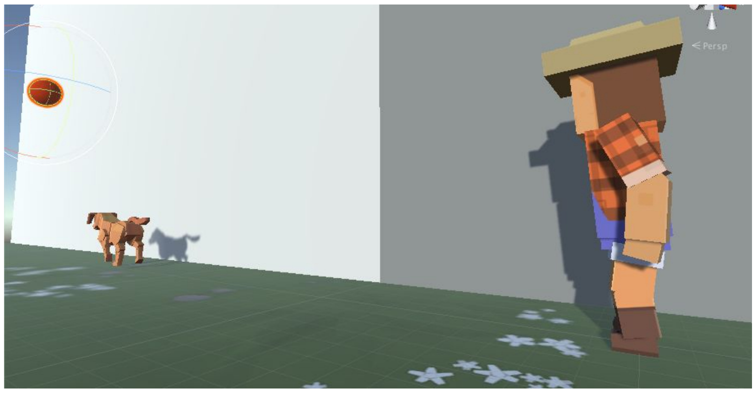
Challenge 2

Play Fetch



Challenge Overview: Use your array and random number generation skills to program this challenge where balls are randomly falling from the sky and you have to send your dog out to catch them before they hit the ground. To complete this challenge, you will have to make sure your variables are assigned properly, your if-statements are programmed correctly, your collisions are being detected perfectly, and that objects are being generated randomly.

Challenge Outcome:

- A random ball (of 3) is generated at a random x position above the screen
- When the user presses spacebar, a dog is spawned and runs to catch the ball
- If the dog collides with the ball, the ball is destroyed
- If the ball hits the ground, a “Game Over” debug message is displayed
- The dogs and balls are removed from the scene when they leave the screen

Challenge Objectives: In this challenge, you will reinforce the following skills/concepts:

- Assigning variables and arrays in the inspector
- Editing colliders to the appropriate size
- Testing xyz positions with greater/less than operators in if-else statements
- Randomly generating values and selecting objects from arrays

Challenge Instructions:

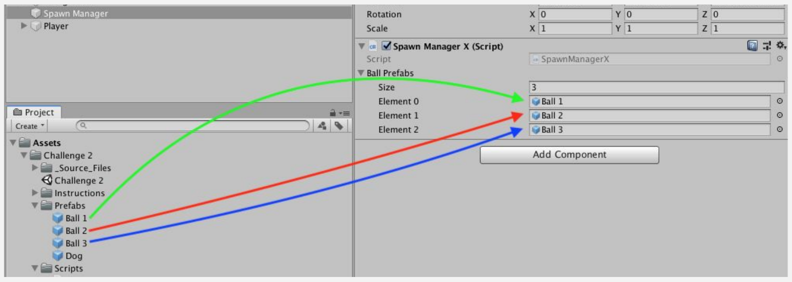
- Open your **Prototype 2** project
- **Download** the "Challenge 2 Starter Files" from the Tutorial Materials section, then double-click on it to **Import**
- In the *Project Window > Assets > Challenge 2 > **Instructions*** folder, use the "Challenge 2 - Instructions" and “Outcome” video as a guide to complete the challenge

Challenge		Task	Hint
1	Dogs are spawning at the top of the screen	Make the balls spawn from the top of the screen	Click on the Spawn Manager object and look at the “Ball Prefabs” array
2	The player is spawning green balls instead of dogs	Make the player spawn dogs	Click on the Player object and look at the “Dog Prefab” variable
3	The balls are destroyed if anywhere near the dog	The balls should only be destroyed when coming into direct contact with a dog	Check out the box collider on the dog prefab
4	Nothing is being destroyed off screen	Balls should be destroyed when they leave the bottom of the screen and dogs should be destroyed when they leave the left side of the screen	In the DestroyOutOfBounds script, double-check the lowerLimit and leftLimit variables, the greater than vs less than signs, and which position (x,y,z) is being tested
5	Only one type of ball is being spawned	Ball 1, 2, and 3 should be spawned randomly	In the SpawnRandomBall() method, you should declare a new random ***int index*** variable, then incorporate that variable into the Instantiate call

Bonus Challenge		Task	Hint
X	The spawn interval is always the same	Make the spawn interval a random value between 3 seconds and 5 seconds	Set the spawnInterval value to a new random number between 3 and 5 seconds in the SpawnRandomBall method
Y	The player can “spam” the spacebar key	Only allow the player to spawn a new dog after a certain amount of time has passed	Search for `Time.time` in the Unity Scripting API and look at the example. And don’t worry if you can’t figure it out - this is a *very difficult* challenge.

