Lab Week 5

50.033 Game Design and Development

1004147 --- Riley Riemann Chin

Are you participating in the Weekly Lab competition? ~~Yes~~/No

Provide the YouTube/other platform link to your screen recording:
https://youtu.be/y7zTKtf1AXw

Provide the link to your lab repository:
https://github.com/rileychin/50.033-Game-Dev-Labs/tree/lab5

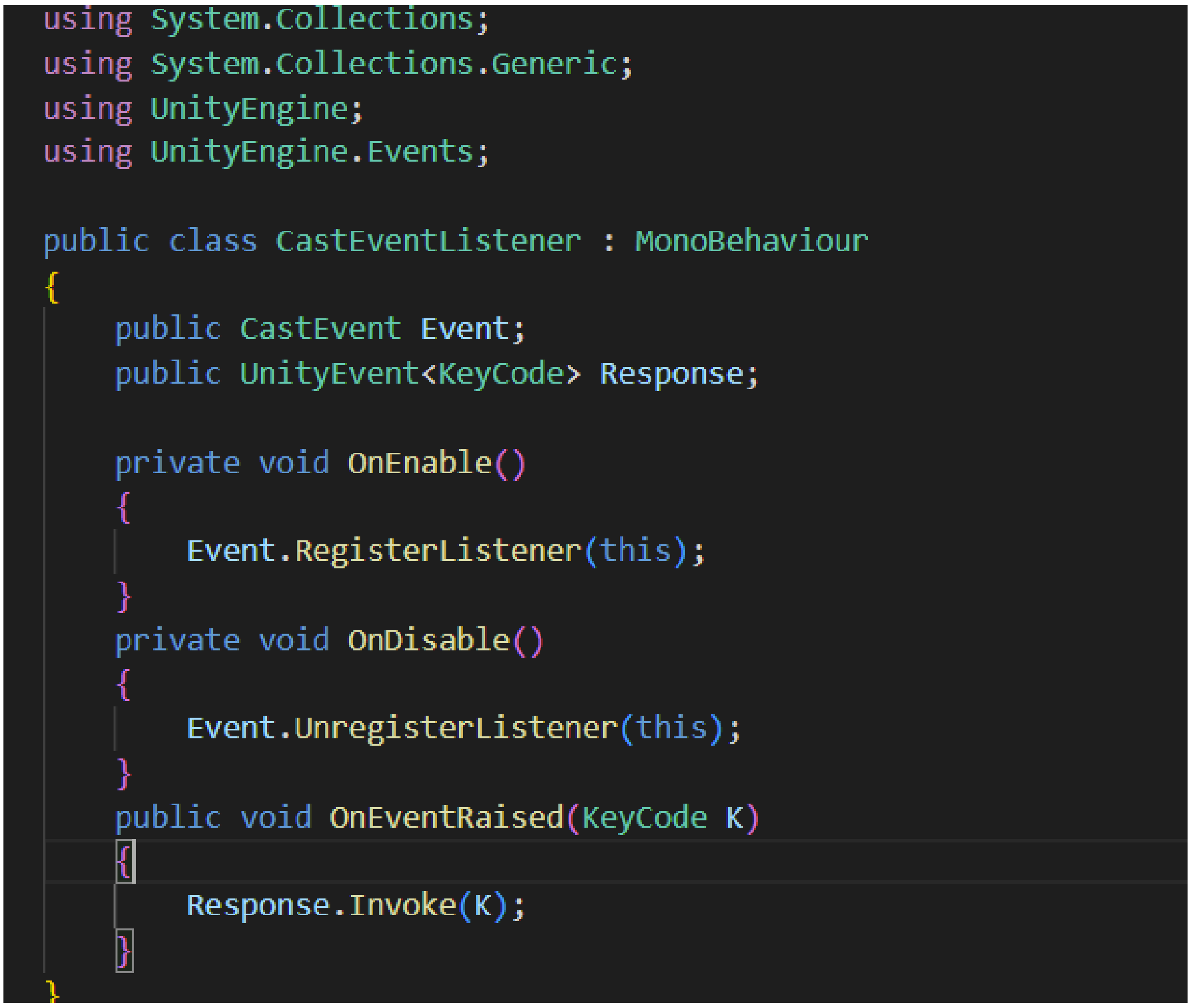
Describe what you have done to achieve the desired checkoff requirement for this lab:
[Your **high level** description here]

- You don't need to be too specific. This is just to assist us when we check your repository
- Scripts added, Assets added if any
- General modifications that you have done: eg animating the enemies, implementing FSM for the NPCs, etc.

