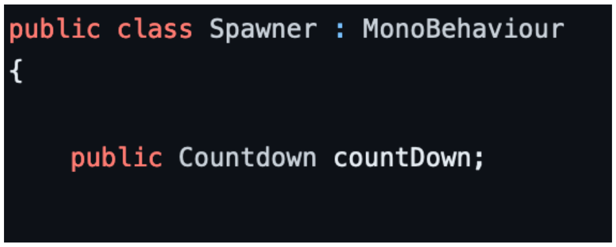
Crit C - Development

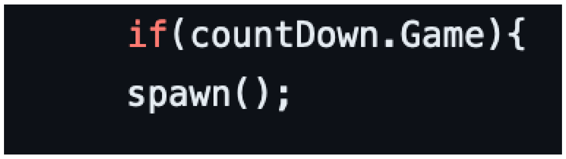
Table of Contents

1. Aggregation



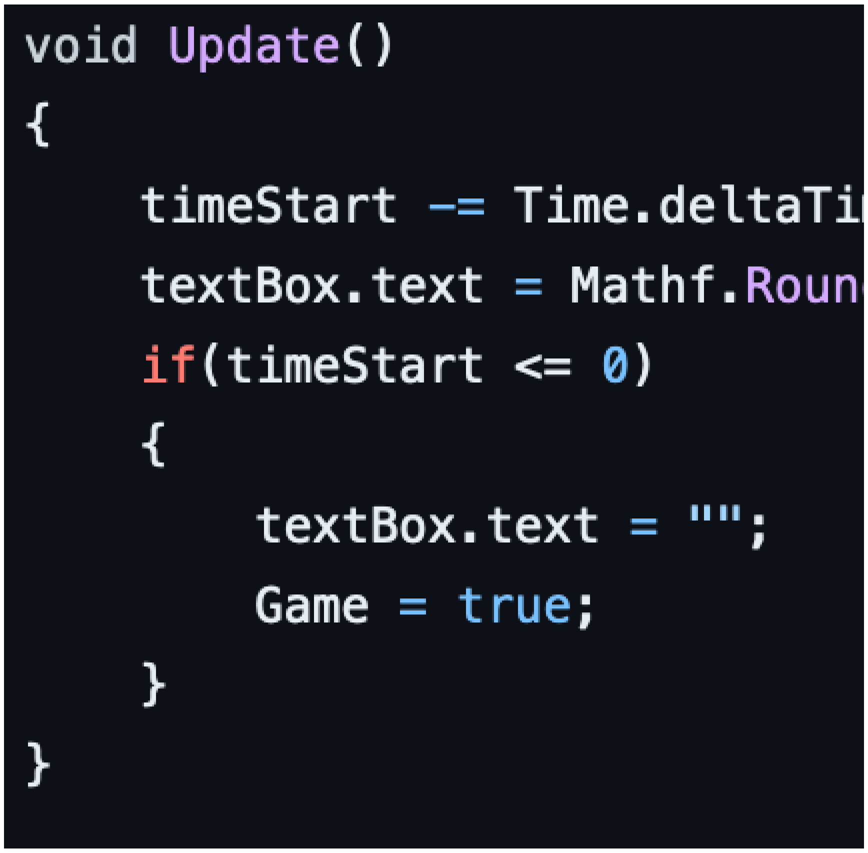
```
public class Spawner : MonoBehaviour
{

    public Countdown countDown;
```

Figure 1: The code that shows the relationship between the Countdown class and Spawner class. (has a relation)



```
        if(countDown.Game){
        spawn();
```

Figure 2: How accessing that class is needed in order to start the game.