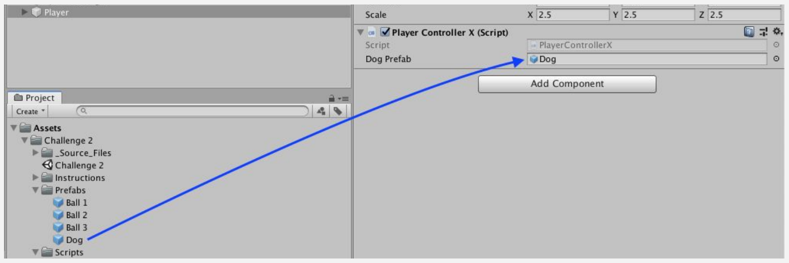
Challenge Solution

1 Select the Spawn Manager object and expand the “Ball Prefabs” array, then drag the **Ball 1, 2, 3** prefabs from *Assets > Challenge 2 > Prefabs* onto **Element 0, 1, 2**



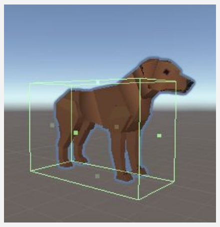
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Spawn Manager X (Script)
Script SpawnManagerX
Ball Prefabs
Size 3
Element 0 Ball 1
Element 1 Ball 2
Element 2 Ball 3

2 Select the Player object and drag the **Dog** prefab from *Assets > Challenge 2 > Prefabs* onto the “Dog Prefab” variable



Scale X 2.5 Y 2.5 Z 2.5
Player Controller X (Script)
Script PlayerControllerX
Dog Prefab Dog

3 Double-click on the Dog prefab, then in the Box Collider component, click **Edit Collider**, and reduce the collider to be the same size as the dog



4 In DestroyOutOfBoundsX.cs, make the leftLimit a negative value, change the greater than to a less than when testing the x position, and test the y value instead of the z for the bottom limit

```
private float leftLimit = -30;
private float bottomLimit = -5;

void Update() {
  if (transform.position.x ~~>~~ < leftLimit) {
    Destroy(gameObject);
  } else if (transform.position.~~z~~ y < bottomLimit) {
    Destroy(gameObject);
  }
}
```

5 In the SpawnRandomBall() method, declare a new random ***int index*** variable between 0 and the length of the Array, then incorporate that index variable into the the Instantiate call

```
void SpawnRandomBall ()
{
  // Generate random ball index and random spawn position
  int index = Random.Range(0, ballPrefabs.Length);
  Vector3 spawnPos = new Vector3(Random.Range(spawnXLeft, spawnXRight), spawnPosY, 0);

  // instantiate ball at random spawn location
  Instantiate(ballPrefabs[~~0~~ index], spawnPos, ballPrefabs[~~0~~ index].transform.rotation);
}
```

Bonus Challenge Solution

X1 In SpawnManagerX, the "InvokeRepeating" method will not work to accomplish this, since it is only capable of calling a single, unchanging method at a pre-set spawnInterval. Instead, we could use the simpler "Invoke" method (which does not specify a spawnInterval), and then in the in SpawnRandomBall() method, randomly reset ***startDelay*** using Random.Range() and re-call the SpawnRandomBall() method again from within the method itself.

```
~~private float spawnInterval = 4.0f;~~

void Start ()
{
  Invoke~~Repeating~~("SpawnRandomBall", startDelay~~, spawnInterval~~);
}

void SpawnRandomBall ()
{
  startDelay = Random.Range(3, 5);
  ...
  Invoke("SpawnRandomBall", startDelay);
}
```

Y1 In PlayerControllerX.cs, declare and initialize new fireRate and nextFire variables. Your "fireRate" will represent the time the player has to wait in seconds, and the nextFire variable will indicate the time (in seconds since the game started) at which the player will be able to fire again (starting at 0.0)

```
public GameObject dogPrefab;
private float fireRate = 1; // time the player has to wait to fire again
private float nextFire = 0; // time since start after which player can fire again
```

Y2 In the if-statement checking if the player pressed spacebar, add a new condition to check that Time.time (the time in seconds since the game started) is *greater* than nextFire (which represents the time after which the player is allowed to fire. If so, nextFire should be *reset* to the current time plus the fireRate.

```
// On spacebar press, if enough time has elapsed since last fire, send dog
if (Input.GetKeyDown(KeyCode.Space) && Time.time > nextFire)
{
  nextFire = Time.time + fireRate; // reset nextFire to current time + fireRate
  Instantiate(dogPrefab, transform.position, dogPrefab.transform.rotation);
}
```