T3Dmake

Simple basketball & soccer

Documentation

Thank you for purchasing simple basketball and soccer

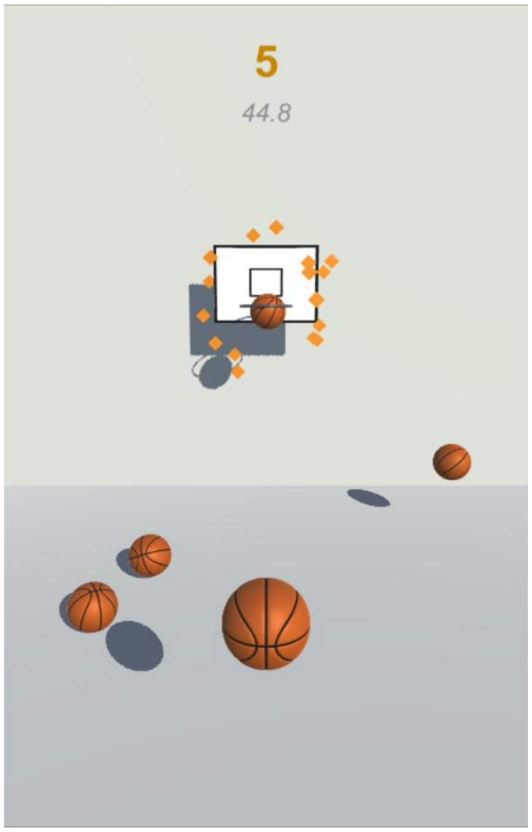
Basketball

The basketball game is a relatively simple game. The main goal is to throw basketballs in a hoop. The game has a counter so the game is over after 60 seconds. The UI is handled by the main script attached to the camera and the timer as well. To shoot the ball, players must swipe in the right direction.

```
void OnMouseExit(){
//if you don't hit the ball anymore, get position and use end position to get x shooting force
if(Input.GetMouseButton(0) && shooting){
end = Input.mousePosition;
GetComponent<Rigidbody>().isKinematic = false;
GetComponent<Rigidbody>().AddForce(new Vector3(((end.x - start.x) * forceX), forceY, forceZ));
shooting = false;

//after shooting, instantiate new ball and destroy this ball after 10 seconds
StartCoroutine(newBall());
Destroy(gameObject, 10);
}
}
```

As you can see in the OnMouseExit function, the y and z force will always remain the same. This means that the player only controls the x direction of the ball. I've tried more complex swipe but the game would become way too difficult. OnMouseExit means that as soon as the player releases the ball, the force is added and the player can't control it anymore. If you would like to have more control during the swipe, you could replace OnMouseExit with OnMouseUp and the force would be added when you release the screen.

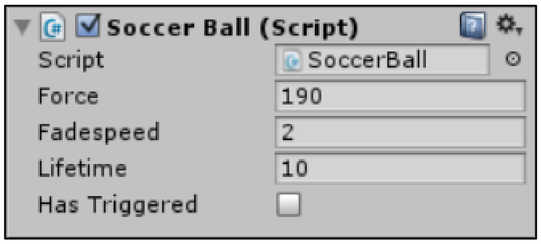


To score points in the basketball game, players throw balls in the hoop. The hoop has a script that adds score on trigger enter and instantiates an effect.

Soccer

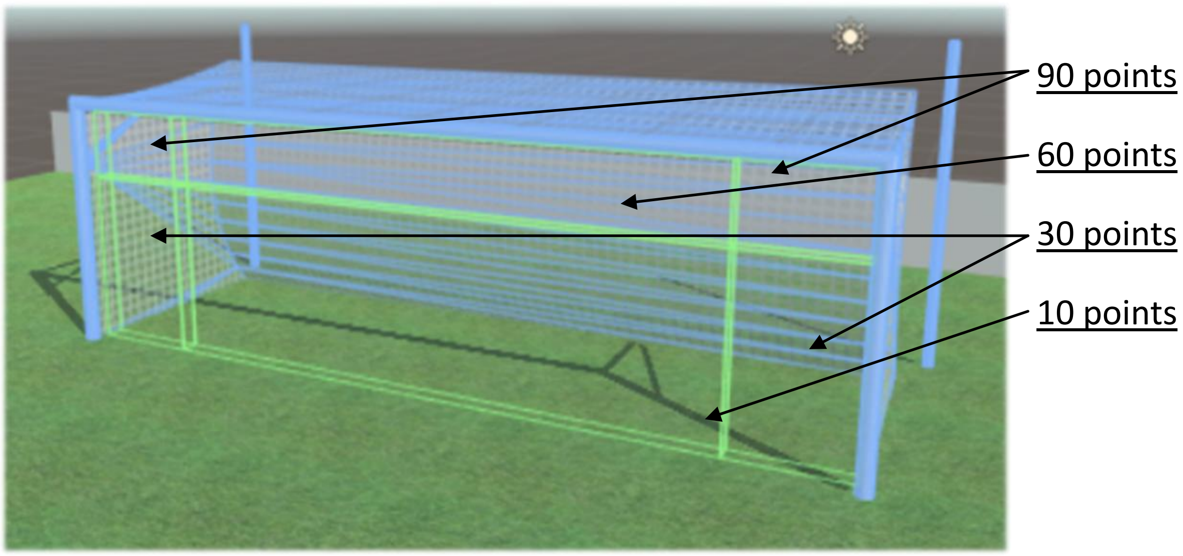
The soccer game looks a lot like the basketball one. However, the soccer game is a little bit more complex. Instead of always applying the same force and just changing the direction, the soccer game also bases force on swipe length. That's also the reason why you shoot when releasing the screen, because it needs to have an endpoint. The shooting force is not based on swipe speed, because that would simply become way too hard. Another more advanced feature of the soccer game is the scoring system. There's not just one trigger that adds points on trigger enter, there are four different types. If you shoot the ball in one of the upper corners, you'll receive 90 points, the lower corners are 30 points, in between the upper corners 60 points and anywhere else in the goal will give you 10 points. The balls bounce using a physics material and just like all other games, there's one main goal. Scoring as much points as possible by shooting the balls in the soccer goal.

