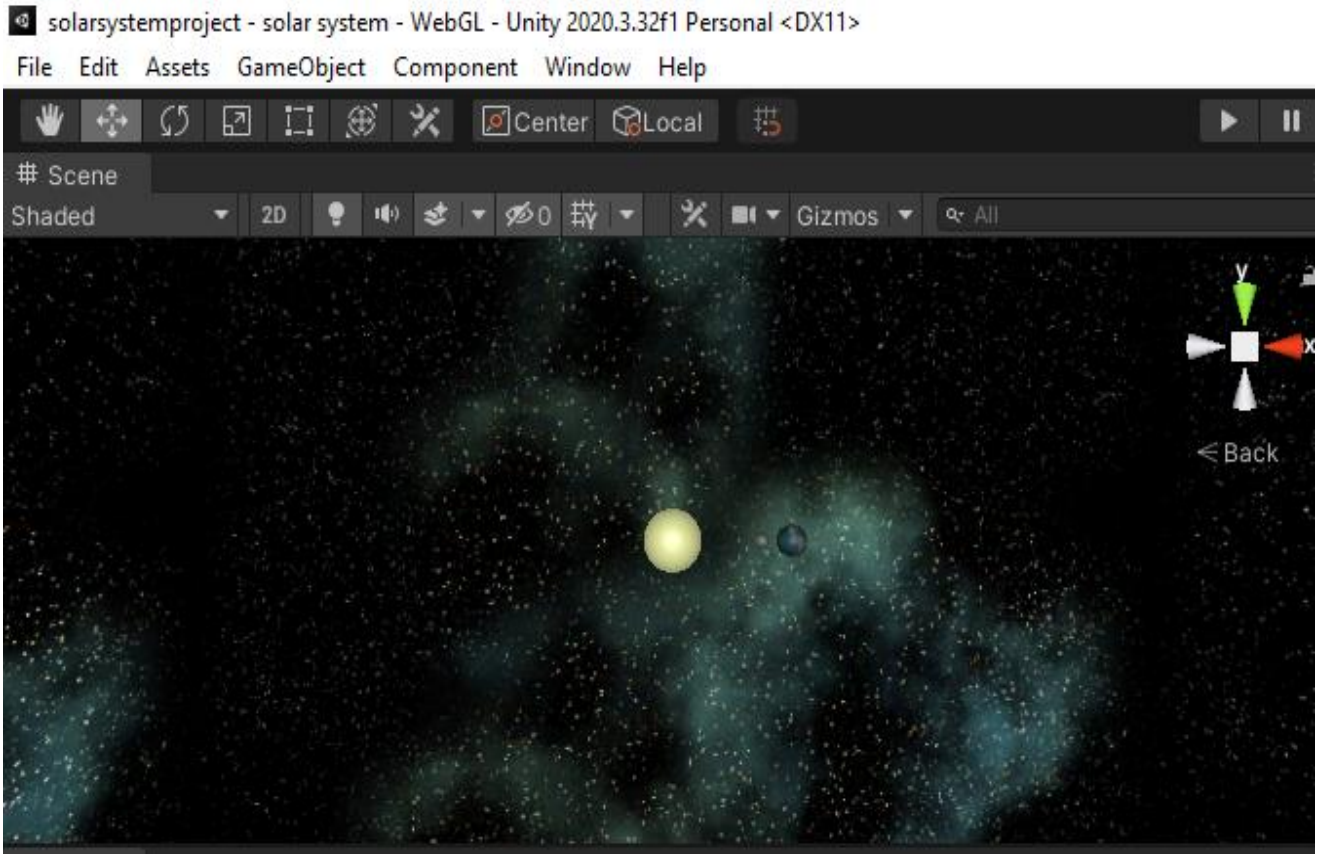
SOLAR SYSTEM SIMULATION PROJECT



Created by:

Anurag Aryal

President and Founder,

CS Association of Nepal

Words from the designer:

This is my first project using Unity Hub, which is one of the most popular game editors. I happen to work on this project while learning game design and development from Professor Brian Winn of Michigan State University. For someone like me who want to master the concept of game development, it is incumbent to understand the fundamentals of game editors like prefabs, game assets, transforms, lighting, and projections. This solar system simulation project provided me with the opportunity to fully comprehend those concepts and familiarized myself with various methods for configuring game objects and adding components, both of which were critical in the 2D shooter game I built after this project. For designing and coding purposes, I used Unity 2020.3.32f1 and Virtual Studio, respectively.

HOW I DEVELOPED THE SIMULATION DESIGN

To commence, three 3D spheres were added to the hierarchy panel of the Unity editor, and their positional, rotational, and scale co-ordinates were configured using the global co-ordinate system and their real-world relative sizes. The sun had a blazing sound, the earth had a drone hum sound, and the moon had no sound, but texture was added to them all. The use of the rotate around script and the parent-child relationship assured that the earth revolved around the sun and the moon revolved around the earth. For the perspective view (the primary view), I utilized the main camera with the audio listener switched on, and the minimap camera for the orthographic view (small rectangular view on bottom left). All audio was kept with the default logarithmic volume roll off, loop, and 3D spatial blend (excluding sun). To keep the sun in focus, I used the Look at target script on the main camera. After that, a new 3D object for the comet was built, with transforms attributed on a real-life scenario. The application of projectile script and the creation of prefabs resulted in comet spawners with a 5 second spawning time. Spawner and Rotate Around scripts were also incorporated for the spawning comets.

