SAC LAB

Exp1. Write a C# script for 2D character controller with jump functionality.

Aim: Create a script for orthographic character locomotion along with jump mechanism.

```
public class PlayerController : MonoBehaviour
{
public float moveSpeed;
public Rigidbody2D theRB;
public float jumpForce;
public Transform groundCheckpoint;
private bool onGround;
public LayerMask whatIsGround;
private Animator anim;
private SpriteRenderer theSR;
   // Start is called before the first frame update
void Start()
   {
anim = GetComponent<Animator>();
theSR = GetComponent<SpriteRenderer>();
   }

   // Update is called once per frame
void Update()
   {
         theRB.velocity = new Vector2 (moveSpeed * Input.GetAxisRaw("Horizontal"),
         theRB.velocity.y);
         onGround = Physics2D.OverlapCircle(groundCheckpoint.position, 0.2f,
whatIsGround);
if(Input.GetButtonDown("Jump"))
      {
if(onGround)
         {
theRB.velocity = new Vector2(theRB.velocity.x, jumpForce);
         }
      }
if(theRB.velocity.x< 0)
      {
theSR.flipX = true;
      }
else if(theRB.velocity.x> 0)
      {
theSR.flipX = false;
      }
anim.SetFloat("moveSpeed", Mathf.Abs(theRB.velocity.x));
anim.SetBool("onGround", onGround);
```

```
    }
}
```

Exp2. Write a simple C# script for Third person character controller (3D).

Aim: Create a 3D character controller with 360 degree rotation feature and locomotion.

```
using UnityEngine;
using System.Collections;
using UnityStandardAssets.CrossPlatformInput;

public class CharController : MonoBehaviour
{
        private Animator ThisAnimator = null;

        private int VertHash = Animator.StringToHash("Vertical");
        private int HorzHash = Animator.StringToHash("Horizontal");

        // Use this for initialization
        void Awake ()
        {
                ThisAnimator = GetComponent<Animator>();
        }

        // Update is called once per frame
        void Update ()
        {
                floatHorz = CrossPlatformInputManager.GetAxis("Horizontal");
                floatVert = CrossPlatformInputManager.GetAxis("Vertical");

                ThisAnimator.SetFloat(HorzHash, Horz, 0.1f, Time.deltaTime);
                ThisAnimator.SetFloat(VertHash, Vert, 0.1f, Time.deltaTime);
        }
}
```

Exp3. Write a C# script for player Health system.

Aim: Create a script for player health using current health and maximum health.

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

public class PlayerHealth : MonoBehaviour
{
Public static PlayerHealth instance;
```

```
Public int currentHealth, maxHealth = 6;
Private SpriteRenderer theSR;

Private void Awake()
    {
instance = this;
    }
void Start()
    {
currentHealth = maxHealth;
theSR = GetComponent<SpriteRenderer>();
    }

Public void DealDamage()
    {
currentHealth--;
if(currentHealth<= 0)
        {
gameObject.SetActive(false);
        }

UIController.instance.UpdateHealthDisplay();

    }

}
```

Exp4. Write a C# script for damaging and killing the player depending on his health.

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

public class KillPlayer : MonoBehaviour
{
private void OnTriggerEnter2D(Collider2D other)
    {
if(other.tag == "Player")
        {
LevelManager.instance.RespawnPlayer();
        }
    }
}
```

Exp 5. Write a c# script to Destroy game objects Over Time (Object pooling).

Aim: Write a script to destroy a GameObject within a specific Time.

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

public class DestroyOverTime : MonoBehaviour
{
public float lifeTime;
   // Start is called before the first frame update
void Start()
   {

   }

   // Update is called once per frame
void Update()
   {
      /* lifeTime -= Time.deltaTime;
if(lifeTime< 0)
       {
Destroy(gameObject);
       } */

Destroy(gameObject, lifeTime);
   }

}
```

Output:

The game object which was attached with this script will be destroyed in the amount of duration specified under lifeTime

Exp.6. Write a c# script for smooth camera follow mechanism.

Aim: Create smooth camera transition with follow target functionality.

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

public class CameraFollow : MonoBehaviour
{
public Transform target;

// Start is called before the first frame update
```

```
void Start()
    {

    }

// Update is called once per frame
void Update()
    {
        transform.position = newVector3(target.position.x, target.position.y,
transform.position.z);
    }
}
```

Exp7. Write a c# script for user interface containging player health which changes visually with respect to the player's current health.

