Asteroid.cs

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


/// Represents an asteroid in the game. Handles spawning, movement, collision
detection, and destruction.

public class Asteroid : MonoBehaviour
{
    public GameObject asteroidObject, spaceshipPrefab, AsteroidFragment,
bulletPrefab;
    private Vector3 spawnPoint;
    private bool ignoreCollisions = true;

    // Start - called before the first frame update
    void Start()
    {
        //Set tag to Asteroid
        asteroidObject.tag = "Asteroid";

        //check the asteroids scale - if it's a small asteroid don't have it
spawn at the edge of the screen
        if(asteroidObject.transform.localScale.x > 0.1f)
        {
            Invoke("DisableCollisionIgnore", 0.1f);
            return;
        }

        //Set the asteroid's position at a random position near the edges of
the screen
        if (Random.Range(0, 2) == 0)
        {
            //Spawn on top or bottom
            if (Random.Range(0, 2) == 0)
            {
                //Spawn on top
                spawnPoint = new Vector3(Random.Range(-30f, 30f), 0, 30);
            }
            else
            {
                //Spawn on bottom
                spawnPoint = new Vector3(Random.Range(-30f, 30f), 0, -30);
            }
        }
        else
        {
            //Spawn on left or right
            if (Random.Range(0, 2) == 0)
            {
                //Spawn on left
                spawnPoint = new Vector3(-30, 0, Random.Range(-30f, 30f));
            }
            else
            {
                //Spawn on right
                spawnPoint = new Vector3(30, 0, Random.Range(-30f, 30f));
            }
        }

        //Set the asteroid's position
        asteroidObject.transform.position = spawnPoint;

        //Move the asteroid in a random direction
        asteroidObject.GetComponent<Rigidbody>().AddForce(new
Vector3(Random.Range(-700f, 700f), 0, Random.Range(-700f, 700f)));

        //Rotate the asteroid in a random direction
        asteroidObject.GetComponent<Rigidbody>().AddTorque(new
Vector3(Random.Range(-500f, 500f), Random.Range(-500f, 500f), Random.Range(-
500f, 500f)));



        //This is a method that disables collisions for a tenth of a second at
spawn in, in order to prevent not valid collisions
        Invoke("DisableCollisionIgnore", 0.1f);
    }

    void DisableCollisionIgnore()
    {
        //Disabling collision ignore boolean
        ignoreCollisions = false;
    }



    /*Each time an asteroid collides with something, spawn a few of the tiny
asteroid prefabs at the point of
impact. They should be destroyed shortly afterwards. */

    void SpawnCollisionDebris(Vector3 collisionPoint, float multiplier)
    {
        //Spawn 3 small asteroids at the point of collision
        for (int i = 0; i < 3 * multiplier; i++)
        {
            GameObject smallAsteroid =
GameObject.Instantiate(AsteroidFragment);
            //Setting position to the collision point with some variance and
scaling it down
            smallAsteroid.transform.position = new Vector3(
                collisionPoint.x + Random.Range(-0.5f, 0.5f),
                collisionPoint.y + Random.Range(-0.5f, 0.5f),
                collisionPoint.z + Random.Range(-0.5f, 0.5f)
            );
            smallAsteroid.transform.localScale = new Vector3(0.01f, 0.01f,
0.01f);
            //Adding a random force and torque to the small asteroids
            smallAsteroid.GetComponent<Rigidbody>().AddForce(new
Vector3(Random.Range(-30f, 30f), 0, Random.Range(-30f, 30f)));
            smallAsteroid.GetComponent<Rigidbody>().AddTorque(new
Vector3(Random.Range(-30f, 30f), Random.Range(-30f, 30f), Random.Range(-30f,
30f)));
        }
    }
    void SpawnSmallerAsteroids(Vector3 collisionPoint)
    {
        //Spawn between 3-4 small asteroids at the point of collision
        Debug.Log("SpawnSmallerAsteroids called");
        for (int i = 0; i < Random.Range(3, 5); i++)
        {
            GameObject asteroid = Instantiate(Resources.Load("Asteroid",
typeof(GameObject))) as GameObject;