Implement a Powerup Cast feature using the ScriptableObject Event system.

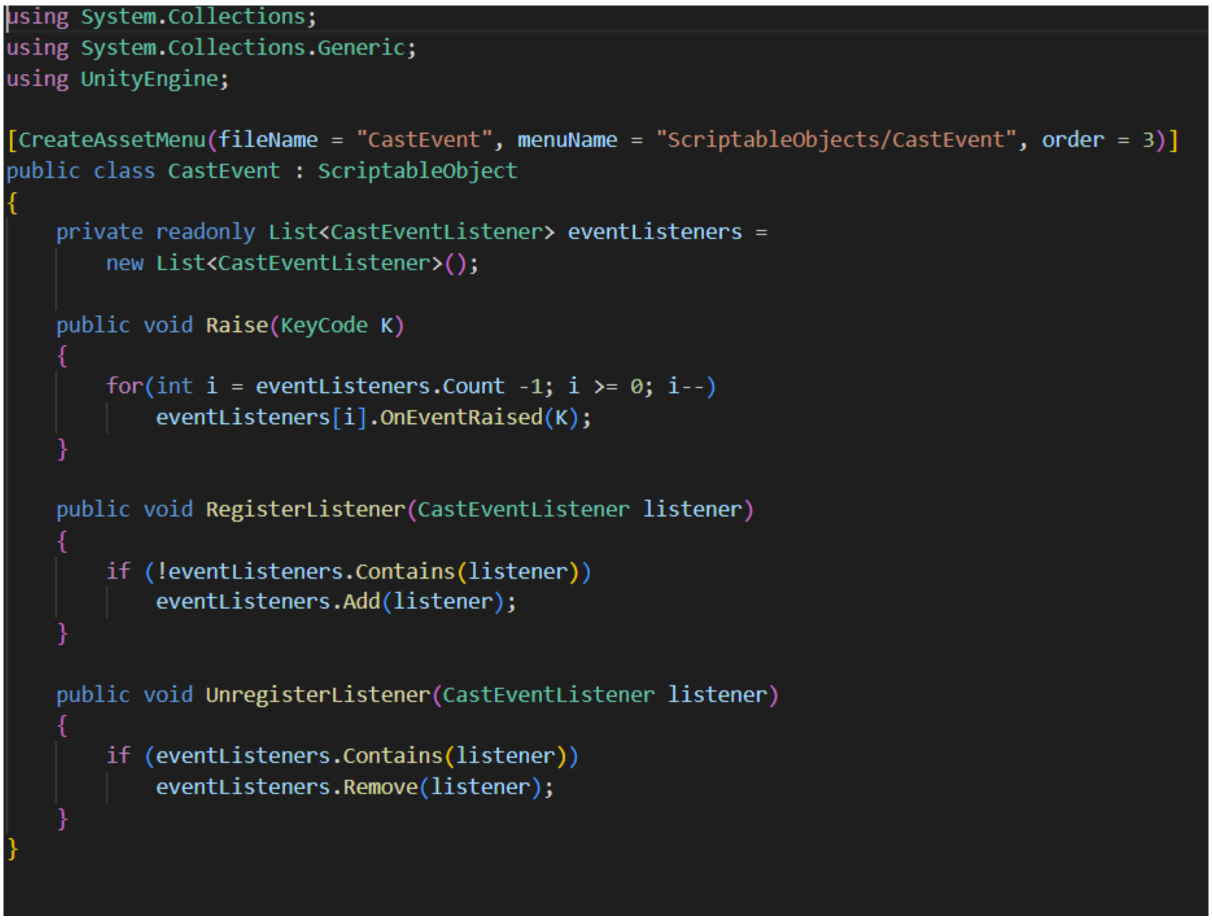
- When key Z is pressed, attempt to cast powerup in the first slot. Nothing should happen if you haven't collected anything there. Else, it must affect Mario's max or jump speed for a specific duration as dictated in the Powerup ScriptableObject instance.
- Similarly with key X, for the powerup in the second slot.
- You need to have at least two different powerups as per previous lab.

