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

namespace teamFourFinalProject
{
   [CreateAssetMenu(fileName = "CardPlatformPowerup", menuName =
"Platformer/Powerups/CardPlatform")]
   public class CardPowerup : PowerupData
   {
      public GameObject redPlat;
      public GameObject blackPlat;

      public GameObject ghostPlat;
      private GameObject activeGhost;

      public LayerMask wallLayerMask;

      public float despawnTimer = 5.0f;
      public float throwCooldown = 0.5f;
      public float spawnDistance = 3f;
      public int cardVal = 0;

      public override void ApplyEffects(PlayerController player)
      {
         Debug.Log("Applying Card Powerup effects");
         var groundChecker = player.GetComponent<GroundChecker>();
         var healthManager = player.GetComponent<HealthManager>();

         if (groundChecker == null || healthManager == null)
         {
            Debug.LogWarning("Missing components on player");
            return;
         }

         if (cardVal == 0)
         {
            healthManager.curHealth += 1;
            player.tempHealth = (int)healthManager.curHealth;
            player.upgradeAppl = true;
         }

         else if (cardVal == 1)
```

```
    {
        //player.changeMoveSpeed(3);
        player.upgradeAppl = true;
    }

    Debug.Log("CardPlatform powerup applied");
}

public override void RemoveEffects(PlayerController player)
{
    var groundChecker = player.GetComponent<GroundChecker>();
    var healthManager = player.GetComponent<HealthManager>();

    if (groundChecker == null || healthManager == null)
    {
        Debug.LogWarning("Missing components on player");
        return;
    }

    if (player.upgradeAppl)
    {
        player.upgradeAppl = false;

        if (cardVal == 0 && healthManager.curHealth == player.tempHealth)
        {
            healthManager.curHealth -= 1;
        }
        else if (cardVal == 1)
        {
            //player.changeMoveSpeed(-3);
        }
    }
    Debug.Log("CardPlatform powerup removed");
}

public void ThrowPlatform(PlayerController player)
{
    Camera cam = Camera.main;
    Ray ray = new Ray(cam.transform.position, cam.transform.forward);
    RaycastHit hit;

    // Raycast to find the wall or surface to stick to
    if (Physics.Raycast(ray, out hit, 10f, wallLayerMask))
    {
```

```
            // If we hit something (wall), adjust the spawn position
            Vector3 spawnPos = hit.point + hit.normal * 0.05f; // Small offset to avoid clipping
            Quaternion spawnRot = Quaternion.LookRotation(-hit.normal);

            // Instantiate the platform at the adjusted position
            GameObject platToSpawn = cardVal == 0 ? redPlat : blackPlat;
            GameObject platform = Instantiate(platToSpawn, spawnPos, spawnRot);
            Destroy(platform, despawnTimer);

            Debug.Log("Platform thrown and aligned to wall");
        }
        else
        {
            // If no wall hit, spawn the platform in front of the player, further away
            Vector3 flatForward = cam.transform.forward;
            flatForward.y = 0f;  // Keep it on the ground
            flatForward.Normalize();

            // Move the spawn position further away based on the spawnDistance
            Vector3 spawnPos = player.transform.position + flatForward * spawnDistance;
            spawnPos.y -= 0.5f; //Adjusts platform position!

            Quaternion spawnRot = Quaternion.LookRotation(flatForward);

            // Instantiate the platform
            GameObject platToSpawn = cardVal == 0 ? redPlat : blackPlat;
            GameObject platform = Instantiate(platToSpawn, spawnPos, spawnRot);
            Destroy(platform, despawnTimer);

            Debug.Log("Platform thrown and aligned to flat ground");
        }
    }

    public void UpdateGhost(PlayerController player)
    {
        Camera cam = Camera.main;
        Ray ray = new Ray(cam.transform.position, cam.transform.forward);
        RaycastHit hit;

        Vector3 ghostPos;
        Quaternion ghostRot;

        if (Physics.Raycast(ray, out hit, 10f, wallLayerMask))
        {
```

```
            // Align to wall surface
            ghostPos = hit.point + hit.normal * 0.05f;
            ghostRot = Quaternion.LookRotation(-hit.normal);
        }
        else
        {
            // Align in front of player, slightly offset and on flat ground
            Vector3 flatForward = cam.transform.forward;
            flatForward.y = 0f;
            flatForward.Normalize();

            ghostPos = player.transform.position + flatForward * spawnDistance;
            ghostPos.y -= 0.5f; //Adjusts ghost platform position!
            ghostRot = Quaternion.LookRotation(flatForward);
        }

        if (activeGhost == null)
        {
            GameObject platToUse = cardVal == 0 ? redPlat : blackPlat;
            activeGhost = GameObject.Instantiate(platToUse);
            SetGhostAppearance(activeGhost);
        }

        activeGhost.transform.position = ghostPos;
        activeGhost.transform.rotation = ghostRot;

        Debug.DrawRay(ray.origin, ray.direction * 10f, Color.cyan, 1f);
    }

    public void HideGhostPreview()
    {
        if (activeGhost != null)
        {
            GameObject.Destroy(activeGhost);
            activeGhost = null;
        }
    }

    private void SetGhostAppearance(GameObject ghost)
    {
        foreach (var renderer in ghost.GetComponentsInChildren<Renderer>())
        {
            renderer.material = Resources.Load<Material>("GhostMaterial");
        }
```

```
            foreach (var collider in ghost.GetComponentsInChildren<Collider>())
            {
                collider.enabled = false;
            }
        }
    }
}
```