            //Setting position to the collision point and scaling it down
            asteroid.transform.position = collisionPoint;

            asteroid.transform.localScale = new Vector3(Random.Range(0.01f,
0.06f), Random.Range(0.01f, 0.06f), Random.Range(0.01f, 0.06f));
            //Adding a random force and torque to the small asteroids
            asteroid.GetComponent<Rigidbody>().AddForce(new
Vector3(Random.Range(-100f, 100f), 0, Random.Range(-100f, 100f)));
            asteroid.GetComponent<Rigidbody>().AddTorque(new
Vector3(Random.Range(-100f, 100f), Random.Range(-100f, 100f), Random.Range(-
100f, 100f)));

        }
    }

    /*Method for calling SpawnCollisionDebris on collisions */
    void OnCollisionEnter(Collision collision)
    {
        if (ignoreCollisions)
        {
            return;
        }

        Debug.Log("Collision object is: " + collision.gameObject.tag);
        switch (collision.gameObject.tag)
        {
            case "Bullet":
                //Calling SpawnCollisionDebris with the point of collision
                SpawnCollisionDebris(collision.contacts[0].point, 3F);
                //Destroying the bullet
                Destroy(collision.gameObject);
                //Destroying the asteroid
                Destroy(asteroidObject);

                if (asteroidObject.transform.localScale.x > 0.1f)
                {
                    //Add points to score
                    GameManager.instance.AddScore(12);

                    SpawnCollisionDebris(collision.contacts[0].point, 1F);
//extra debris for larger asteroids (also fun)
                    //Destroying the asteroid
                    SpawnSmallerAsteroids(collision.contacts[0].point);
                }
                else if (asteroidObject.transform.localScale.x > 0.05F)
                {
                    //Add points to score
                    GameManager.instance.AddScore(7);

                    SpawnCollisionDebris(collision.contacts[0].point, 2F);
//extra debris for larger asteroids (also fun)
                }
                break;
            case "SpaceShip":
                //Destroy & respawn spaceship handled in spaceship script -
return
                break;
            case "Asteroid":
                SpawnCollisionDebris(collision.contacts[0].point, 1.5F);
                break;
            default:
                break;
        }
    }


}
```

UIManager.cs

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

public class UIManager : MonoBehaviour
{
    public static GameObject instance;

    //UI Panel references
    public GameObject menuPanel;
    public GameObject playingPanel;
    //UI Text references
    public TextMeshProUGUI scoreText;
    public TextMeshProUGUI highScoreText;
    public TextMeshProUGUI livesText;
    public Button startButton;



    void Start()
    {
        instance = this.gameObject;

        // //Get references to UI panels and text
        // menuPanel = instance.Find("MenuPanel");
        // playingPanel = instance.Find("PlayingPanel");
        // scoreText =
instance.Find("ScoreText").GetComponent<TextMeshProUGUI>();
        // highScoreText =
instance.Find("HighScoreText").GetComponent<TextMeshProUGUI>();
        // livesText =
instance.Find("LivesText").GetComponent<TextMeshProUGUI>();
    }

    void onClick()
    {
        Debug.Log("Start button clicked");
        GameManager.instance.StartNewGame();
    }

    public void ShowMenu()
    {
        //Show menu panel, hide playing panel
        menuPanel.SetActive(true);
        playingPanel.SetActive(false);
```

```csharp
    }

    public void ShowPlaying()
    {
        //Show playing panel, hide menu panel
        menuPanel.SetActive(false);
        playingPanel.SetActive(true);
    }

    public void UpdateScoreText()
    {
        //Update score text
        scoreText.text = "Score: " + GameManager.gameScore;
    }

    public void UpdateHighScoreText()
    {
        //Update high score text
        highScoreText.text = "High Score: " + GameManager.highScore;
    }

    public void UpdateLivesText()
    {
        //Update lives text
        livesText.text = "Lives: " + GameManager.lives;
    }

    public void UpdateUI()
    {
        //Update all UI elements
        UpdateScoreText();
        UpdateHighScoreText();
        UpdateLivesText();
    }

    public void Play_Button_Clicked()
    {
        // //Start the game
        GameManager.instance.StartNewGame();
        Debug.Log("Play button clicked");

    }
}
```