To implement the checkoff for the mario powerup, first I had to create CastEvent and CastEventListener.cs so that they will act as a new event for when a powerup is pressed.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;

public class CastEventListener : MonoBehaviour
{
    public CastEvent Event;
    public UnityEvent<KeyCode> Response;

    private void OnEnable()
    {
        Event.RegisterListener(this);
    }
    private void OnDisable()
    {
        Event.UnregisterListener(this);
    }
    public void OnEventRaised(KeyCode K)
    {
        Response.Invoke(K);
    }
}
```

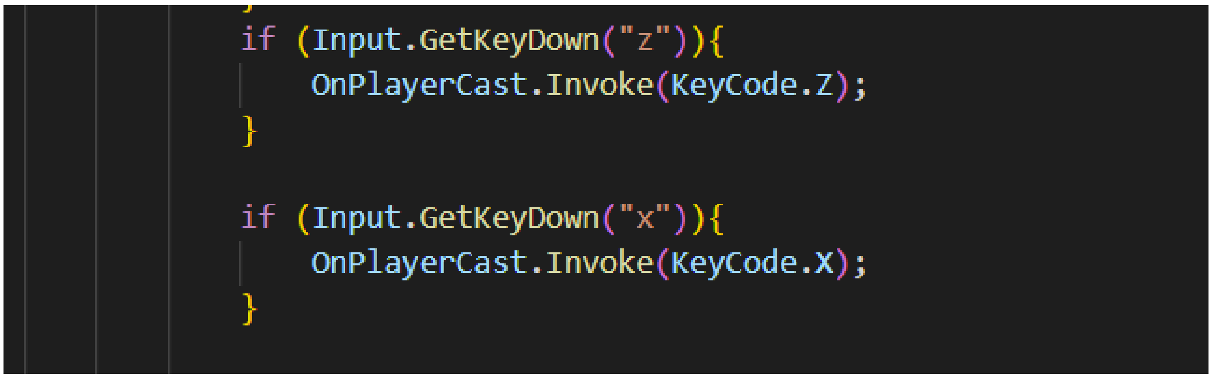
CastEventListener



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[CreateAssetMenu(fileName = "CastEvent", menuName = "ScriptableObjects/CastEvent", order = 3)]
public class CastEvent : ScriptableObject
{
    private readonly List<CastEventListener> eventListeners =
        new List<CastEventListener>();

    public void Raise(KeyCode K)
    {
        for(int i = eventListeners.Count -1; i >= 0; i--)
            eventListeners[i].OnEventRaised(K);
    }

    public void RegisterListener(CastEventListener listener)
    {
        if (!eventListeners.Contains(listener))
            eventListeners.Add(listener);
    }

    public void UnregisterListener(CastEventListener listener)
    {
        if (eventListeners.Contains(listener))
            eventListeners.Remove(listener);
    }
}
```

