GAME PROGRAMMING

CSE3122

Name: Vinay Santosh Menon

RegNo: 20BAI1103

Course Code: CSE3122

LAB FAT

Question:

SET -1

Create a **2D game** using UNITY game engine by incorporating the game rules and building the game mechanics as mentioned.

Game Play: Player needs to survive by avoiding colliding with the auto-movable obstacles and should able to reach the goal point.

Rule:

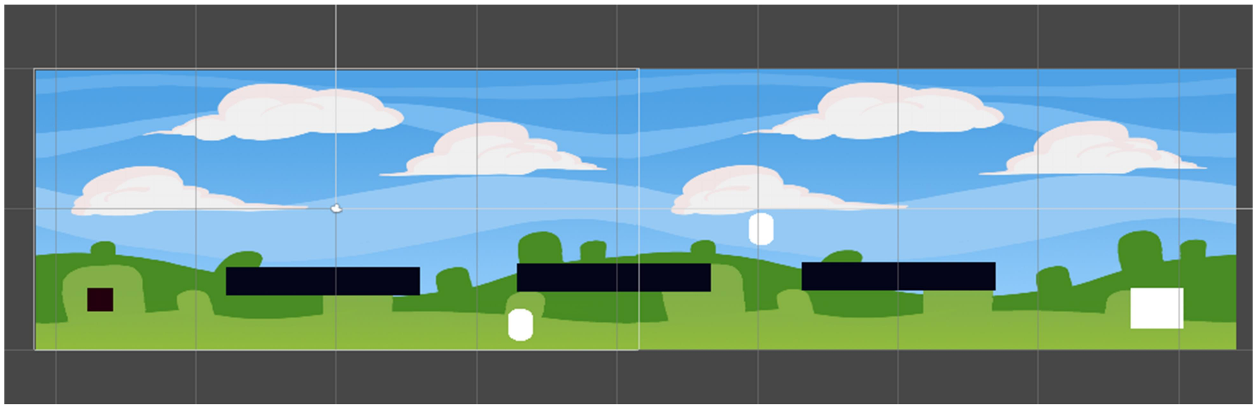
- Player should be within a fixed game environment; otherwise he should lose his life.
- Only 3 lives should be permitted, after that display "Game Over"

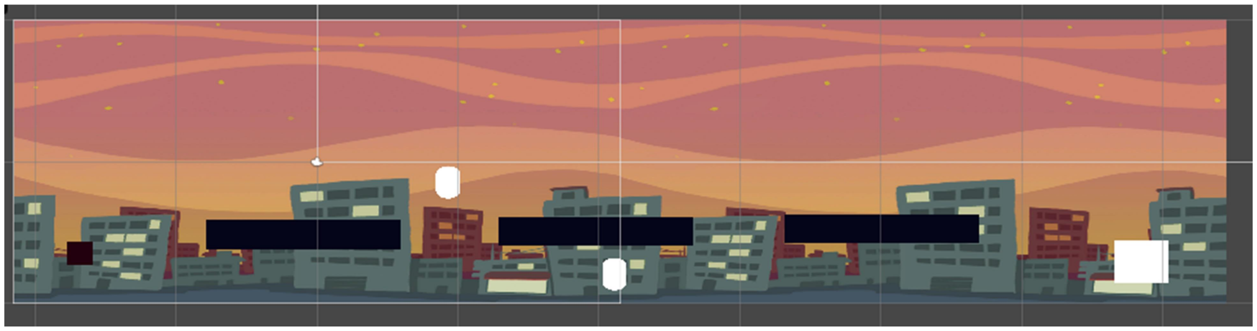
Mechanics:

- Player should be able to move left, right
- Minimum one obstacle is needed.
- Two levels should be provided
- Variation in obstacles should be in 2nd level
- Once player completes the game, Player won caption should be provided
- Proper light effects which suits the game environment
- Proper audio/music/sound effects which suits the game environment

Final ScreenShots:

Scene 1





Scripts:

Movement:

Patrol

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

public class enemy_patrol : MonoBehaviour
{
    public float speed = 5f;
    public float directionChangeInterval = 3f;
    public bool orientToDirection = false;
    public Enums.Directions lookAxis = Enums.Directions.Up;
    [Header("Stops")]
    public Vector2[] waypoints;

    private Vector2[] newWaypoints;
    private int currentTargetIndex;

    void Start ()
    {
        currentTargetIndex = 0;

        newWaypoints = new Vector2[waypoints.Length+1];
        int w = 0;
        for(int i=0; i<waypoints.Length; i++)
        {
            newWaypoints[i] = waypoints[i];
            w = i;
        }

        //Add the starting position at the end, only if there is at least another
point in the queue - otherwise it's on index 0
        int v = (newWaypoints.Length > 1) ? w+1 : 0;
        newWaypoints[v] = transform.position;
        //waypoints = newWaypoints;

        if(orientToDirection)
        {
            Utils.SetAxisTowards(lookAxis, transform, ((Vector3)newWaypoints[1] -
transform.position).normalized);
        }
    }

    public void FixedUpdate ()
    {
        Vector2 currentTarget = newWaypoints[currentTargetIndex];

