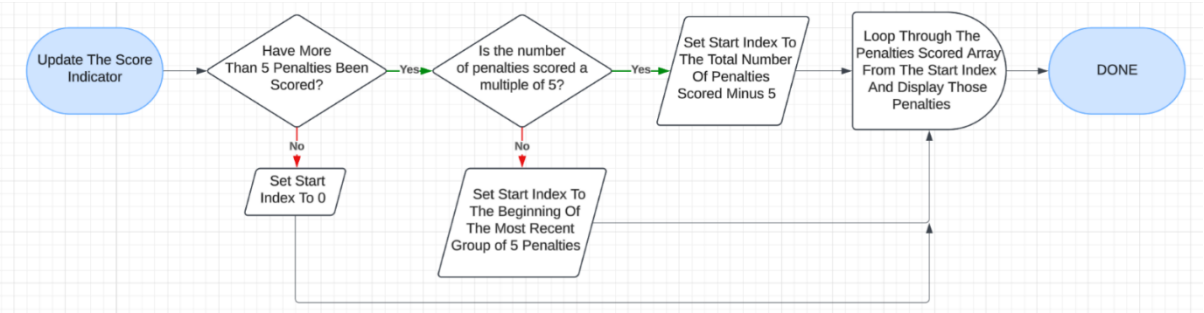
Daily Login

Here is a flowchart showing the logic of updating the daily login rewards UI:



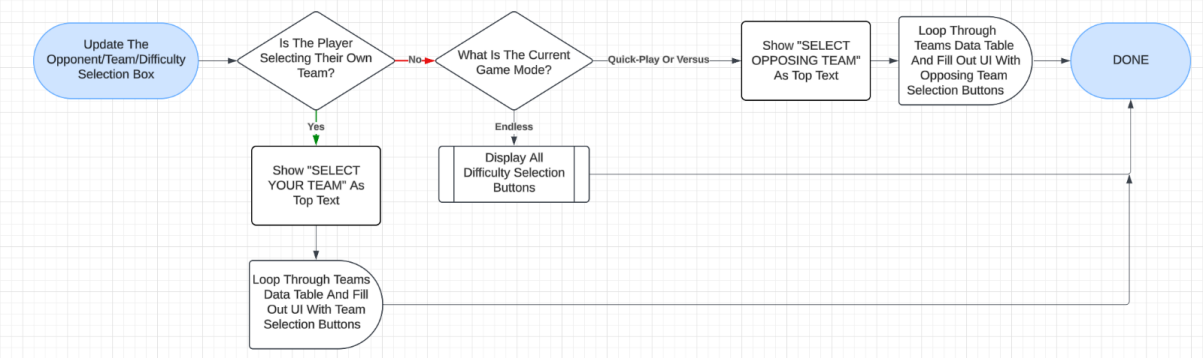
Penalty Score Indicator

Here is a flowchart showing the logic of updating penalty score indicator:



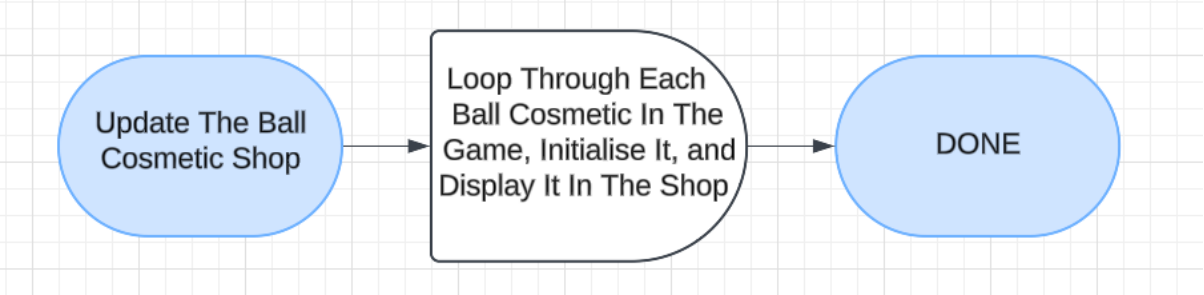
Opponent Select

Here is a flowchart showing the logic of updating the team/opponent/difficulty selection UI:



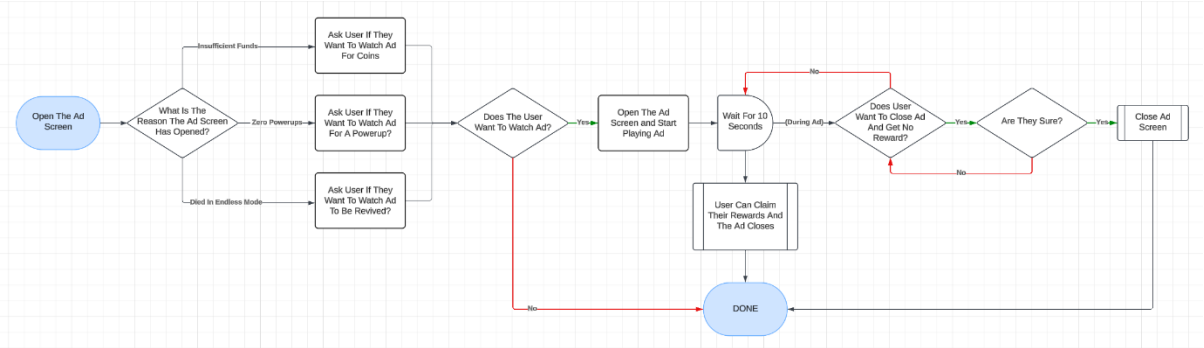
Shop

Here is a flowchart showing the logic of updating the ball cosmetics in the shop:



Ad Screen

Here is a flowchart showing the logic of checking if the user wants to watch an Ad:



Technical Specs

Engine Choice

My chosen engine for this project is Unreal Engine 5.4. With my development skills and the game type Unreal Engine 5 would be the most suitable engine for me to be able to create the most detailed and effective project I can.