CODES I USED FOR THIS PROJECT:

Look at Target script

```
using UnityEngine;
using System.Collections;

public class LookAtTarget : MonoBehaviour {

    [Tooltip("This is the object that the script's game object will look at by default")]
    public GameObject defaultTarget; // the default target that the camera should look at

    [Tooltip("This is the object that the script's game object is currently look at based
on the player clicking on a gameObject")]
    public GameObject currentTarget; // the target that the camera should look at

    // Start happens once at the beginning of playing. This is a great place to setup the
behavior for this gameObject
       void Start () {
              if (defaultTarget == null)
              {
            defaultTarget = this.gameObject;
                    Debug.Log ("defaultTarget target not specified. Defaulting to parent
GameObject");
              }

        if (currentTarget == null)
        {
            currentTarget = this.gameObject;
            Debug.Log("currentTarget target not specified. Defaulting to parent
GameObject");
        }
    }

       // Update is called once per frame
    // For clarity, Update happens constantly as your game is running
    void Update()
    {
        // if primary mouse button is pressed
        if (Input.GetMouseButtonDown(0))
        {
            // determine the ray from the camera to the mousePosition
            Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

            // cast a ray to see if it hits any gameObjects
            RaycastHit[] hits;
            hits = Physics.RaycastAll(ray);

            // if there are hits
            if (hits.Length>0)
            {
                // get the first object hit
                RaycastHit hit = hits[0];
                currentTarget = hit.collider.gameObject;

                Debug.Log("defaultTarget changed to "+currentTarget.name);
            }
        } else if (Input.GetMouseButtonDown(1)) // if the second mouse button is pressed
        {
            currentTarget = defaultTarget;
```

```
            Debug.Log("defaultTarget changed to " + currentTarget.name);
        }

       // if a currentTarget is set, then look at it
        if (currentTarget!=null)
        {
            // transform here refers to the attached gameobject this script is on.
            // the LookAt function makes a transform point it's Z axis towards another
point in space
            // In this case it is pointing towards the target.transform
            transform.LookAt(currentTarget.transform);
        } else // reset the look at back to the default
        {
            currentTarget = defaultTarget;
            Debug.Log("defaultTarget changed to " + currentTarget.name);
        }
    }
}
```

Projectile script

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

/// <summary>
/// A class to make projectiles move
/// </summary>
public class Projectile : MonoBehaviour
{

    [Tooltip("The distance this projectile will move each second in meters.")]
    public float projectileSpeed = 3.0f;

    [Tooltip("How far away from the main camera before destroying the projectile
gameobject in meters.")]
    public float destroyDistance = 20.0f;

    /// <summary>
    /// Description:
    /// Standard Unity function called once per frame
    /// Inputs:
    /// none
    /// Returns:
    /// void (no return)
    /// </summary>
    private void Update()
    {
        MoveProjectile();
    }

    /// <summary>
    /// Description:
    /// Move the projectile in the direction it is heading
    /// Inputs:
    /// none
```

```
    /// Returns:
    /// void (no return)
    /// </summary>
    private void MoveProjectile()
    {
        // move the transform
        transform.position = transform.position + transform.forward * projectileSpeed *
Time.deltaTime;

        // calculate the distance from the main camera
        float dist = Vector3.Distance(Camera.main.transform.position,
transform.position);

        // if the distance is greater than the destroyDistance
        if (dist>destroyDistance)
        {
            Destroy(this.gameObject); // destroy the gameObject
        }
    }
   }
```

Rotate around Target Script

```
using UnityEngine;
using System.Collections;

public class RotateAround : MonoBehaviour {

    [Tooltip("This is the object that the script's game object will rotate around")]
       public Transform target; // the object to rotate around
    [Tooltip("This is the speed at which the object rotates")]
       public int speed; // the speed of rotation

       void Start() {
              if (target == null)
              {
                     target = this.gameObject.transform;
                     Debug.Log ("RotateAround target not specified. Defaulting to this
GameObject");
              }
       }

       // Update is called once per frame
       void Update () {
              // RotateAround takes three arguments, first is the Vector to rotate around
              // second is a vector that axis to rotate around
              // third is the degrees to rotate, in this case the speed per second
              transform.RotateAround(target.transform.position,target.transform.up,speed
* Time.deltaTime);
       }
}
```

Spawner Script

```
using System.Collections;
```

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

public class Spawner : MonoBehaviour
{

    [Tooltip("The Prefab to be spawned into the scene.")]
    public GameObject spawnPrefab = null;

    [Tooltip("The time between spawns")]
    public float spawnTime = 5.0f;

    // keep track of time passed for next spawn
    private float nextSpawn = 0f;

    // Start is called before the first frame update
    void Start()
    {
        nextSpawn = 0f;
    }

    // Update is called once per frame
    void Update()
    {
        // update the time until nextSpawn
        nextSpawn += Time.deltaTime;

        // if time to spawn
        if (nextSpawn>spawnTime)
        {
            // Spawn the gameObject at the spawners current position and rotation
            GameObject projectileGameObject = Instantiate(spawnPrefab,
transform.position, transform.rotation, null);

            // reset the time until nextSpawn
            nextSpawn = 0f;
        }

    }
```

USER GUIDE:

I have made my simulation project available on the internet @ https://anurag-aryal.itch.io/solar-system (click on run game). I recommend you to click on earth or comet to change the point of view of main camera for hyper-realistic experience.