Aim: create user interface controller to manage player health system visually.

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

public class UIController : MonoBehaviour
{
public static UIController instance;
public Image heart1, heart2, heart3;
public Sprite heartFull, heartHalf, heartEmpty;

private void Awake()
    {
instance = this;
    }

public void UpdateHealthDisplay()
    {
switch (PlayerHealth.instance.currentHealth)
        {
case 6:
            heart1.sprite = heartFull;
            heart2.sprite = heartFull;
            heart3.sprite = heartFull;

break;
```

```
case 5:
        heart1.sprite = heartFull;
        heart2.sprite = heartFull;
        heart3.sprite = heartHalf;

break;

case 4:
        heart1.sprite = heartFull;
        heart2.sprite = heartFull;
        heart3.sprite = heartEmpty;

break;

case 3:
        heart1.sprite = heartFull;
        heart2.sprite = heartHalf;
        heart3.sprite = heartEmpty;
break;

case 2:
        heart1.sprite = heartFull;
        heart2.sprite = heartEmpty;
        heart3.sprite = heartEmpty;
break;

case 1:
        heart1.sprite = heartHalf;
        heart2.sprite = heartEmpty;
        heart3.sprite = heartEmpty;
break;

case 0:
        heart1.sprite = heartEmpty;
        heart2.sprite = heartEmpty;
        heart3.sprite = heartEmpty;

break;

default:
        heart1.sprite = heartEmpty;
        heart2.sprite = heartEmpty;
        heart3.sprite = heartEmpty;

break;
```

```
    }
  }

}
```

Exp8. Write a c# script for accessing external attributes of multiple scripts as a level manager.

Aim: create a level manager script which manages various elements on HUD (Heads Up Display).

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

public class LevelManager : MonoBehaviour
{
public static LevelManager instance;
public float waitToRespawn;
public int gemsCollected;

private void Awake()
  {
instance = this;
  }

public void RespawnPlayer()
  {
StartCoroutine(RespawnCo());
  }

Private IEnumerator RespawnCo()
  {
Player.instance.gameObject.SetActive(false);
AudioManager.instance.PlaySFX(8);
yield return new WaitForSeconds(waitToRespawn);
Player.instance.gameObject.SetActive(true);
Player.instance.transform.position = CheckpointController.instance.spawnPoint;
PlayerHealth.instance.currentHealth = PlayerHealth.instance.maxHealth;
UIController.instance.UpdateHealthDisplay();
  }
}
```

Exp.9. Write a c# script for visual feedback of on/off of a checkpoint.

Aim: creating a visual representation of checkpoint toggle system.

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

```
using UnityEngine;

public class Checkpoint : MonoBehaviour
{
publicSpriteRenderertheSR;
public Sprite cpOn, cpOff;
    // Start is called before the first frame update
void Start()
    {

    }
    // Update is called once per frame
void Update()
    {

    }

private void OnTriggerEnter2D(Collider2D other)
    {
if(other.CompareTag("Player"))
        {
CheckpointController.instance.DeactiveCheckpoints();
theSR.sprite = cpOn;
CheckpointController.instance.SetSpawnPoint(transform.position);
        }

    }
public void ResetCheckpoint()
    {
theSR.sprite = cpOff;
    }

}
```

Exp. 10. Write a c# script for managing checkpoints using a dynamic array.

Aim: creating a checkpoint controller using dynamic array.

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

public class CheckpointController : MonoBehaviour
{
public static CheckpointController instance;
public Checkpoint[] checkpoints;
```

```
public Vector3 spawnPoint;
    // Start is called before the first frame update
private void Awake()
    {
instance = this;
    }

void Start()
    {
checkpoints = FindObjectsOfType<Checkpoint>();
spawnPoint = Player.instance.transform.position;
    }