Bullet.cs

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

/// This class represents the behavior of a bullet in the game. It checks if
the bullet is offscreen every 0.2 seconds and destroys it if so.

public class Bullet : MonoBehaviour
{
    public GameObject bullet;
    public GameObject spaceship;

    void Start()
    {
        //Check if bullet is offscreen every 0.2 seconds - destroy if so
        InvokeRepeating("DestroyIfOffScreen", 0.2f, 0.2f);
    }


    void DestroyIfOffScreen()
    {
        Vector3 pos = transform.position;
        Vector3 vel = GetComponent<Rigidbody>().velocity;
        //if offscreen, destroy bullet
        if ((pos.x > GameManager.screenTopRight.x && vel.x >= 0f)
        || (pos.x > GameManager.screenTopRight.x && vel.x >= 0f)
        || (pos.z < GameManager.screenBottomLeft.z && vel.z <= 0f)
        || (pos.z > GameManager.screenTopRight.z && vel.z >= 0f))
        {
            Destroy(bullet);
        }
    }
}
```

GameManager.cs

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

public class GameManager : MonoBehaviour
{

    public GameObject asteroidPrefab, spaceshipPrefab;

    public GameObject uiManager;
    public static GameManager instance;
    public static Vector3 screenBottomLeft, screenTopRight;
    public static float screenWidth, screenHeight;
    public static int currentGameLevel, gameScore, lives, highScore;

    public enum GameState { Menu, Playing }
    GameState currentState = GameState.Menu;
    public UIManager uiManagerScript;

    // Start is called before the first frame update
    void Start()
    {
        instance = this;
        uiManagerScript = uiManager.GetComponent<UIManager>();
        UpdateUI();
    }

    public void UpdateUI()
    {
        switch (currentState)
        {
            case GameState.Menu:
                uiManagerScript.ShowMenu();
                break;
            case GameState.Playing:
                uiManagerScript.ShowPlaying();
                break;
            default:
                break;
        }
    }


    public void StartNewGame()
    {
        currentState = GameState.Playing;
        currentGameLevel = 0;
        gameScore = 0;
        lives = 3;
        UpdateUI();


        Camera.main.transform.position = new Vector3(0, 30, 0);
        Camera.main.transform.LookAt(new Vector3(0, 0, 0), new Vector3(0, 0,
1));

        CreatePlayerSpaceship();
        StartNextLevel();
    }

    void StartNextLevel()
    {
        currentGameLevel++;
        int numberOfAsteroids = currentGameLevel * 5;

        screenBottomLeft = Camera.main.ViewportToWorldPoint(new Vector3(-0.1f,
-0.1f, 30f));
        screenTopRight = Camera.main.ViewportToWorldPoint(new Vector3(1.1f,
1.1f, 30f));
        screenWidth = screenTopRight.x - screenBottomLeft.x;
        screenHeight = screenTopRight.z - screenBottomLeft.z;
        Debug.Log("BottomLeft: " + screenBottomLeft);
        Debug.Log("TopRight: " + screenTopRight);
        Debug.Log("Width: " + screenWidth);
        Debug.Log("Height: " + screenHeight);

        //instantiate a set of asteroids towards the edges of the visible
screen using a for loop
        for (int i = 0; i < numberOfAsteroids; i++)
        {
            GameObject go = Instantiate(instance.asteroidPrefab) as
GameObject;

            float x, z;
            if (Random.Range(0f, 1f) < 0.5f)
                x = screenBottomLeft.x + Random.Range(0f, 0.15f) *
screenWidth; // near the left edge
            else
                x = screenTopRight.x - Random.Range(0f, 0.15f) * screenWidth;
// near the right edge
            if (Random.Range(0f, 1f) < 0.5f)
                z = screenBottomLeft.z + Random.Range(0f, 0.15f) *
screenHeight; // near the bottom edge
            else
```

```
                z = screenTopRight.z - Random.Range(0f, 0.15f) * screenHeight;
// near the top edge

            go.transform.position = new Vector3(x, 0f, z);

            go.transform.localScale = new Vector3(Random.Range(0.1f, 0.17f),
Random.Range(0.1f, 0.17f), Random.Range(0.1f, 0.17f));
        }
    }

    private static void CreatePlayerSpaceship()
    {
        // instantiate the player's spaceship
        GameObject go = Instantiate(instance.spaceshipPrefab);
        go.transform.position = Vector3.zero;
        go.transform.localScale = new Vector3(0.2f, 0.2f, 0.2f);
    }

    public void AddScore(int score)    {
        gameScore += score;
        if (gameScore > highScore)
            highScore = gameScore;
        uiManagerScript.UpdateHighScoreText();
        uiManagerScript.UpdateScoreText();
    }

    public void SubtractLife()
    {
        lives--;
        uiManagerScript.UpdateLivesText();
        if (lives <= 0)
        {
            GameObject.FindGameObjectsWithTag("Asteroid").ToList().ForEach(ast
eroid => Destroy(asteroid));
            currentState = GameState.Menu;
            UpdateUI();
        }
    }

    public void AsteroidCheck()
    {
        if (GameObject.FindGameObjectsWithTag("Asteroid").Length == 0)
        {
            StartNextLevel();
        }
    }

}
```

