Penalty Kick Game

A simple game built in Unity3D simulating a penalty shootout with obstacles

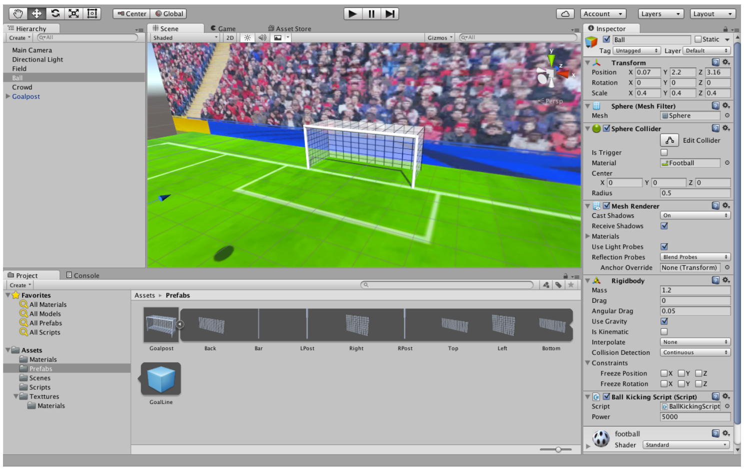
Introduction

The idea for this game was inspired by New Star Soccer which is a game for desktop and touch devices where the primary gameplay is clicking or tapping on a moving ball to shoot it towards the goal. There are many other elements to that game but this is the one that I am modifying by adding obstacles instead of a goal keeper. I'm also using a different method to shoot the ball. These modifications change the focus of the game to make it more of a puzzle and skill game instead of a strategy and management game.

Development

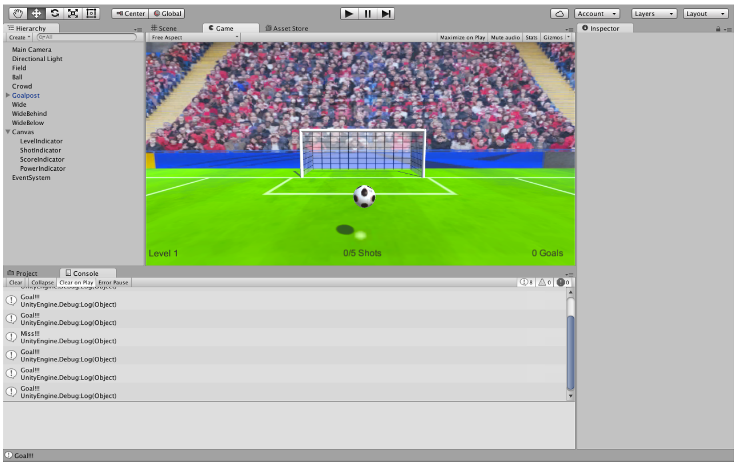
The first step of development was to build a simple plane to represent the playing field and a sphere as the football. I coded a simple script to create the basic mechanics of kicking the football. I wanted the direction of the ball to be dependent on where on the sphere the player clicks so I used Ray Tracing to determine the location of the mouse click. I then subtracted the Vector3 of this location from the centre of the sphere (football) to provide a vector in the correct direction. By applying a force to the football in this direction, I was able to send the ball in the correct location. For example, if the player click on the bottom right of the football, it is sent up and to the left.

Once the basic mechanics were working, I focused on improving the appearance of the game. I found a tutorial (http://www.thegamecontriver.com/p/how-to-make-penalty-shootout-kick-game.html) on building a different penalty shootout game which contained textures for the football field, crowd, and football. It also had a prefab of the goal which I was able to import. I then created new physics materials for the ball and goal for more accurate behaviour. This goal contained a trigger plane in the goal mouth which I added a goal tag to. I then added in three more trigger planes: one behind the goal, one behind the camera position, and one beneath the playing field. I was then able to add a collision handler to the ball script which would detect when it hit one of these planes and which one it hit. I logged if it was a goal or miss to the console and reset the level when this happened.