Testing

This game will require a comprehensive testing plan to cover all areas of the game. Below are each of the laid-out areas and individual features/parts that will need to be tested for this project:

Game Mechanics

Main Menu

Game Mode Buttons:

- Loading
- Opponent Select
- Difficulty Selection
- Animations
- Sound
- Transition
- Game Start

Shop Button:

- Transitions
- Loading
- Shop Persistence
- Powerups
- Coins
- Purchasable Cosmetics and Items

Options Button:

- Transitions
- Settings
- Quitting

Leaderboard Button:

- Leaderboard Data
- Animation
- Scaling

Team Select Button:

- Countries
- Difficulties
- Spacing
- Scaling

Daily Rewards:

- Animation
- Appearance at the right time
- Claiming
- Max reward
- Day Tracking

In Game

Penalty Shooting:

- Target Movement
- Penalty Accuracy
- Spam Protection

Game Panels:

- Panel Text
- Answer Correctness
- Panel Collisions

Powerups:

- Powerup Use
- Powerup Button
- Powerup Count

Pause Menu:

- Gameplay Stops
- Confirmation Menus
- Resume Button
- Quit and Main Menu Buttons

Questions:

- Question Accuracy
- Hint Text
- Answer Text
- Question Text
- Font

Football:

- Ball Movement
- Ball Collisions
- Ball Size
- Ball Material

Data and Persistence

Save Data

Game Instance:

- Level Persistence
- Data Resetting
- Game Start
- Game End
- Autosave

Save Game:

- Saving Data
- Loading Data

Settings:

- Saving Settings
- Applying Settings
- Data Resetting
-

Analytics:

- Data Tracking
- Data Saving
- Data Debugging

Coding Standards

Programming Standards

For naming conventions, I will attempt to follow the unreal engine recommended asset naming conventions linked here:

https://dev.epicgames.com/documentation/en-us/unreal-engine/recommended-asset-naming-conventions-in-unreal-engine-projects

For efficiency rules, I will attempt to make as much of the blueprint code as modular as I possibly can. This means creating large parent actors to create other world objects and actors, the use of enumeration and structures to ensure that data is correctly and efficiently laid out.

In terms of variable naming conventions, I use PascalCase with a '?' suffix for Booleans which ensures that my blueprint code is clear, concise and easily readable.

Style Guide

There are many practices that I will be using to ensure that my code is readable.

Firstly, I use initialisation chains to ensure a clear and obvious hierarchy between different scripts. Along with this, using initialisation chains means that different variables and scripts can be passed down through the code. This makes the project much more efficient as rather than using casting or interfaces, all of the project initialisations can be done and startup, and scripts at the bottom of the hierarchy chain can access other scripts on the same level.

Asides from commenting, I ensure that my blueprint code is neat and tidy by making full use of the 'Straighten Connections' feature to ensure that the connection between two nodes is straight. This ensures that my blueprint logic maintains a neat and tidy formatting.

Furthermore, I organise my nodes in clear left-to-right logical flow, and space them out appropriately to avoid clutter. When necessary, I also make use of reroute nodes to keep wires from overlapping and maintain an uncluttered layout. Grouping related logic with comment boxes or color-coded node groups is another technique I use to keep the visual scripting environment tidy and easy to follow.

Finally, one of the last practices that I use to ensure the script is readable is the use of functions to break up large code blocks. Instead of having a large chain of code, I make use of functions that can handle code that may be repeated or called again. This prevents long and difficult to read code blocks and can instead shorten some of it by replacing it with well named and obvious functions.

Commenting Rules

For commenting rules, I will be commenting all larger and important blocks of code with a simple comment title that briefly explains the functionality of the block. For comment colours I also plan on using certain colours for certain types of code. For example, using yellow for debug code and blue for initialisation code. This is being done to ensure that the code is readable. If code is readable it allows for collaborative work with other developers, future project development after a long time a way (comments act as reminders), or even the ability to move the project on to someone else or another company.

Here is a table showing my comment types and their colours:

Comment Type	Hex sRGB
Initialisation	0000FFFF
Event	FF0600FF
Function	802399FF
Comment within Comment	000000FF
Debug	FFEC00FF

Code Review Procedures

The code in my project will be reviewed very regularly. This is because I am going to be learning a lot about unreal during this development process, I plan on noting down areas of inefficient code during the creation process so that they may be returned to later. Furthermore, as my knowledge of the best coding practices increases as well as my skill in unreal, I will be consistently updating and improving code throughout the project.

Production Overview

Moscow

Must Have:

- A range of varied and up to date football trivia questions with a range of answers including one that is correct, three that are wrong, and a hint for each.
- A full penalty shootout system that can continue until there is a winner.
- Basic monetisation such as watching adverts to be able to try again.

Should Have:

- Different game modes such as versus mode, quick play, and endless.
- Different powerups for the trivia such as 50/50 mode, hint, etc.
- More advanced monetisation such following powerups similar to those from "Who wants to be a millionaire?" such as 50/50 mode, hint, etc.
- Different selectable difficulties
- In game currency that can be gained through how well the user has played

Could Have:

- A leaderboard system to display certain things like endless mode points or who has the most in game currency.
- A shop system where the user can buy new skins for the ball, powerups, and more with in game currency.
- A way to purchase in game currency or bundle packs for real money.

Won't Have (but could be done in the future):

- Seasonal / Event modes such as World Cup, Euros, Christmas etc
- Different Trivia Layouts such as guess the player, or football player wordle.