Adding the Player Spaceship – Assignment 5

Game Manager

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

public class GameManager : MonoBehaviour
{
    public int currentGameLevel = 1; // Set the starting game level
    public GameObject asteroidPrefab; // Reference to the asteroid prefab
    public GameObject spaceshipPrefab;

    void Start()
    {
        // Set the camera's position
        Camera.main.transform.position = new Vector3(0f, 40f, 0f);
        Camera.main.transform.LookAt(new Vector3(0f, 0f, 0f), Vector3.up);

        // Call the method to start a new level
        StartNewLevel();

        // Call the create spaceship method
        CreatePlayerSpaceship();
    }

    // Method to start a new game level
    void StartNewLevel()
    {
        // Calculate the number of asteroids based on the current game level
        int numAsteroids = currentGameLevel * 3;

        for (int i = 0; i < numAsteroids; i++)
        {
            // Generate a random spawn position within the screen boundaries
            Vector3 spawnPosition = new Vector3(Random.Range(-15f, 15f), 0f,
Random.Range(-15f, 15f));

            // Ensure the Y position is at ground level (0)
            spawnPosition.y = 0f;

            // Add a buffer to the Z position to prevent immediate wrap-around
            spawnPosition.z += 2f;

            // Instantiate asteroid
            Instantiate(asteroidPrefab, spawnPosition, Quaternion.identity);
        }
    }

    // Method to create the player spaceship at the center of the screen
    void CreatePlayerSpaceship()
    {
        Quaternion rot = spaceshipPrefab.transform.rotation;
        Instantiate(spaceshipPrefab, Vector3.zero, rot);
    }
}
```

Asteroids

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

public class Asteroid : MonoBehaviour
{
    Rigidbody rb;
    public float moveSpeed = 6f;
    public GameObject smallAsteroidPrefab;
    public int numSmallAsteroidsToSpawn = 3; // Number of small asteroids to
spawn on collision.

    void Start()
    {
        rb = GetComponent<Rigidbody>();

        // Calculate a random direction vector on the XZ plane
        Vector3 randomDirection = Random.onUnitSphere;
        randomDirection.y = 0f;

        // Set the initial velocity based on your moveSpeed
        rb.velocity = randomDirection * moveSpeed;

        // Generate random torque (angular velocity)
        Vector3 randomTorque = new Vector3(
            Random.Range(5f, 15f),
            Random.Range(5f, 15f),
            Random.Range(5f, 15f)
        );

        InvokeRepeating("CheckScreenEdges", 0f, 0.2f);
    }

    void CheckScreenEdges()
    {
        Debug.Log("Current position: " + transform.position);
        // Check if the asteroid has left the screen
        if (Mathf.Abs(transform.position.x) > 22f ||
Mathf.Abs(transform.position.z) > 20f)
        {
            // Wrap around to the opposite side
            transform.position = new Vector3(-transform.position.x, 0, -
transform.position.z);
        }
    }

    private void OnCollisionEnter(Collision collision)
    {
        // Check if the collision is with another object (e.g., player
spaceship).
        if (collision.gameObject.CompareTag("Player") ||
collision.gameObject.CompareTag("Asteroid"))
        {
            // Spawn small asteroids at the collision point.
            SpawnSmallAsteroids(collision.contacts[0].point);
        }
    }

    private void SpawnSmallAsteroids(Vector3 spawnPosition)
    {
        for (int i = 0; i < numSmallAsteroidsToSpawn; i++)
```

```
        {
            // Instantiate small asteroid prefab at the collision point.
            GameObject smallAsteroidInstance = Instantiate(smallAsteroidPrefab,
spawnPosition, Quaternion.identity);

            // Destroy the small asteroids after a short delay.
            Destroy(smallAsteroidInstance, 2f); // Adjust the delay as needed.
        }
    }
}
```

Spaceship

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

public class Spaceship : MonoBehaviour
{
    public float upForce = 8f;  // Adjust the force for forward acceleration.
    public float rotationSpeed = 5f;  // Adjust the rotation speed.

    private Rigidbody rb;

    private void Start()
    {
        // Get the Rigidbody component attached to the spaceship.
        rb = GetComponent<Rigidbody>();

        InvokeRepeating("CheckScreenEdges", 0f, 0.2f);
    }

    private void Update()
    {
        // Check for user input in the Update method.
        // Use GetKey or GetKey(KeyCode) for detecting if keys are held down.

        // Accelerate forward when the Up arrow is held.
        if (Input.GetKey(KeyCode.UpArrow))
        {
            // Apply a forward force to the spaceship.
            rb.AddForce(transform.up * upForce);
        }

        // Rotate left when the Left arrow is held.
        if (Input.GetKey(KeyCode.LeftArrow))
        {
            Vector3 currentRotation = transform.rotation.eulerAngles;
            currentRotation.y -= 100 * Time.deltaTime;
            transform.rotation = Quaternion.Euler(currentRotation);
        }

        // Rotate right when the Right arrow is held.
        if (Input.GetKey(KeyCode.RightArrow))
        {
            Vector3 currentRotation = transform.rotation.eulerAngles;
            currentRotation.y += 100 * Time.deltaTime;
            transform.rotation = Quaternion.Euler(currentRotation);
        }
```

```
    }

    void CheckScreenEdges()
    {
        Debug.Log("Current position: " + transform.position);
        // Check if the asteroid has left the screen
        if (Mathf.Abs(transform.position.x) > 20f ||
Mathf.Abs(transform.position.z) > 15f)
        {
            // Wrap around to the opposite side
            transform.position = new Vector3(-transform.position.x, 0, -
transform.position.z);
        }
    }
}
```