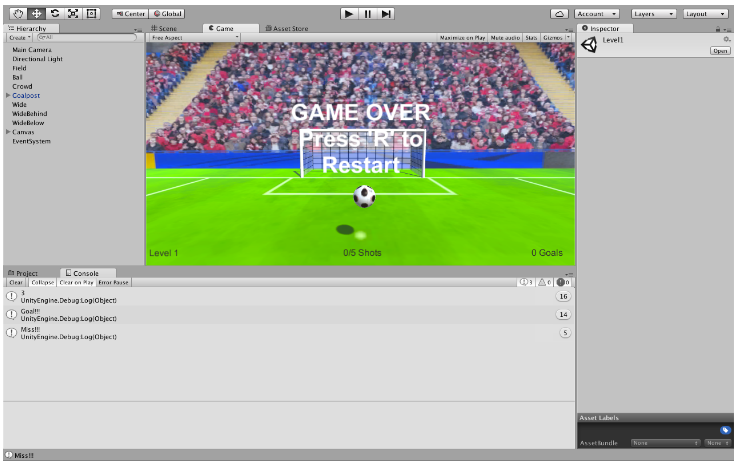
Ball ☑ ☐ Static
Tag Untagged Layer Default
Transform
Position X 0.07 Y 2.2 Z 3.16
Rotation X 0 Y 0 Z 0
Scale X 0.4 Y 0.4 Z 0.4
Sphere (Mesh Filter)
Mesh Sphere
☑ Sphere Collider
Is Trigger ☐
Material Football
Center X 0 Y 0 Z 0
Radius 0.5
☑ Mesh Renderer
Cast Shadows On
Receive Shadows ☑
Materials
Use Light Probes ☑
Reflection Probes Blend Probes
Anchor Override None (Transform)
Rigidbody
Mass 1.2
Drag 0
Angular Drag 0.05
Use Gravity ☑
Is Kinematic ☐
Interpolate None
Collision Detection Continuous
Constraints
Freeze Position ☐X ☐Y ☐Z
Freeze Rotation ☐X ☐Y ☐Z
☑ Ball Kicking Script (Script)
Script BallKickingScript
Power 5000
football
Shader Standard

The next step was to create a GUI layer to provide feedback to the player such as the number of goals, which level they are on, and the number of shots left in the level. I then added in the functionality to change the power of the kick. This required changing the script to kick the ball when the mouse button is released and adding in some new code when the mouse button is held down. This code increments and decrements the power variable within the min and max values. I displayed this value initially on a text label on the right of the display. Once I had the logic working smoothly, I changed it to a visual power bar to provide more intuitive feedback.



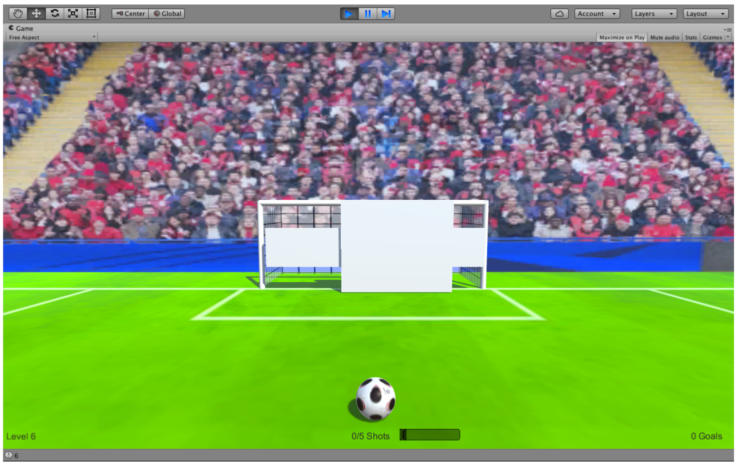
```
Goal!!!
UnityEngine.Debug:Log(Object)
Goal!!!
UnityEngine.Debug:Log(Object)
Miss!!!
UnityEngine.Debug:Log(Object)
Goal!!!
UnityEngine.Debug:Log(Object)
Goal!!!
UnityEngine.Debug:Log(Object)
Goal!!!
UnityEngine.Debug:Log(Object)
Goal!!!
```