Spaceship.cs

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

/// <summary>
/// Spaceship class that controls the spaceship movement, shooting, and
collision detection.
/// </summary>
public class Spaceship : MonoBehaviour
{
    // inspector settings
    public GameObject spaceship, bullet;
    public GameObject gameManager;
    public static int bulletCount = 0;
    // public member data
    [HideInInspector] public bool isInvulnerable = true;

    // Start is called before the first frame update
    void Start()
    {


        //Wrap spaceship to other side of screen, check every 0.2 seconds. 5
times a second
        InvokeRepeating("CheckIfOffScreen", 0.2f, 0.2f);

        //Reset bullet count every second
        InvokeRepeating("ResetBulletCount", 1f, 1f);

        //Make spaceship vulnerable after 2 seconds
        Invoke("MakeVulnerable", 2f);

    }

    private void MakeVulnerable()
    {
        isInvulnerable = false;
        Debug.Log("No longer invulnerable");
    }

    // Update is called once per frame
    void Update()
    {
        /*apply a physics force to accelerate the spaceship forward if the Up
arrow is held, or
        rotate it left/right if the Left/Right arrows are held.*/
        //Checking if the Up arrow is held, if so check if within velocity
limit, if so add force
        if (Input.GetKey(KeyCode.UpArrow) &&
GetComponent<Rigidbody>().velocity.magnitude < 14)
        {
            GetComponent<Rigidbody>().AddForce(transform.up * 7);
        }
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            GetComponent<Rigidbody>().AddTorque(transform.forward * -4);
        }
        if (Input.GetKey(KeyCode.RightArrow))
        {
            GetComponent<Rigidbody>().AddTorque(transform.forward * 4);
        }