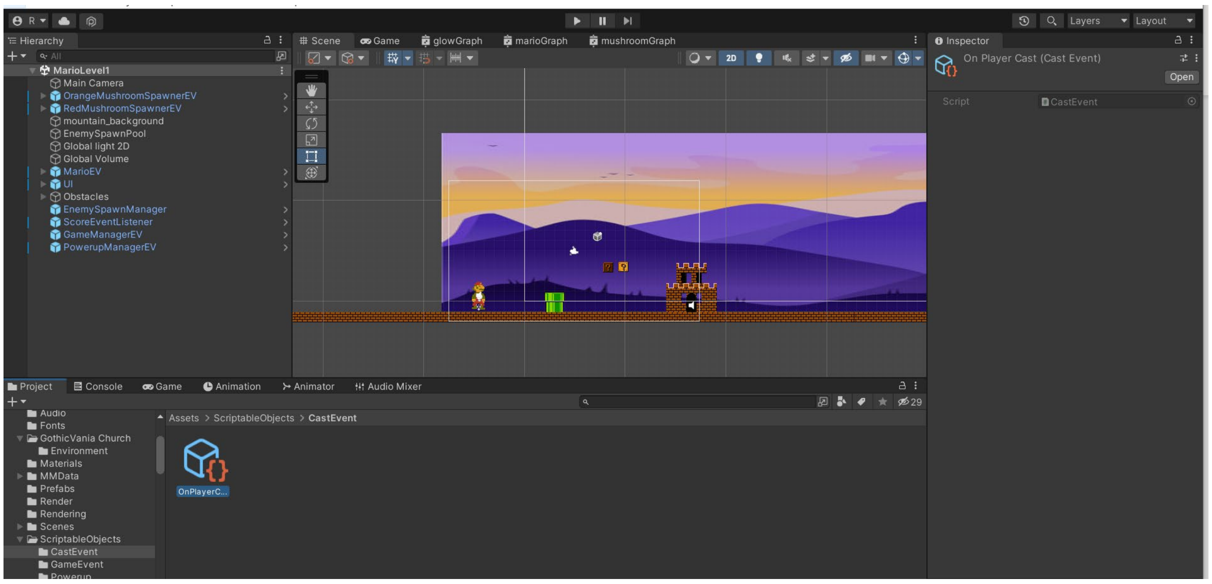
CastEvent

The raise and OnEventRaise accepts a keycode K argument for which key is pressed. Then, on PlayerController, I added an invoke for when the player pressed z or x so as to invoke the OnPlayerCast event.

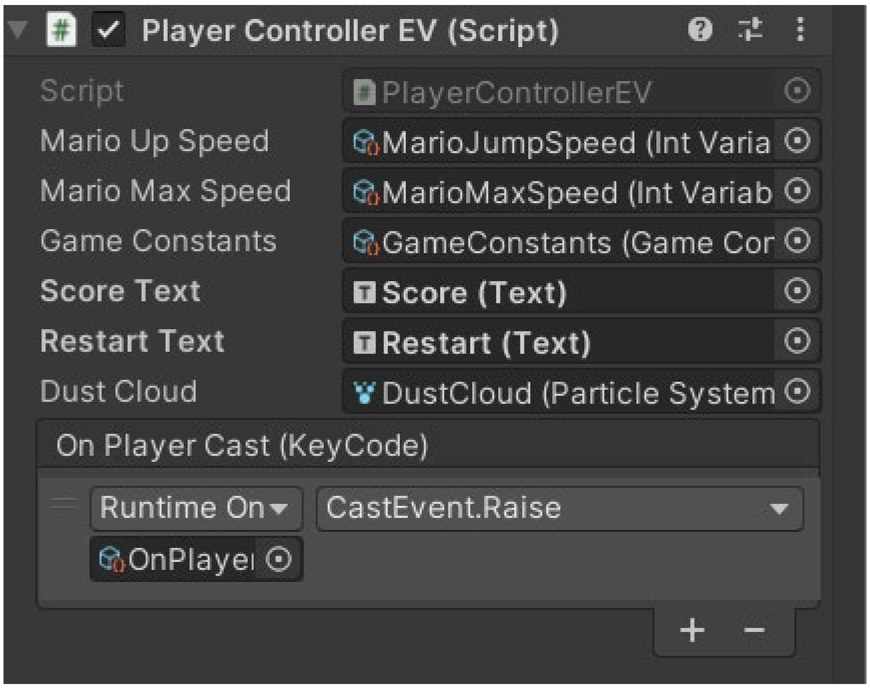


```
if (Input.GetKeyDown("z")){
    OnPlayerCast.Invoke(KeyCode.Z);
}

if (Input.GetKeyDown("x")){
    OnPlayerCast.Invoke(KeyCode.X);
}
```

OnPlayerCast is a new scriptableObject instance



Used to handle all the cast event when players click z or x.