The last stage for this level was to extend the GameController script to update levels and show a game over message when there are no more levels. This message also shows the player how many goals they scored out of a possible total. I also implemented a key listener to restart the game when the user types 'R'. This resets the key variables in in the GameController script and allows the player to play again. I coded it in such a way that I could easily extend the number of levels and only change one variable in the script. One issue with the GUI layer was that the text labels and progress bar were based on pixel measurements and therefore did not scale well. To resolve this, I changed the anchor points of the fixed UI elements and changed the location that the progress bar displays at to use Screen.width and Screen.height to calculate the position relative to the screen dimensions.

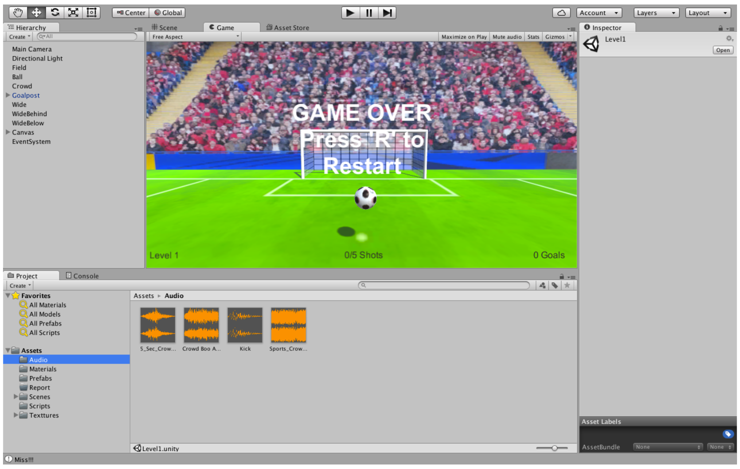


```
3
UnityEngine.Debug:Log(Object)      16
Goal!!!
UnityEngine.Debug:Log(Object)      14
Miss!!!
UnityEngine.Debug:Log(Object)       5
Miss!!!
```

My next step was to add more levels to the game to provide a sense of escalating challenge. The next 2 levels have a thin stationary box placed infant of the goal which the player has to avoid. It is more difficult to chip the ball higher so I made the higher level with the bottom half of the goal blocked. The next 3 levels consist of moving boxes that block the player's shots. To create the scripts for the moving boxes, I simply translate the box along an axis within bounds that keep it in front of the goal. Each level contains different size boxes moving at different speeds. This provides an escalating sense of difficulty and a sense of challenge for the player.



```
6
```



GAME OVER
Press 'R' to
Restart

Miss!!!

Once I was happy with the general mechanics, I started to add a couple finishing touches like audio for the crowd, goals and misses. I found some creative commons sound effects on SoundBible.com which were suitable for use in the game such as crowd noise, kicking, cheers and boos. This required changing some code in the controller to delay the loading of levels so sounds have a chance to play. I also made some minor cosmetic improvements around the collision of the ball to make the game more realistic.

Conclusion

Overall, I coded the game in such a matter that it is easily extensible and robust. To add a new level, you just need to drop in a new scene and all of the scripts can dynamically handle the addition. It would also be possible to create new scenes simulating free kicks where the user has to chip the ball over a wall. I am happy with the result of the game and I find that it provides enough of a challenge to remain interesting while being very simple to play.

How to Play

The aim of the game is to score as many goals as possible. It starts off as simply as possible, with an open goal. To shoot the football, hold the mouse down on the ball to set the power and direction and release to shoot. You can shoot until the ball goes wide, in the goal, or back behind the player. Once the player has taken 5 shots, they move up a level where more obstacles are added in the goal. There are a total of 6 levels available to play. Once the user runs out of levels, they are showed a game over screen with their final score. As per the instructions on-screen, just hit 'R' to play again.