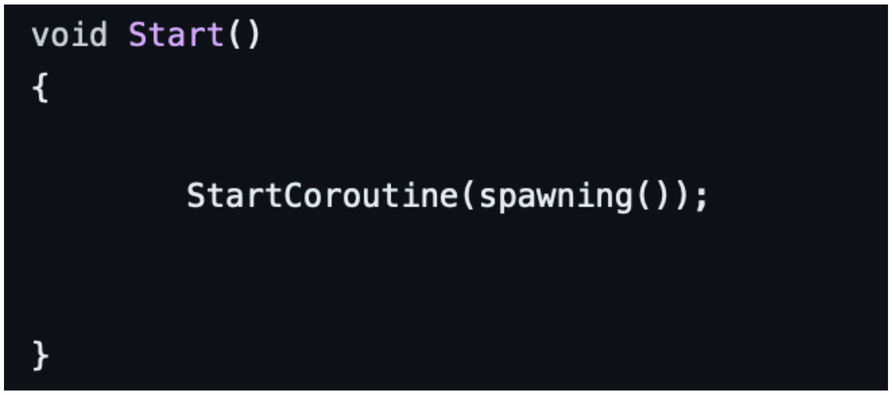


```
void Update()
{
    timeStart -= Time.deltaTi
    textBox.text = Mathf.Roun
    if(timeStart <= 0)
    {
        textBox.text = "";
        Game = true;
    }
}
```

Figure 3: The code that shows how it determines if the Game is “true” in Figure 2

The aggregation relationship signifies that one object belongs to another object and none other. That is, there exists a special kind of association between two objects, the parent and the child object, and the child object cannot belong to another parent object. (Dimitriou) In this case, the child class is the Spawner class and the parent class is the Countdown class. I need this technique because the timer in the Countdown class needs to hit 0 so the game could start with spawning green dots. The Figure 1 screenshot demonstrates how the parent class was involved with the child class. The Figure 2 screenshot shows the condition that has to be met before executing the spawn() function. In order to execute the function, the Game function in the Countdown class needs to be true and how the Game will become true is demonstrated in Figure 3. In Figure 3, we see that the if statement to check if the time start is less than 0 is in the Update() function. It means that the code will run automatically in each frame of the game. That is why it is possible to check if the timer is less than 0. So when the timer is less than 0, the Game is true, allowing the spawn() function to execute. This is why we need to use the "has a" relationship between the Countdown class and the Spawner class in order to execute the code if a certain condition is true from another class.

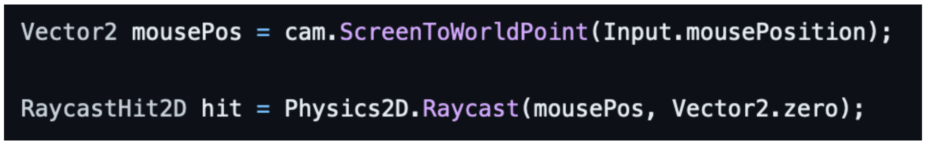
2. StartCoroutine



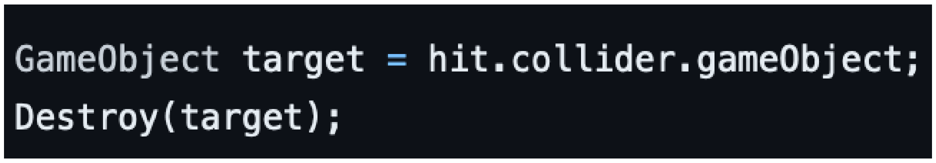
```
void Start()
{

        StartCoroutine(spawning());


}
```

In this code, StartCoroutine(spawning()) is called in the Start() method. This method starts the spawning() coroutine using StartCoroutine(). The spawning() coroutine is used to spawn new targets at random locations on the screen. A while loop is used to run the coroutine multiple times until the maximum number of spawns is reached. Inside the spawning() coroutine, yield return new WaitForSeconds(spawnTime) is used to introduce a delay between each spawn. This is used to control the spawning rate. StartCoroutine() is a coroutine execution function that executes a coroutine. The coroutine executes in parallel to the Update() method of the MonoBehaviour component it was called from. What I mean is that spawning targets is a time-consuming operation as it requires generating random coordinates, and checking collision and instantiating game objects. If this was done inside the Update() method, it can significantly reduce the frame rate and negatively impact game performance. But by using coroutines, the spawning() method is executed separately and does not affect the execution of Update(). I need this function in my game because my dots appear at some time that I set so there are an infinite amount of dots coming in the scene but some by some so the players can adapt to the game.