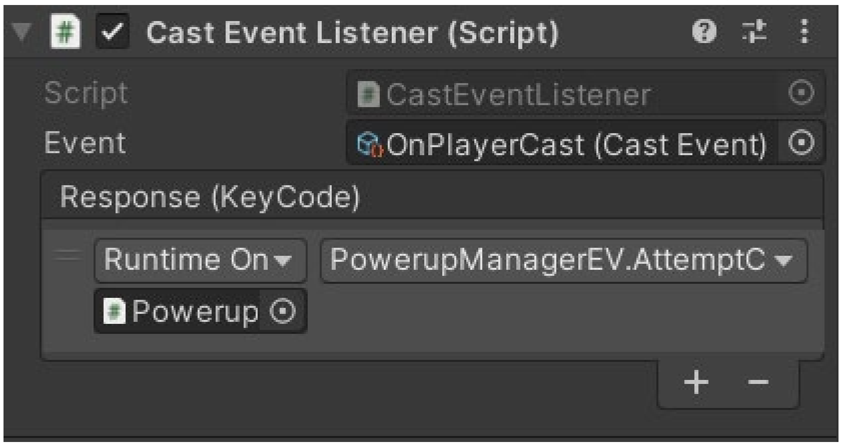
Player Controller EV (Script)

Script	PlayerControllerEV
Mario Up Speed	MarioJumpSpeed (Int Varia
Mario Max Speed	MarioMaxSpeed (Int Variab
Game Constants	GameConstants (Game Cor
Score Text	Score (Text)
Restart Text	Restart (Text)
Dust Cloud	DustCloud (Particle System

On Player Cast (KeyCode)

Runtime On | CastEvent.Raise
OnPlayer

We subscribe playercontroller EV to the onplayer cast when he clicks z or x, which will then be listened to by the casteventlistener in powerupmanager gameobject.



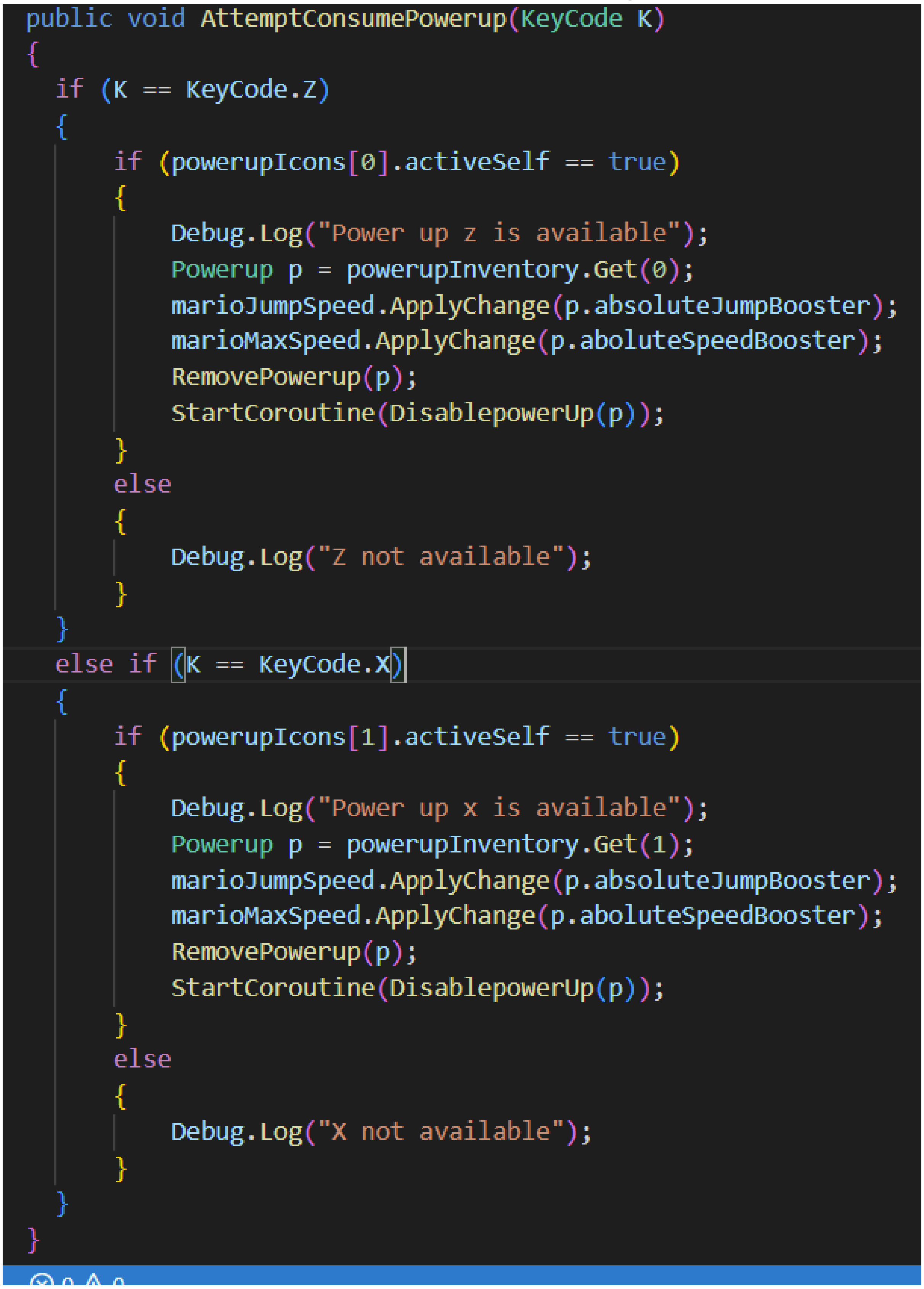
Cast Event Listener (Script)

