**Coroutines**

Repeating any event after a certain time interval

A type of a function

 IEnumerator Example()

    {

        yield return new WaitForSeconds(2f);

        Debug.Log("This just got executed");

        yield return new WaitForSeconds(2f);

        Debug.Log("This just got executed 2");

        yield return new WaitForSeconds(2f);

        Debug.Log("This just got executed 3");

    }

And inside void start()

{

StartCoroutine(Example());

}

**Inheritance**

Inherits all the public values from one class to another

public class Blizzard : Player

{

}//class

Here, Blizzard is the name of this class and Player is another class,

Blizzard class inherits public values from the Player class.

In the Player class the following must be specified

public  Player()

{

}

**Components**

Add Components to the GameObject in Unity and in C# eg:

Rigidbody rigid;

void start(){

rigid = GetComponent<Rigidbody>();

-------code for the rigidbody mechanism -------

}

All these are above the start function

public float moveForce = 10f;

    public float jumpForce = 11f; These values are for the movement & physics

    public float maxVelocity = 22f;

    private float movementX; For movement

    private Rigidbody2D myBody;

    private Animator animator; These 3 are the components attached to the player

    private SpriteRenderer sprite;

    private string Walk\_Animation = "Walk"; For animation, in the animator, a Boolean ‘Walk’ is created and in the code it is stored here, so when Walk\_Animation receives a true value, the walk animation begins else it goes into idle animation

**Player Movement**

   void Update()

    {

        PlayerMoveKeyboard();

    }

    //Moves the sprite with the keyboard

    void PlayerMoveKeyboard(){

        movementX = Input.GetAxisRaw("Horizontal");

transform.position += new Vector3(movementX, 0f, 0f) \* Time.deltaTime \* moveForce; -> This smoothens the movement of the player and transforms its position

    }

\*\*Vector3 takes in 3 params i.e for 3D

A class named PlayerMoveKeyboard is made and inside it the Input for movement is recorded. The .GetAxisRaw gets the value of the axis of the alignment mentioned in the bracket i.e Horizontal or Vertical.

**Player animation**

void AnimatePlayer(){

        //movement towards the right

        if(movementX > 0){

            animator.SetBool(Walk\_Animation, true);

            sprite.flipX = false;

        }

        //movement towards the left

        else if(movementX < 0){

            animator.SetBool(Walk\_Animation, true);

            sprite.flipX = true;

        }

        //no movement

         else {

            animator.SetBool(Walk\_Animation, false);

        }

    }

**Player Jump**

  private void FixedUpdate() {

        PlayerJump();

    }

\*\* FixedUpdate() -> predefined Function, this function updates itself in every 0.02s where as Update() updates itself once per frame

void PlayerJump(){

        if(Input.GetButtonDown("Jump")){

             if(Input.GetButtonDown("Jump")){

            myBody.AddForce(new Vector2(0f, jumpForce), ForceMode2D.Impulse);

        }

        }

    }

//for checking the ground

 private void OnCollisionEnter2D(Collision2D other) {

        if(other.gameObject.CompareTag(Ground\_Tag)){

                isgrounded = true;

                Debug.Log("Landed");

        }

    }

\*\*OncollisionEnter2D is a predefined function which when sprite collides with another object executes its code

\*\*Ground\_Tag is a string value storing GroundTag as a value, this tag is added as a custom tag to the GroundHolder in Unity

\*\*Vector2 takes two params i.e for 2D

\*\*ForceMode2D -> applies force to the body using mass

ForceMode2D.Force -> adds force slowly

ForceMode2D.Impulse -> Adds force instantly

.GetButtonDown() -> this executes a task when a button that is specified inside the () is pressed

.GetButtonUp() -> also executes a task similar to Down but when the button is released

.GetButton() -> performs task when the button is both pressed and released

**Camera Follows the Player**

\*\*A new script is made and is attached to the Main camera

public class CameraFollow : MonoBehaviour

{

    private Transform player;

    private Vector3 tempPos;

    // Start is called before the first frame update

    void Start()

    {

        player = GameObject.FindWithTag("Player").transform;

    }

    // Update is called once per frame

    void LateUpdate()

    {

         tempPos = transform.position;

  tempPos.x = player.position.x;

        transform.position = tempPos;

    }

}//class

\*\*The .FindWithTag() will search the GameObject in unity for a tag specified inside the () and will transform it.

\*\*The tempos will transform the position of the camera according to the position of the player in x axis.

\*\*LateUpdate() -> will update itself in every frame when enabled and executes only when the update() is executed.