    // Update is called once per frame
void Update()
    {

    }

public void DeactiveCheckpoints()
    {
for(int i=0; i<checkpoints.Length; i++)
        {
checkpoints[i].ResetCheckpoint();
        }
    }
public void SetSpawnPoint(Vector3 newSpawnPoint)
    {
spawnPoint = newSpawnPoint;
    }
}
```

Exp.11. Write a c# script to collect health and gem pickup objects.

Aim: creating a functionality for collecting two different pickup objects.

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

public class Pickup : MonoBehaviour
{
public bool isGem, isHeal;
private bool isCollected;
public GameObject pickupEffect;
    // Start is called before the first frame update
```

```
void Start()
    {

    }

    // Update is called once per frame
void Update()
    {

    }

private void OnTriggerEnter2D(Collider2D other)
    {
if(other.CompareTag("Player") && !isCollected)
        {
if(isGem)
            {
LevelManager.instance.gemsCollected++;
isCollected = true;
Destroy(gameObject);
Instantiate(pickupEffect, transform.position, transform.rotation);
UIController.instance.UpdateGemCount();
AudioManager.instance.PlaySFX(6);
            }
if(isHeal)
            {
if(PlayerHealth.instance.currentHealth != PlayerHealth.instance.maxHealth)
                {
PlayerHealth.instance.HealPlayer();
isCollected = true;
Destroy(gameObject);
Instantiate(pickupEffect, transform.position, transform.rotation);
AudioManager.instance.PlaySFX(7);
                }
            }
        }
    }
}
```

Exp.12 Write a c# script for managing audio using array in audio manager.

Aim: creating an audio manager script for managing all sound effects in game.

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

public class AudioManager : MonoBehaviour
{
public static AudioManager instance;
public AudioSource[] soundEffects;
public AudioSource bgm, levelEndMusic;

    // Start is called before the first frame update

private void Awake()
    {
instance = this;
    }
void Start()
    {

    }

    // Update is called once per frame
void Update()
    {

    }

public void PlaySFX(int soundToPlay)
    {
soundEffects[soundToPlay].Stop();
soundEffects[soundToPlay].pitch = Random.Range(0.9f, 1.1f);
soundEffects[soundToPlay].Play();

    }
}
```

Exp. 13 Write a c# script for constantly rotate the associated game object.

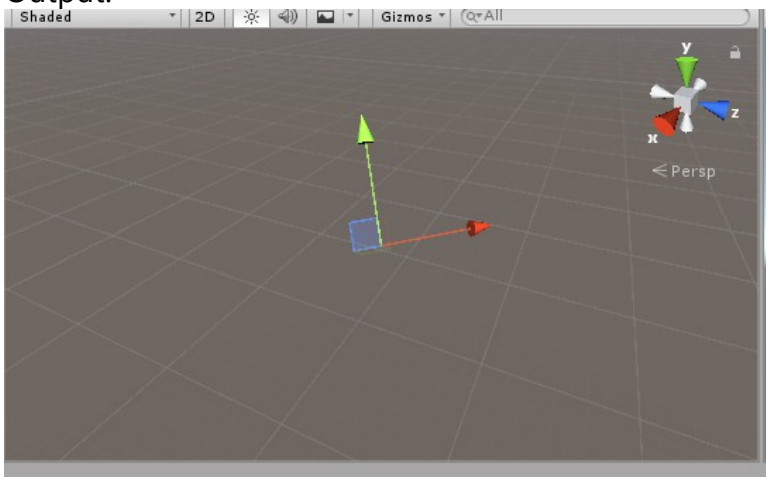
Aim: To rotate a 3D object with aligned main camera to showcase the output

```
using UnityEngine;
using System.Collections;

publicclassRotate : MonoBehaviour
{
publicfloatrotateSpeed;

        void Update ()
   {
transform.eulerAngles += Vector3.up * rotateSpeed * Time.deltaTime;
        }
}
```

Output:



Exp. 14. Write a c# script for Toggling a light switch.

Aim: create a on/off mechanism for light switch.

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

public class LightSwitch : MonoBehaviour {

        public bool onSwitch;
        public bool lightStatus;
        public GameObject theLight;

        voidOnTriggerEnter(Collider other)
        {
                onSwitch = true;
        }

        voidOnTriggerExit(Collider other)
```

```
{
	onSwitch = false;
}

void Update()
{
	if(theLight.active == true)
	{
		lightStatus = true;
	}
	else
	{
		lightStatus = false;
	}

	if (onSwitch)
	{
		if (lightStatus)
		{
			if (Input.GetKeyDown(KeyCode.E))
			{
				theLight.active = false;
			}
		}
		else
		{
			if (Input.GetKeyDown(KeyCode.E))
			{
				theLight.active = true;
			}
		}
	}
}
void OnGUI()
{
	if (onSwitch)
	{
		if (lightStatus)
		{
			GUI.Box(new Rect(0, 0, 200, 20), "Press E to close the light");

		}
		else
		{
			GUI.Box(new Rect(0, 0, 200, 20), "Press E to open the light");
```

```
                }
            }
        }
}
```

Exp. 15 Write a c# script for counting down the timer for round.