The soccer game ends, just like the basketball game, after 60 seconds. Then the score gets saved and you can start again.

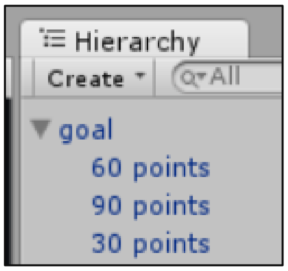

Soccer Ball (Script)

Script	SoccerBall
Force	190
Fadespeed	2
Lifetime	10
Has Triggered	☐

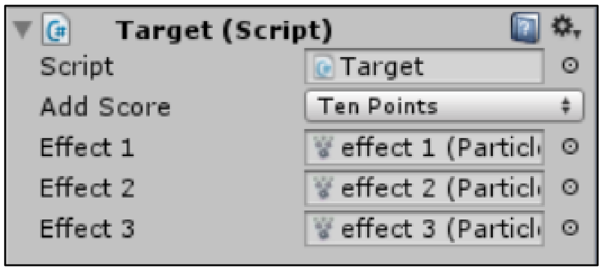
If we take a look at the soccer ball script, which handles swipe shooting and scoring, there's one very important setting. The 'has triggered' Boolean. You should always leave this unchecked. This simply tells the goals triggers whether this ball has added points already or not. That's pretty important; otherwise a ball could trigger one collider, bounce back and trigger another one. This tells the second collider that it has triggered a collider already. The force variable is the force multiplied by the swipe force. Fade speed is the speed with which the balls fade away after their lifetime. If you want the ball to shoot before you released the screen (looks more natural), you could use OnMouseExit or a coroutine.



There are 4 different triggers in the goal. Each one of them has a target script attached to it. You can select the amount of points for each target script. If you look at the goals hierarchy, you see the main object with three trigger object. Two of them with two colliders and two with one collider. As I said earlier, if you shoot the ball in one of the upper corners, you'll receive 90 points, the lower corners are 30 points, in between the upper corners 60 points and anywhere else in the goal will give you 10 points. The target script just checks for a trigger enter and then adds the right score. It also adds an effect so you can see how much points you have scored.



goal
60 points
90 points
30 points



Target (Script)

Script	Target
Add Score	Ten Points
Effect 1	effect 1 (Particl
Effect 2	effect 2 (Particl
Effect 3	effect 3 (Particl

Resetting the games

Resetting the games (deleting the high score) is quite easy. Find the ‘destroy playerprefs’ script, which is attached to the main camera. Toggle ‘destroy playerprefs’ on and press play. Then quit and toggle it off again. It has now deleted all playerprefs (data like highscore).

User interface

The scenes have a main script attached to the camera. This script handles the entire UI, but also things like spawning, score and difficulty. These scripts are necessary for the games. The scripts control UI with simple public functions and all UI buttons have onclick events referring to the main script. If you find any problems with buttons not working properly, please check for an EventSystem and then check the button’s onclick events.

Destroy particles

One very small yet important script is the destroy particles script. It removes particles (anything actually) from the scene after a certain amount of time. You can just set the time and use it anywhere. This way, we prevent the scenes from getting chockfull.

Saving high score

High scores are saved using int and float. In this case, when saving scores, it’s handier to use int, while float is better when saving time for example.
I used playerprefs and just saved the high score using the name of the game.