**Auto Sprite Movements**

 [HideInInspector]

    public float speed;

    private Rigidbody2D myBody;

    private void Awake() {

        myBody = GetComponent<Rigidbody2D>();

        speed = 7f;

    }

    // Start is called before the first frame update

    void Start()

    {

    }

    // Update is called once per frame

    void FixedUpdate()

    {

        //Automatically moves the vector

        myBody.velocity = new Vector2(speed, myBody.velocity.y);

    }

**Auto Spawn Sprites**

[SerializeField]

    private GameObject[] monsterReference;

    private GameObject spawnnedMonster;

    [SerializeField]

    private Transform leftPos, rightPos; //2 GameObjects Right and Left are assigned in the Unity to these transform values

    private int randomIndex;

    private int randomSide;

    // Start is called before the first frame update

    void Start()

    {

        StartCoroutine(SpawnMonsters());

    }

    IEnumerator SpawnMonsters(){

     while (true){

            yield return new WaitForSeconds(Random.Range(1,5)); //spawns a new monster between every 1 & 5 sec

      //sets the index value to the length of monsterReference Array and

     //randomizes them for spawnning

        randomIndex = Random.Range(0, monsterReference.Length);

        randomSide = Random.Range(0,2);

        spawnnedMonster = Instantiate(monsterReference[randomIndex]); //sets the monsterReference index value to the randomIndex value

        //left side

        if(randomSide == 0){

            spawnnedMonster.transform.position = leftPos.position;

            spawnnedMonster.GetComponent<Monster>().speed = Random.Range(4,10); //sets the speed value between 4 & 10

        }else{

            //right side

            spawnnedMonster.transform.position = rightPos.position;

            spawnnedMonster.GetComponent<Monster>().speed = -Random.Range(4,10);

            //flips the sprite

            spawnnedMonster.transform.localScale = new Vector3(-1f, 1f, 1f);

        }

     }//while loop, this will spawn multiple monsters

    }

**Destroy the player when it comes in contact of other characters**

   In Player.cs file

private void OnCollisionEnter2D(Collision2D other) {

//checks the ground

        if(other.gameObject.CompareTag(Ground\_Tag)){

                isgrounded = true;

                Debug.Log("Landed");

        }

        //destroy player

        //destorys the player when it collides with anything attached to the enemy tag

        if (other.gameObject.CompareTag(Enemy\_Tag)){

            Destroy(gameObject);

        }

    }

//in CameraFollow.cs file

void LateUpdate()

    {

        //checks if player is destroyed, if yes then it exits else continues with the code block

        if(!player){

            return;

        }

//attaches the camera to the player and follows the player till min & max X

        tempPos = transform.position;

        tempPos.x = player.position.x;

        if(tempPos.x < minX){

            tempPos.x = minX;

        }

        if(tempPos.x > maxX){

            tempPos.x = maxX;

        }

        transform.position = tempPos;

    }

Edit -> Project Setting -> Physics 2D -> add layer, attach the sprite to that layer so that they don’t collide with each other

**Destroy sprites which are off the scene**

public class Collector : MonoBehaviour

{

//for enemies

   private void OnTriggerEnter2D(Collider2D other) {

       if (other.CompareTag("Enemy")){

           Destroy(other.gameObject);

       }

//for player

       if(other.gameObject.CompareTag("Player")){

           Destroy(other.gameObject);

           Debug.Log("Player is deleted");

       }

   }

}

In unity, a GameObject named Collector Holder was made containing Left and Right GameObjects and in which this cs file was added

**Scenes**

File -> new scene -> save

**Main Menu**

Hierarchy -> RMB -> UI -> desired component

Canvas ->

1. Render mode -> Screen Space – Camera
2. Render Camera -> Main camera
3. Canvas Scaler -> x: 1920 y: 1080, screen match mode -> 0.5

UI Component

Rect Transform -> Rectangle -> Position your ui component in any corner or center

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement; -> for changing scene

public class MainMenuController : MonoBehaviour

{

//here the scene changes from the main menu to gameplay scene

    public void PlayGame(){

        SceneManager.LoadScene("Gameplay");

    }

}//class

.LoadScene() -> () accepts the scene name

Loading Scene in Unity:

Desired scene -> File -> Build Setting -> Add open scene

**Selecting a Player**

public class GameManager : MonoBehaviour

{

//creates a game manager -> instance

    public static GameManager instance;

//for player id (sort of)

    [SerializeField]

    private GameObject[] players;

//gets the sort of player id

    private int \_charIndex;

    public int CharIndex{

        get {

            return \_charIndex;

        }

        set {

            \_charIndex = value;

        }

    }

    //instance of MonoBehavious class

        private void Awake() {

        if(instance == null){

            instance = this;

        }

    }

Creates an instance for all the public functions and variable in this GameManager class so that other cs class files can use it to store and edit data (simply creates a static function)

##### MainMenuController #####

 public void PlayGame(){

        //shows the name of the button clicked & converts to integer

        int selectedPlayer =

            int .Parse(UnityEngine.EventSystems.EventSystem.current.currentSelectedGameObject.name);

        GameManager.instance.CharIndex = selectedPlayer;

        //   SceneManager.LoadScene("Gameplay");

    }