3. Collision Detection



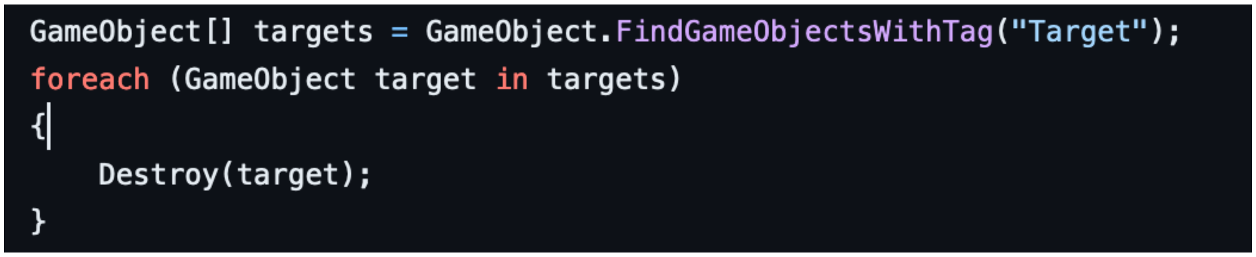
```
Vector2 mousePos = cam.ScreenToWorldPoint(Input.mousePosition);

RaycastHit2D hit = Physics2D.Raycast(mousePos, Vector2.zero);
```



```
GameObject target = hit.collider.gameObject;
Destroy(target);
```

This code is the basis of every collision with the dots. Vector2 mousePos = cam.ScreenToWorldPoint(Input.mousePosition); - First, this line of code gets the position of the mouse cursor from the screen coordinate system and converts it into the world coordinate system using the camera's ScreenToWorldPoint() method. After getting the mouse position, the RayCastHit2D object stores information about what has been hit by the raycast. This line of code casts a 2D ray originating from the mousePos position in the direction of (0, 0) and stores the result in a hit. With this code, we can basically create lives and scores function because we know if it's being hit or not.

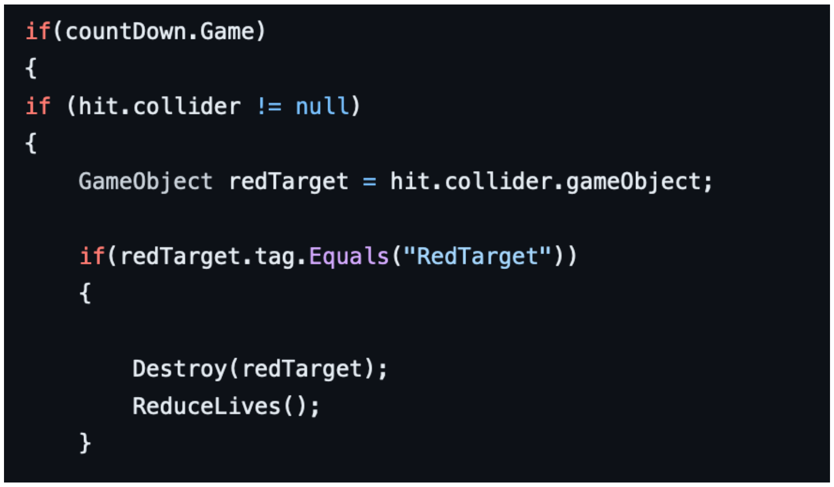
4. Arrays



```
GameObject[] targets = GameObject.FindGameObjectsWithTag("Target");
foreach (GameObject target in targets)
{
    Destroy(target);
}
```

Arrays are a common data structure used in programming to store collections of values. In the code above, an array is used to store a list of all the game objects that have the "Target" tag. This is done using the FindGameObjectsWithTag method of the GameObject class. FindGameObjectsWithTag returns an array of all the game objects that have the specified tag. In this case, the tag being searched for is "Target". The returned array is then stored in the targets variable, which is declared as a GameObject array. After finding all the GameObjects, I used a foreach loop to iterate over each game object in the targets array. For each game object, the Destroy method is called, which destroys the game object and removes it from the scene. I need to have an array to store a list of all the game objects beause I need to destroy everything in the list when the game is over.