        GetComponent<Rigidbody2D>().MovePosition(transform.position +
((Vector3)currentTarget - transform.position).normalized * speed *
Time.fixedDeltaTime);

        if(Vector2.Distance(transform.position, currentTarget) <= .1f)
        {
            //new waypoint has been reached
            currentTargetIndex = (currentTargetIndex<newWaypoints.Length-1) ?
currentTargetIndex +1 : 0;
```

```
            if(orientToDirection)
            {
                currentTarget = newWaypoints[currentTargetIndex];
                Utils.SetAxisTowards(lookAxis, transform, ((Vector3)currentTarget
- transform.position).normalized);
            }
        }
    }

    public void Reset()
    {
        waypoints = new Vector2[1];
        Vector2 thisPosition = transform.position;
        waypoints [0] = new Vector2 (2f, .5f) + thisPosition;
    }
}
```

Camera

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

public class follow_camera : MonoBehaviour
{
    public Transform target;

    //Bound camera to limits
    private bool limitBounds = false;
    private float left = -5f;
    private float right = 5f;
    private float bottom = -5f;
    private float top = 5f;

    private Vector3 lerpedPosition;

    private Camera _camera;

    private void Awake() {
        _camera = GetComponent<Camera>();
    }

    // FixedUpdate is called every frame, when the physics are calculated
    void FixedUpdate()
    {
        if(target != null)
        {
            // Find the right position between the camera and the object
            lerpedPosition = Vector3.Lerp(transform.position, target.position,
Time.deltaTime * 10f);
            lerpedPosition.z = -10f;
        }
    }



    // LateUpdate is called after all other objects have moved
    void LateUpdate ()
    {
        if(target != null)
        {
            // Move the camera in the position found previously
            transform.position = lerpedPosition;

            // Bounds the camera to the limits (if enabled)
            if(limitBounds) {
                Vector3 bottomLeft = _camera.ScreenToWorldPoint(Vector3.zero);
                Vector3 topRight = _camera.ScreenToWorldPoint(new
Vector3(_camera.pixelWidth, _camera.pixelHeight));
                Vector2 screenSize = new Vector2(topRight.x - bottomLeft.x,
topRight.y - bottomLeft.y);

                Vector3 boundPosition = transform.position;
                if (boundPosition.x > right - (screenSize.x / 2f)) {
                    boundPosition.x = right - (screenSize.x / 2f);
                }
                if (boundPosition.x < left + (screenSize.x / 2f)) {
                    boundPosition.x = left + (screenSize.x / 2f);
                }

                if (boundPosition.y > top - (screenSize.y / 2f)) {
                    boundPosition.y = top - (screenSize.y / 2f);
                }
                if (boundPosition.y < bottom + (screenSize.y / 2f)) {
                    boundPosition.y = bottom + (screenSize.y / 2f);
                }
                transform.position = boundPosition;
            }
        }
    }
}
```

Last scene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class last_scene : MonoBehaviour
{
    public Text scoreText;
    public void OnTriggerEnter2D(Collider2D col)
    {
        if(col.tag=="Player")
        {
           scoreText.text = "Game Won!!";
        }
    }
}
```

Jump

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

public class player_jump : MonoBehaviour
{
   [Header("Jump setup")]
    // the key used to activate the push
    public KeyCode key = KeyCode.Space;

    // strength of the push
    public float jumpStrength = 10f;

    [Header("Ground setup")]
    //if the object collides with another object tagged as this, it can jump
again
```

```
    public string groundTag = "Ground";

    //this determines if the script has to check for when the player touches the 
ground to enable him to jump again
    //if not, the player can jump even while in the air
    public bool checkGround = true;

    private bool canJump = true;

    // Read the input from the player
    void Update()
    {
        if(canJump
            && Input.GetKeyDown(key))
        {
            // Apply an instantaneous upwards force
            GetComponent<Rigidbody2D>().AddForce(Vector2.up * jumpStrength, 
ForceMode2D.Impulse);
            canJump = !checkGround;
        }
    }

    private void OnCollisionEnter2D(Collision2D collisionData)
    {
        if(checkGround
            && collisionData.gameObject.CompareTag(groundTag))
        {
            canJump = true;
        }
    }
}
```

Movement

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

public class player_movement : MonoBehaviour
{
    [Header("Input keys")]
    private Enums.KeyGroups typeOfControl = Enums.KeyGroups.ArrowKeys;
    public float speed = 5f;
    public Enums.MovementType movementType = Enums.MovementType.AllDirections;

    public bool orientToDirection = false;
    // The direction that will face the player
    public Enums.Directions lookAxis = Enums.Directions.Up;

    private Vector2 movement, cachedDirection;
    private float moveHorizontal;
    private float moveVertical;