        //Fire bullet if spacebar is pressed - spawn at front of spaceship
        //Position should be positioned and rotated appropriately, with
rigidbody given an appropriate velocity
        //Limit of 4 bullets fired per second spaceship.
        if (Input.GetKeyDown(KeyCode.Space) && bulletCount < 4)
        {
            GameObject bullet = Instantiate(Resources.Load("Bullet",
typeof(GameObject))) as GameObject;
            bullet.transform.position = spaceship.transform.position +
spaceship.transform.up * 1.5f;
            bullet.transform.rotation = spaceship.transform.rotation;
            bullet.GetComponent<Rigidbody>().velocity = spaceship.transform.up
* 20;
            bulletCount++;
        }
    }

    void ResetBulletCount()
    {
        bulletCount = 0;
    }
    /// Detects collision with an asteroid and destroys the spaceship. A new
spaceship is spawned in the center of the screen.
    /// </summary>
    /// <param name="col"></param>
    void OnCollisionEnter(Collision col)
    {
        if (col.gameObject.tag == "Asteroid" && !isInvulnerable)
        {
            //Subtract a life from game manager using SubtractLife()
            GameManager.instance.SubtractLife();
            Destroy(gameObject.transform.parent.gameObject);

            Debug.Log("Spaceship destroyed");
            //Spawn a new spaceship in the center of the screen
            GameObject spaceship = Instantiate(Resources.Load("Spaceship",
typeof(GameObject))) as GameObject;
            spaceship.transform.transform.localScale = new Vector3(0.2f, 0.2f,
0.2f);
        }

    }


    // Having the player spaceship respond to moving off-screen, in the same
way that asteroids already do
    void CheckIfOffScreen()
    {
        Vector3 currentWorldPos = spaceship.transform.position;
        Vector3 viewPosition =
Camera.main.WorldToViewportPoint(currentWorldPos);
        if (viewPosition.x > 1f)
        {
            spaceship.transform.position = new Vector3(-currentWorldPos.x + 1,
0, currentWorldPos.z);
        }

        if (viewPosition.y < 0f)
        {
            spaceship.transform.position = new Vector3(currentWorldPos.x, 0, -
currentWorldPos.z - 1);
        }

        if (viewPosition.x < 0f)
        {
            spaceship.transform.position = new Vector3(-currentWorldPos.x - 1,
0, currentWorldPos.z);
        }

        if (viewPosition.y > 1f)
        {
            spaceship.transform.position = new Vector3(currentWorldPos.x, 0, -
currentWorldPos.z + 1);
        }


    }
}
```

TimedLife.cs

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

/// <summary>
/// Destroys the game object after a random amount of time between minLifetime
and maxLifetime.
/// </summary>
public class TimedLife : MonoBehaviour
{
    public float minLifetime, maxLifetime;
    void Start()
    {
        StartCoroutine(HandleLifetime());
    }
    private IEnumerator HandleLifetime()
    {
        yield return new WaitForSeconds(Random.Range(minLifetime,
maxLifetime));
        Destroy(gameObject);
    }
}
```

ScreenWrapper.cs

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

/// <summary>
/// This class wraps the game object around the screen if it goes off-screen.
/// </summary>
public class ScreenWrapper : MonoBehaviour
{
    // Start is called before the first frame update

    // inspector settings
    public Rigidbody rigidBody;
    //
    // Use this for initialization
    void Start()
    {
        // start periodically checking for being off-screen
        InvokeRepeating("CheckScreenEdges", 0.1f, 0.1f);
    }
    private void CheckScreenEdges()
    {
        Vector3 pos = transform.position;
        Vector3 vel = rigidBody.velocity;
        float xTeleport = 0f, zTeleport = 0f;
        if (pos.x < GameManager.screenBottomLeft.x && vel.x <= 0f)
            xTeleport = GameManager.screenWidth;
        else if (pos.x > GameManager.screenTopRight.x && vel.x >= 0f)
            xTeleport = -GameManager.screenWidth;
        if (pos.z < GameManager.screenBottomLeft.z && vel.z <= 0f)
            zTeleport = GameManager.screenHeight;
        else if (pos.z > GameManager.screenTopRight.z && vel.z >= 0f)
            zTeleport = -GameManager.screenHeight;
        if (xTeleport != 0f || zTeleport != 0f)
            transform.position = new Vector3(pos.x + xTeleport, 0f, pos.z +
zTeleport);
    }
}
```