Aim: Creating a round timer script for counting the amount of duration player has to finish the level.

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

public class LapTimer : MonoBehaviour
{
        private string currentTime;
        private float totalTime = 60.0f;
        public bool startime = false;

        void Start()
        {

        }
        void Update ()
        {
                if(startime == true)
                {
                totalTime -= Time.deltaTime;
                currentTime = (Mathf.Floor(totalTime).ToString());
                }
        }
        void OnGUI()
        {
        GUIStyle Label2 = newGUIStyle(GUI.skin.GetStyle("label"));
        Label2.fontSize = 28;
        Label2.normal.textColor = Color.yellow ;
        GUI.Label(new Rect(50,40,200,200), "Lap Time : " +      currentTime, Label2);

        if(totalTime<= 1)
                {
                        SceneManager.LoadScene ("gameover");
                }
        }
```

```
}
```

Exp. 16 Write a c# script for assigning LOS (Line Of Sight) for an AI character.

Aim: Creating a visual perception for AI character using Line of sight mechanism.

```
using UnityEngine;
using System.Collections;
//---------------------------------------
public class LineSight : MonoBehaviour
{
	//---------------------------------------
	//How sensitive should we be to sight
	public enum SightSensitivity {STRICT, LOOSE};

	//Sight sensitivity
	Public SightSensitivitySensitity = SightSensitivity.STRICT;

	//Can we see target
	public bool CanSeeTarget = false;

	//FOV
	public float FieldOfView = 45f;

	//Reference to target
	public Transform Target = null;

	//Reference to eyes
	public Transform EyePoint = null;

	//Reference to transform component
	private Transform ThisTransform = null;

	//Reference to sphere collider
	Private SphereColliderThisCollider = null;

	//Reference to last know object sighting, if any
	public Vector3 LastKnowSighting = Vector3.zero;
	//---------------------------------------
	void Awake()
	{
		ThisTransform = GetComponent<Transform>();
		ThisCollider = GetComponent<SphereCollider>();
		LastKnowSighting = ThisTransform.position;
	}
	//---------------------------------------
```

```
Bool InFOV()
{
	//Get direction to target
	Vector3 DirToTarget = Target.position - EyePoint.position;

	//Get angle between forward and look direction
	float Angle = Vector3.Angle(EyePoint.forward, DirToTarget);

	//Are we within field of view?
	if(Angle <= FieldOfView)
		return true;

	//Not within view
	return false;
}
//----------------------------------------
Bool ClearLineofSight()
{
	RaycastHit Info;

	if(Physics.Raycast(EyePoint.position, (Target.position -
EyePoint.position).normalized, out Info, ThisCollider.radius))
	{
		//If player, then can see player
		if(Info.transform.CompareTag("Player"))
			return true;
	}

	return false;
}
//----------------------------------------
Void UpdateSight()
{
	switch(Sensitity)
	{
		Case SightSensitivity.STRICT:
			CanSeeTarget = InFOV() &&ClearLineofSight();
		break;

		case SightSensitivity.LOOSE:
			CanSeeTarget = InFOV() || ClearLineofSight();
		break;
	}
}
//----------------------------------------
```

```
        Void OnTriggerStay(Collider Other)
        {
                UpdateSight();

                //Update last known sighting
                if(CanSeeTarget)
                        LastKnowSighting=  Target.position;
        }
}
//---------------------------------------
```

Exp. 17. Write a c# script utilizing various player states for an AI character.

Aim: Create a script for AI character to patrol, chase and attack player.