    // Update gets called every frame
    void Update ()
    {
        // Moving with the arrow keys
        if(typeOfControl == Enums.KeyGroups.ArrowKeys)
        {
            moveHorizontal = Input.GetAxis("Horizontal");
            moveVertical = Input.GetAxis("Vertical");
        }
        else
        {
            moveHorizontal = Input.GetAxis("Horizontal2");
            moveVertical = Input.GetAxis("Vertical2");
        }

        //zero-out the axes that are not needed, if the movement is constrained
        switch(movementType)
        {
            case Enums.MovementType.OnlyHorizontal:
                moveVertical = 0f;
                break;
            case Enums.MovementType.OnlyVertical:
                moveHorizontal = 0f;
                break;
        }

        movement = new Vector2(moveHorizontal, moveVertical);


        //rotate the GameObject towards the direction of movement
        //the axis to look can be decided with the "axis" variable
        if(orientToDirection)
        {
            if(movement.sqrMagnitude >= 0.01f)
            {
                cachedDirection = movement;
            }
            Utils.SetAxisTowards(lookAxis, transform, cachedDirection);
        }
    }



    // FixedUpdate is called every frame when the physics are calculated
    void FixedUpdate ()
    {
        // Apply the force to the Rigidbody2d
        GetComponent<Rigidbody2D>().AddForce(movement * speed * 10f);
    }
}
```

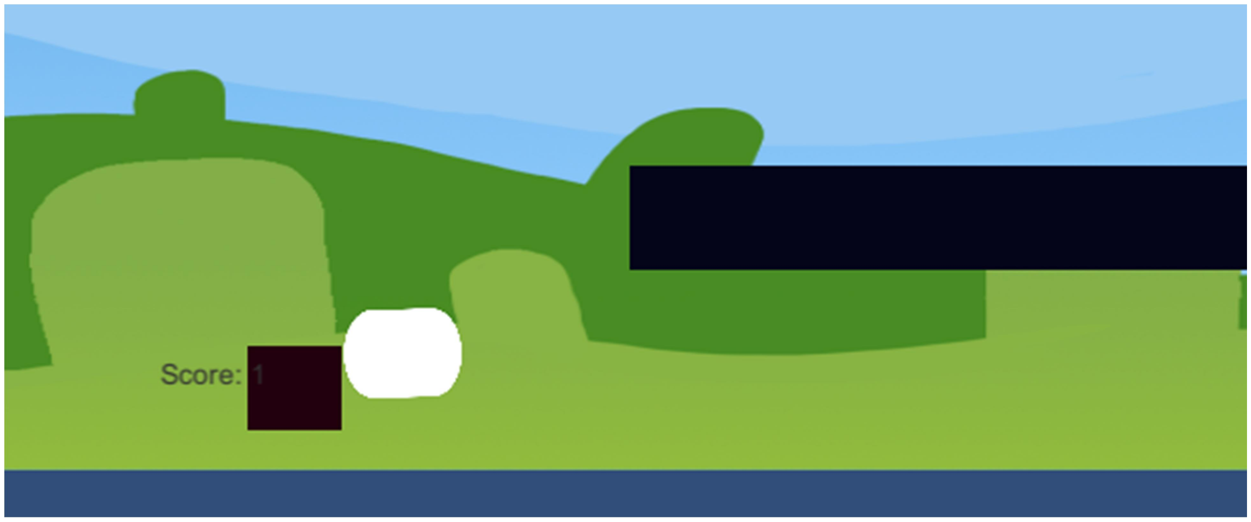
Status

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class player_status : MonoBehaviour
{
    private int currentScore;
    public Text scoreText;

    void Start ()
    {
        currentScore = 3;

    }

    private void HandleScore ()
    {
        scoreText.text = "Score: " + currentScore;
    }

    void OnCollisionEnter2D(Collision2D col)
    {
        if(currentScore<=0)
        {
            scoreText.text = "Game Over";
        }
        else if (col.gameObject.tag == "Player")
        {
            currentScore --;
            HandleScore ();
        }
    }

}
```

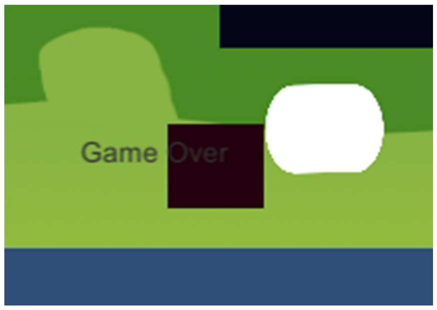
Scene change

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class scene_change : MonoBehaviour
{
    public Text scoreText;
    public void OnTriggerEnter2D(Collider2D col)
    {
        scoreText.text = "Next Level";
        if(col.tag=="Player")
        {
            Debug.Log("hey");
            SceneManager.LoadScene("Scene1");
        }
    }
}
```

Colliding



Health system and game over



Similarly in scene 2