Script	CastEventListener
Event	OnPlayerCast (Cast Event)

Response (KeyCode)

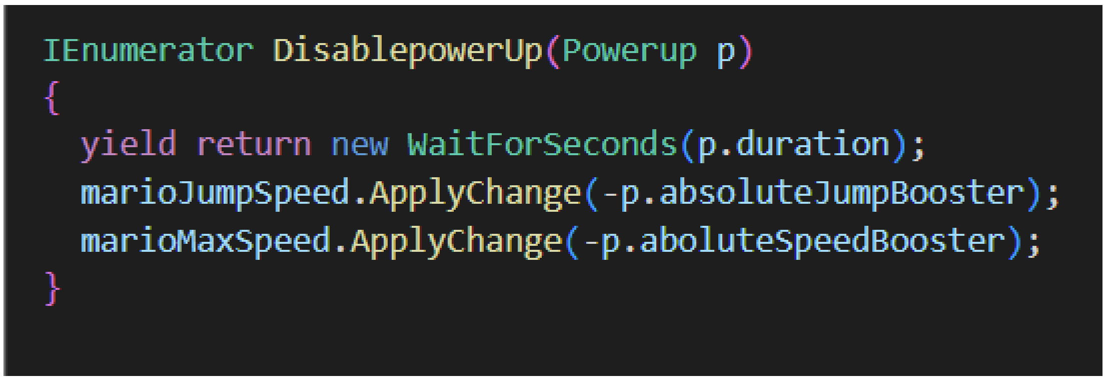
Runtime On | PowerupManagerEV.AttemptC
Powerup

Then it will call the AttemptConsumePowerup in powerupmanagerEV.



```
public void AttemptConsumePowerup(KeyCode K)
{
    if (K == KeyCode.Z)
    {
        if (powerupIcons[0].activeSelf == true)
        {
            Debug.Log("Power up z is available");
            Powerup p = powerupInventory.Get(0);
            marioJumpSpeed.ApplyChange(p.absoluteJumpBooster);
            marioMaxSpeed.ApplyChange(p.aboluteSpeedBooster);
            RemovePowerup(p);
            StartCoroutine(DisablepowerUp(p));
        }
        else
        {
            Debug.Log("Z not available");
        }
    }
    else if (K == KeyCode.X)
    {
        if (powerupIcons[1].activeSelf == true)
        {
            Debug.Log("Power up x is available");
            Powerup p = powerupInventory.Get(1);
            marioJumpSpeed.ApplyChange(p.absoluteJumpBooster);
            marioMaxSpeed.ApplyChange(p.aboluteSpeedBooster);
            RemovePowerup(p);
            StartCoroutine(DisablepowerUp(p));
        }
        else
        {
            Debug.Log("X not available");
        }
    }
}
```

Then the powerup will be disable after a few seconds



```
IEnumerator DisablepowerUp(Powerup p)
{
  yield return new WaitForSeconds(p.duration);
  marioJumpSpeed.ApplyChange(-p.absoluteJumpBooster);
  marioMaxSpeed.ApplyChange(-p.aboluteSpeedBooster);
}
```

Which will revert the changes for IntVariable for MarioJumpSpeed and MarioMaxSpeed!