```
using UnityEngine;
using System.Collections;
//---------------------------------------
public class AI_Enemy : MonoBehaviour
{
        //---------------------------------------
        public enum ENEMY_STATE {PATROL, CHASE, ATTACK};
        //---------------------------------------
        public ENEMY_STATE CurrentState
        {
                get{return currentstate;}

                set
                {
                        //Update current state
                        currentstate = value;

                        //Stop all running coroutines
                        StopAllCoroutines();

                        switch(currentstate)
                        {
                                case ENEMY_STATE.PATROL:
                                        StartCoroutine(AIPatrol());
                                break;

                                case ENEMY_STATE.CHASE:
                                        StartCoroutine(AIChase());
                                break;

                                case ENEMY_STATE.ATTACK:
```

```
                        StartCoroutine(AIAttack());
                    break;
                }
            }
}
//----------------------------------------
[SerializeField]
private ENEMY_STATE currentstate = ENEMY_STATE.PATROL;

//Reference to line of sight component
Private LineSightThisLineSight = null;

//Reference to nav mesh agent
Private UnityEngine.AI.NavMeshAgent ThisAgent = null;

//Reference to transform
private Transform ThisTransform = null;

//Reference to player health
public Health PlayerHealth = null;

//Reference to player transform
private Transform PlayerTransform = null;

//Reference to patrol destination
public Transform PatrolDestination = null;

//Damage amount per second
public float MaxDamage = 10f;
//----------------------------------------
void Awake()
{
        ThisLineSight = GetComponent<LineSight>();
        ThisAgent = GetComponent<UnityEngine.AI.NavMeshAgent>();
        ThisTransform = GetComponent<Transform>();
        PlayerTransform = PlayerHealth.GetComponent<Transform>();
}
//----------------------------------------
void Start()
{
        //Configure starting state
        CurrentState = ENEMY_STATE.PATROL;
}
//----------------------------------------
Public  IEnumeratorAIPatrol()
```

```
{
	//Loop while patrolling
	while(currentstate == ENEMY_STATE.PATROL)
	{
		//Set strict search
		ThisLineSight.Sensitity = LineSight.SightSensitivity.STRICT;

		//Chase to patrol position
		ThisAgent.Resume();
		ThisAgent.SetDestination(PatrolDestination.position);

		//Wait until path is computed
		while(ThisAgent.pathPending)
			yield return null;

		//If we can see the target then start chasing
		if(ThisLineSight.CanSeeTarget)
		{
			ThisAgent.Stop();
			CurrentState = ENEMY_STATE.CHASE;
			yield break;
		}

		//Wait until next frame
		yield return null;
	}
}
//---------------------------------------
Public IEnumeratorAIChase()
{
	//Loop while chasing
	while(currentstate == ENEMY_STATE.CHASE)
	{
		//Set loose search
		ThisLineSight.Sensitity = LineSight.SightSensitivity.LOOSE;

		//Chase to last known position
		ThisAgent.Resume();
		ThisAgent.SetDestination(ThisLineSight.LastKnowSighting);

		//Wait until path is computed
		while(ThisAgent.pathPending)
			yield return null;

		//Have we reached destination?
```

```
                if(ThisAgent.remainingDistance<= ThisAgent.stoppingDistance)
                {
                        //Stop agent
                        ThisAgent.Stop();

                        //Reached destination but cannot see player
                        if(!ThisLineSight.CanSeeTarget)
                                CurrentState = ENEMY_STATE.PATROL;
                        else //Reached destination and can see player. Reached
attacking distance
                                CurrentState = ENEMY_STATE.ATTACK;

                        yield break;
                }

                //Wait until next frame
                yield return null;
        }
    }
    //---------------------------------------
    Public IEnumeratorAIAttack()
    {
        //Loop while chasing and attacking
        while(currentstate == ENEMY_STATE.ATTACK)
        {
                //Chase to player position
                ThisAgent.Resume();
                ThisAgent.SetDestination(PlayerTransform.position);

                //Wait until path is computed
                while(ThisAgent.pathPending)
                        yield return null;

                //Has player run away?
                if(ThisAgent.remainingDistance>ThisAgent.stoppingDistance)
                {
                        //Change back to chase
                        CurrentState = ENEMY_STATE.CHASE;
                        yield break;
                }
                else
                {
                        //Attack
                        PlayerHealth.HealthPoints -= MaxDamage * Time.deltaTime;
                }
```

```
                //Wait until next frame
                yield return null;
            }

            yield break;
        }
        //----------------------------------------
}
//----------------------------------------
```