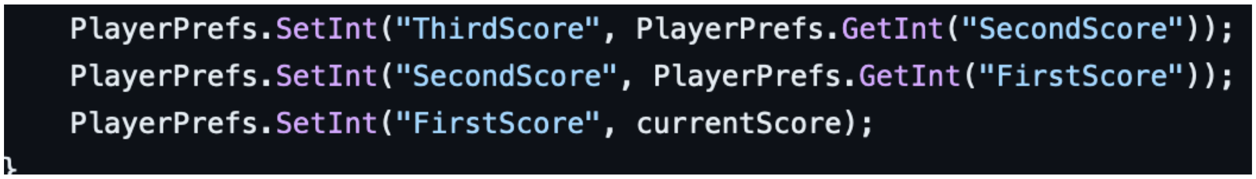
5. Conditional Algorithm



```
if(countDown.Game)
{
if (hit.collider != null)
{
    GameObject redTarget = hit.collider.gameObject;

    if(redTarget.tag.Equals("RedTarget"))
    {

        Destroy(redTarget);
        ReduceLives();
    }
```

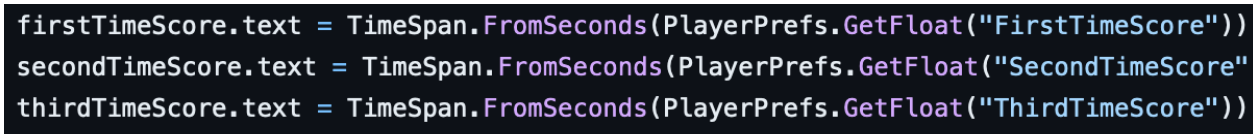
An algorithm is a procedure used for solving a problem or performing a computation. (Gillis) The algorithm used here computes more than one factor of cases. As seen in inheritance, when the countdown hits 0 and the Game turns to true, this algorithm is executed. This algorithm we see in the code is a conditional algorithm meaning that a decision is made between two courses of action. (Algorithms) We can use a repetitive conditional algorithm to check everything at once. We can execute planned actions based on the matching conditions. I need this algorithm to check if the user is clicking on the green dot, red dot or the screen in order to examine the score. Firstly it checks if the countDown.Game is true. If that condition is true, it moves on to the next one. If the player clicked on a dot, it sets the game object to redTarget and if the user did not hit any dots, it reduces the score because of clicking the screen. Then it checks if the tag of that dot is equal to “RedTarget”. If it is, it executes Detroy(redTarget) and ReduceLives() functions. If not, it sets the GameObject to be the target which is green dots and destroys it, and adds score. This is really important in my game because it increases my efficiency in the code. It allows me to check more conditions at once.

6. PlayerPrefs



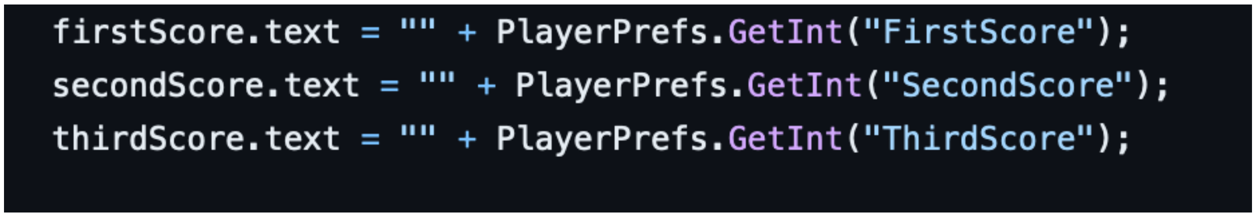
```
    PlayerPrefs.SetInt("ThirdScore", PlayerPrefs.GetInt("SecondScore"));
    PlayerPrefs.SetInt("SecondScore", PlayerPrefs.GetInt("FirstScore"));
    PlayerPrefs.SetInt("FirstScore", currentScore);
}
```

