GameManager.cs

```
using NUnit.Framework;
using System.Collections.Generic;
using TMPro;
using Unity.VisualScripting;
using UnityEngine;
using UnityEngine.UIElements;


public class GameManager : MonoBehaviour
{
   public int DiskLevel = 3;
   [SerializeField] List<GameObject> Disks;
   [SerializeField] List<Vector3> DiskPositions;
   [SerializeField] GameObject Picket3;
   [SerializeField] GameObject PanelWin;
   // Reference to the Timer script
   [SerializeField] private Timer gameTimer;


  // [SerializeField] private TextMeshProUGUI moveCounterText;  // Assign the UI Text in the
Inspector
   //private int moveCounter = 0;  // Tracks the number of moves



   private void Awake()
   {
      // Iterate through all child objects of the GameManager
```

```
    foreach (Transform item in transform)
    {
        Disks.Add(item.gameObject);// Add each child object(disk)
        DiskPositions.Add(item.transform.position);// Record the initial position of each disk
    }

    StartGame();
}

void StartGame()
{
   // gameTimer.ResetTimer();

    //moveCounter = 0;  // Reset move counter
    //UpdateMoveCounterUI();  // Immediately update the UI with the reset value


    int x = 0;
    for (int i = Disks.Count - 1; i >= 0; i--)// Iterate through the Disks list in reverse order.

    {
        if (x < DiskLevel)
        {
            Disks[i].SetActive(true);// Activate the disk (make it visible and functional)
            Disks[i].transform.SetParent(transform, false);// Set the parent of the disk to the current GameObject without modifying its world position.
```

```
            Disks[i].transform.position = DiskPositions[i];// Reset the disk's position to its initial starting position.


        }
        else
        {
            Disks[i].SetActive(false);
        }
        x++;// Increment the counter to track the number of disks processed.


    }
    PanelWin.SetActive(false);
}
/*
 public void IncrementMoveCounter()
 {
    moveCounter++;  // Increase the move counter
    UpdateMoveCounterUI();  // Update the UI text to reflect the new value
 }
 */


 public void IsWinner()
 {
    //panel win screen appears when you win(all disks on 3rd picket)
    if (Picket3.transform.childCount == DiskLevel)
    {
```

```
            PanelWin.SetActive(true);
            // gameTimer.StopTimer();//stops timer

        }
    }

    public void NextLevel()
    {
        //adds disk for the next level
        DiskLevel++;
        StartGame();
        // gameTimer.ResetTimer(); //resets timer when next is clicked
    }

    public void ExitGame()
    {
        //quits application
        Application.Quit();
    }
}
```

MoveDisk.cs

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

public class MoveDisk : MonoBehaviour
{
   private Vector3 screenPoint, offset;
   private bool endDrag = false;
   private Vector3 startPosition;
   [SerializeField] GameManager gameManager; // Reference to the GameManager script to invoke game logic

   private void Start()
   {
      // Sets custom gravity strength for the disks
      Physics.gravity = new Vector3(0f, -50f, 0f);
   }

   private void OnMouseDown()
   {
      // Triggered when the player clicks on the disk to drag it
      if (!canMove()) return; // Ensures only the topmost disk can be moved
      startPosition = transform.position; // Save the current position as the start position
      endDrag = false; // Reset the drag completion flag
```

```
    DiskIsKinematic(true); // Temporarily disable physics interaction for the disk and siblings
    screenPoint = Camera.main.WorldToScreenPoint(transform.position); // Get the screen position of the disk
    offset = transform.position - Camera.main.ScreenToWorldPoint(new Vector3(
      Input.mousePosition.x, Input.mousePosition.y, screenPoint.z)); // Calculate the offset from mouse position
  }

  private void OnMouseDrag()
  {
    // Triggered when the player is dragging the disk
    if (!canMove()) return; // Ensure only movable disks can be dragged
    Vector3 curScreenPoint = new Vector3(
      Input.mousePosition.x, Input.mousePosition.y, screenPoint.z); // Track the current mouse position
    Vector3 curPosition = Camera.main.ScreenToWorldPoint(curScreenPoint) + offset; // Calculate the new disk position
    transform.position = new Vector3(curPosition.x, curPosition.y, transform.position.z); // Update disk position
  }

  private void OnMouseUp()
  {
    // Triggered when the player releases the mouse after dragging the disk
    endDrag = true; // Mark drag operation as completed
    DiskIsKinematic(false); // Re-enable physics for the disk and siblings
  }
```

```
  private void DiskIsKinematic(bool enabled)
  {
    // Toggles kinematic state for all sibling disks under the same parent
    foreach (Transform item in transform.parent)
    {
      item.GetComponent<Rigidbody>().isKinematic = enabled; // Enable/disable kinematic state for each disk
    }
  }

  private void OnTriggerEnter(Collider other)
  {
    // Triggered when the disk collides with another object, like a peg
    if (!endDrag || !canMove()) return; // Ensure collision handling only occurs after dragging is completed

    Transform trPicket = other.transform;
    int currentNumber = int.Parse(gameObject.name);
    // Move validation: Ensure disks are only stacked in ascending order
    foreach (Transform item in trPicket)
    {
      if (currentNumber < int.Parse(item.gameObject.name)) // Check if the current disk is larger than the existing disk
      {
        transform.position = startPosition; // Reset position if move is invalid
        return; // Abort the move
```

```
        }
    }

    // Valid move: Change the disk's parent to the new peg
    transform.SetParent(other.transform); // Assign the disk to the peg as its new parent
    transform.localPosition = new Vector3(0, transform.localPosition.y, 0); // Set position relative to the new parent
    gameManager.IsWinner(); // Check if the win condition is satisfied
  }

  private bool canMove()
  {
    // Determines whether the disk can be moved (only the topmost disk can be moved)
    int currentNumber = int.Parse(gameObject.name); // Get the numeric value of the current disk

    foreach (Transform item in transform.parent)
    {
      if (currentNumber < int.Parse(item.gameObject.name)) // Check if any disk above is smaller
      {
        return false; // Block movement for disks below the topmost one
      }
    }

    return true;
  }
```

```
}
```

Timer.cs

```
using UnityEngine;
using TMPro;

public class Timer : MonoBehaviour
{
   [SerializeField] private TextMeshProUGUI timerText;
   private float elapsedTime;
   private bool isRunning = true;  // Flag to control the timer

   void Update()
   {
      if (isRunning)
      {
         elapsedTime += Time.deltaTime;
         int minutes = Mathf.FloorToInt(elapsedTime / 60);
         int seconds = Mathf.FloorToInt(elapsedTime % 60);

         timerText.text = string.Format("Timer: {0:00}:{1:00}", minutes, seconds);
      }
```

```
    }

    //method to stop the timer
    public void StopTimer()
    {
        isRunning = false;
    }

    //method to reset and restart the timer
    public void ResetTimer()
    {
        elapsedTime = 0f;
        isRunning = true;
        timerText.text = "Timer: 00:00";
    }
}
```