The Gameplay Video link: https://youtu.be/avmcdFPdLs0

Rules:

Win Condition: Get 10+ Score and reach final to Win

Lost Condition: Drop out of ground or score<10 to Lost

Pick up items:

Green cube: +1

Red cube: -1

Blue cylinder: accelerate

Yellow cylinder: slowdown

Black capsule: bigger

Gold capsule: smaller

Red sphere: Final

Six signification feature enhancements

1. make the map longer, and change angles of grounds to make game more playable.
2. Add different kinds of cube to decrease score, change size or change speed.
3. Remove some walls, when player drop off from ground, player will lose.
4. Add background music and sounds for pick up.
5. Add some moving obstacles to increase difficulty of game.
6. Add jump action with input space.

How Code Works?

1. Player Controller

Firstly, declare all reference we need in player controller

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

public class PlayerController : MonoBehaviour
{
    //referance for speed and size
    public float speed;
    public Vector3 size;

    //count and win text
    public Text countText;
    public Text winText;

    //all audio source
    public AudioSource increaseScore;
    public AudioSource decreaseScore;
    public AudioSource accelerate;
    public AudioSource slowdown;
    public AudioSource jumpSound;
    public AudioSource bigger;
    public AudioSource smaller;
    public AudioSource victory;
    public AudioSource fail;

    //physics
    private Rigidbody rb;
    private int count;

    //replay button
    public Button replay;
```

Then, initial parameters when game start

```

    void Start()
    {
        //init
        rb = GetComponent<Rigidbody>();
        size = transform.GetComponent<Renderer>().bounds.size;
        speed = 10;
        count = 0;
        SetCountText();
        replay.gameObject.SetActive(false);
        winText.text = "";

    }
```

Keep updating lost condition when game is running.

```
//update lost condition
    void Update()
    {
        if(transform.position.y < -30.0f){
            Fail();
        }
    }
```

Catch keyboard input to move(up,down,left,right) and jump(space)

```
//move and jump
    void FixedUpdate()
    {
        float moveHorizontal = Input.GetAxis("Horizontal");
        float moveVertical = Input.GetAxis("Vertical");

        float jump;
        if(Input.GetKeyDown(KeyCode.Space))
        {
            jumpSound.Play();
            jump = 25f;
        }
        else
        {
            jump = 0;
        }
        Vector3 movement = new Vector3(moveHorizontal,jump,moveVertical);
        rb.AddForce(movement * speed);
    }
```

When catch up cube with "Pick Up" tag, score +1;
When catch up cube with "Decrease Score" tag, score-1;
When catch up cube with "Bigger" tag, player becomes bigger with size x,y,z doubled;
When catch up cube with "Smaller" tag, Player becomes smaller with size x,y,z * 0.6;
When catch up cube with "Accelerate" tag, speed * 1.5 times;
When catch up cube with "Slowdown" tag, speed * 0.6 times;
When catch up cube with "Final" tag, go to calculate result of game;
It will play different sounds for picking up different items.

```
//pick up items
    void OnTriggerEnter(Collider other) {
        //add score
        if(other.gameObject.CompareTag("Pick Up"))
        {
            other.gameObject.SetActive(false);
            increaseScore.Play();
            count = count + 1;
            SetCountText();
        }

        //minus score
```

```
       if(other.gameObject.CompareTag("Decrease Score"))
       {
          other.gameObject.SetActive(false);
          decreaseScore.Play();
          count = count - 1;
          SetCountText();
       }

       //bigger
        if(other.gameObject.CompareTag("Bigger"))
       {
          other.gameObject.SetActive(false);
          bigger.Play();
          Vector3 biggerSize = new Vector3 (size.x * 2, size.y * 2, size.z *
2);
          transform.localScale = biggerSize;
          size = biggerSize;
       }

       //smaller
       if(other.gameObject.CompareTag("Smaller"))
       {
          other.gameObject.SetActive(false);
          smaller.Play();
          Vector3 smallerSize = new Vector3 (size.x * 0.6f, size.y * 0.6f,
size.z * 0.6f);
          transform.localScale = smallerSize;
          size = smallerSize;
       }

       //accelerate
       if(other.gameObject.CompareTag("Accelerate"))
       {
          other.gameObject.SetActive(false);
          accelerate.Play();
          speed = speed * 1.5f;
       }

       //slowdown
       if(other.gameObject.CompareTag("Slowdown"))
       {
           other.gameObject.SetActive(false);
           slowdown.Play();
           speed = speed * 0.6f;
       }

       //final
       if(other.gameObject.CompareTag("Final"))
       {
           other.gameObject.SetActive(false);
           Final();
       }

   }
```

When player catch up “Final” tag sphere, if score >= 10, player win, otherwise, player lose. If player win, update text and play victory music.

```
//win or lose when reach final
    void Final()
    {
        if(count >= 10){
            victory.Play();
            replay.gameObject.SetActive(true);
            winText.text = "You Win!";
        }
        else if(count < 10)
        {
            Fail();
        }
    }
```

Function to update Win Text

```
//update count text
    void SetCountText()
    {
        countText.text = "Count: "+ count.ToString();

    }
```

When player lose, update text and play fail music.

```
//lose
    void Fail()
    {
        replay.gameObject.SetActive(true);
        fail.Play();
        winText.text = "You Failed!";
    }
```

2. Camera Controller

When game start, calculate offset between camera and player.
When game is running, keep updating position of camera.

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

public class CameraController : MonoBehaviour
{
    public GameObject player;

    private Vector3 offset;

    //calculate offset
    void Start()
    {
        offset = transform.position - player.transform.position;
    }

    ///keep camara going with player
    void LateUpdate()
    {
        transform.position = player.transform.position + offset;
    }
}
```

3. Replay Controller

When player click Replay button, use SceneManager to reload MiniGame Scene to restart game.

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

```
public class ReplayController : MonoBehaviour
{
    public void Restart(){
        SceneManager.LoadScene("MiniGame");
    }
}
```

4. Rotator

Use transform to keep updating items’ rotation angles to make items rotate.

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

public class Rotator : MonoBehaviour
{

    // keep all pick up items rotating
    void Update()
    {
        transform.Rotate(new Vector3(15,30,45) * Time.deltaTime);
    }
}
```

5. Obstacle Move

Use a time and a speed to record obstacle move speed and transform time of game. Obstacles will change position ever 0.1f and change direction ever 8.0f

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

public class ObstacleMove : MonoBehaviour
{
    public float Speed;
    private float Time;
    void Start()
    {
        Speed = 0.1f;
        Time = 0.0f;
    }
    // Update is called once per frame
    void Update()
    {
        Time = Time + 0.1f;
        transform.Translate(Vector3.right * Speed);

        if (Time > 8.0f)
        {
            transform.Rotate(0, 180, 0);
            Time = 0.0f;
        }
    }
}
```