Figure 1: Changing the rank by Setting and Getting methods



```
firstTimeScore.text = TimeSpan.FromSeconds(PlayerPrefs.GetFloat("FirstTimeScore"))
secondTimeScore.text = TimeSpan.FromSeconds(PlayerPrefs.GetFloat("SecondTimeScore"
thirdTimeScore.text = TimeSpan.FromSeconds(PlayerPrefs.GetFloat("ThirdTimeScore"))
```

Figure 2: A screenshot that demonstrates getting float from the set value



```
firstScore.text = "" + PlayerPrefs.GetInt("FirstScore");
secondScore.text = "" + PlayerPrefs.GetInt("SecondScore");
thirdScore.text = "" + PlayerPrefs.GetInt("ThirdScore");
```

Figure 3: Putting the data into the GameObject in unity

PlayerPrefs is a class that stores Player preferences between game sessions. It can store string, float, and integer values into the user's platform registry. (Technologies) I used this class because I needed to work across the scenes and using PlayerPrefs class was the best way to set the high score in one scene and access that high score from another scene. Static methods such as SetFloat, SetInt, GetFloat, and SetFloat are used to create a ranking system in my game. SetFloat sets the float value of the preference identified by the given key and SetInt sets a single integer value for the preference identified by the given key. (Technologies). The GetFloat and GetInt return the value of the key when it is called. Not just getting the high score of the games from the users, but I actually made a ranking system where the players can see their top 3 high scores in the leaderboard. As shown in Figure 1, the third score is changed to the second score, and the second score is replaced with the first. I programmed it so that the highest score always remains at the top, just pushing down the scores like how the ranking system should work. Figure 2 demonstrates how GetFloat is used because it's in a different mode and that mode measures time which is a float value. Lastly, Figure 3 demonstrates putting those values in Unity Texts that I made. So PlayerPrefs was really helpful to me in terms of making a leaderboard so I could provide the users with their high scores.

Sources in MLA

"I Made Reflex Training Game - Unity Devlog." *YouTube*, 30 Apr. 2021, www.youtube.com/watch?v=Hicqg1Il6jY.

"Easy Heart Health System in Unity." *YouTube*, 30 Mar. 2022, www.youtube.com/watch?v=5NViMw-ALAo.

"Working with Time in Unity." *YouTube*, 10 Aug. 2018, www.youtube.com/watch?v=ijAN0QI70UU.

"Floating Text in Unity." *YouTube*, 15 Nov. 2020, www.youtube.com/watch?v=lKEKTWK9efE.

"Pair Matching Game - Unity Tutorial (Episode 2)." *YouTube*, 24 Mar. 2020, www.youtube.com/watch?v=7gxMVOs6rEs.

"Complete Unity Multiplayer Tutorial (Netcode for Game Objects)." *YouTube*, 26 Sept. 2022, www.youtube.com/watch?v=3yuBOB3VrCk.

"How to Make a High Score in Unity." *YouTube*, 29 Mar. 2017, www.youtube.com/watch?v=vZU51tbgMXk.

"Pop up Menu/Options Panel on Clicked Button in Unity." *YouTube*, 26 Sept. 2021, www.youtube.com/watch?v=7HAm_AgwwCg.

GlasDevGuy, and John French. "How to Use PlayerPrefs in Unity." *Game Dev Beginner*, 20 Oct. 2022, gamedevbeginner.com/how-to-use-player-prefs-in-unity/.

"What Is C# Programming? A Beginner's Guide." *Pluralsight*, 22 Nov. 2022, https://www.pluralsight.com/blog/software-development/everything-you-need-to-know-about-c-.

Dimitriou, Kostas, and Markos Hatzitaskos. *Core Computer Science: For the IB Diploma Program (International Baccalaureate)*. Express Publishing, 2018.

Gillis, Alexander S. “What Is an Algorithm? - Definition from Whatis.Com.” *WhatIs.Com*, 19 May 2022, www.techtarget.com/whatis/definition/algorithm.

Algorithms, www.cs.utexas.edu/users/mitra/csSpring2017/cs303/lectures/algo.html. Accessed 31 May 2023.

Technologies, Unity. “PlayerPrefs.” *Unity*, docs.unity3d.com/ScriptReference/PlayerPrefs.html. Accessed 31 May 2023.