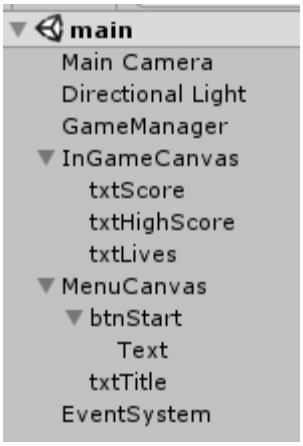
CT3536 Unity3D Lab 7
Sample Solution




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

public class GameManager : MonoBehaviour {

    public enum GAMESTATES {
        MENU,
        PLAYING
    }

    // inspector settings
    public GameObject asteroidPrefab, spaceshipPrefab;
    public Camera camera;
    public GameObject menuCanvas, inGameCanvas;
    public TMP_Text txtScore, txtHighScore, txtLives;

    // class-level statics
    public static GameManager instance;
    public static int currentGameLevel;
    public static Vector3 screenBottomLeft, screenTopRight;
    public static float screenWidth, screenHeight;
    public static GAMESTATES gameState = GAMESTATES.MENU;
    public static int score, highscore=0, lives;

    void Start () {
        instance = this;
        camera.transform.position = new Vector3 (0f, 30f, 0f);
        camera.transform.LookAt (Vector3.zero, new Vector3 (0f, 0f, 1f));
        SetGameState (GAMESTATES.MENU);
    }

    public static void SetGameState(GAMESTATES state) {
        gameState = state;
        switch (state) {
            case GAMESTATES.MENU:
                instance.inGameCanvas.SetActive (false);
                instance.menuCanvas.SetActive (true);
            break;

            case GAMESTATES.PLAYING:
                instance.inGameCanvas.SetActive (true);
                instance.menuCanvas.SetActive (false);
                StartNewGame ();
            break;
        }
    }

    public static void StartNewGame() {
        currentGameLevel = 0;
        SetScore (0);
        SetLives (3);
        Asteroid.DestroyAllAsteroids ();

        StartNextLevel ();
        CreatePlayerSpaceship ();
    }

    public static void StartNextLevel() {
        currentGameLevel++;

        // find screen corners and size, in world coordinates
        // for ViewportToWorldPoint, the z value specified is in world units from the camera
        screenBottomLeft = instance.camera.ViewportToWorldPoint(new Vector3(0f,0f,30f));
        screenTopRight = instance.camera.ViewportToWorldPoint (new Vector3(1f,1f,30f));
        screenWidth = screenTopRight.x - screenBottomLeft.x;
        screenHeight = screenTopRight.z - screenBottomLeft.z;
        // instantiate some asteroids near the edges of the screen
        for (int i = 0; i < currentGameLevel + 1; 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.GetComponent<Asteroid> ().SetScale (0.08f, 0.12f);
        }

    }

    public static void CreatePlayerSpaceship() {
        // instantiate the player's spaceship
        GameObject go = Instantiate (instance.spaceshipPrefab) as GameObject;
        go.transform.position = Vector3.zero;
    }

    public static void SetScore(int sc) {
        score = sc;
        instance.txtScore.text = "Score: "+score;
        if (score > highscore) {
            highscore = score;
            instance.txtHighScore.text = "Best Score: "+highscore;
        }
    }

    public static void SetLives(int li) {
        lives = li;
        instance.txtLives.text = "Lives: "+lives;
    }

    public void OnClickedStartButton() {
        SetGameState (GAMESTATES.PLAYING);
    }

}
```

```
public class Asteroid : MonoBehaviour {

    // inspector settings
    public Rigidbody rigidBody;

    // class-level statics
    private static List<GameObject> asteroids = new List<GameObject>();

    // Use this for initialization
    void Start () {
        // randomise velocity
        rigidBody.velocity = new Vector3 (Random.Range (-10f, 10f), 0f, Random.Range (-10f, 10f));
        rigidBody.angularVelocity = new Vector3 (Random.Range (-4f, 4f), Random.Range (-
4f, 4f), Random.Range (-4f, 4f));
        asteroids.Add (this.gameObject);
    }

    public void SetScale(float min, float max) {
        transform.localScale = new Vector3(Random.Range(min,max), Random.Range(min,max), Random.Range(min,max
));
        rigidBody.mass = transform.localScale.x * transform.localScale.y * transform.localScale.z;
    }

    void OnCollisionEnter(Collision collision) {
        if (!collision.gameObject.name.Contains("asteroid")) {
            Spaceship ss = collision.gameObject.GetComponent<Spaceship> ();
            if (ss != null && ss.isInvulnerable)
                return;

            // we've collided with something other than another asteroid
            GameManager.SetScore(GameManager.score + 100);
            asteroids.Remove (this.gameObject);
            Destroy(collision.gameObject); // if it's the player spaceship, the Spaceship scripts' OnDestroy
will look after re-creating it
            Destroy(this.gameObject);
```

```
            if (rigidBody.mass > 0.0003f) {
                float minScale = rigidBody.mass * 50f;
                float maxScale = minScale * 1.1f;
                for (int i = 0; i < 3; i++) {
                    GameObject go = Instantiate (GameManager.instance.asteroidPrefab) as GameObject;
                    go.transform.position = transform.position;
                    go.GetComponent<Asteroid> ().SetScale (minScale, maxScale);


                }
            }
            else if (asteroids.Count == 0)
                GameManager.StartNextLevel ();

        }
    }

    public static void DestroyAllAsteroids() {
        foreach (GameObject go in asteroids)
            Destroy (go);
        asteroids.Clear ();
    }
}
```

```
public class Spaceship : MonoBehaviour {

    // inspector settings
    public Rigidbody rigidBody;
    public GameObject bulletPrefab;

    // public member data
    [HideInInspector] public bool isInvulnerable = true;

    // private member data
    private float lastFiredTime = 0f;

    void Start() {
        Invoke ("MakeVulnerable", 2f);
    }

    private void MakeVulnerable() {
        isInvulnerable = false;
    }

    void FixedUpdate () {
        if (Input.GetKey (KeyCode.UpArrow)) {
                rigidBody.AddForce (transform.forward * (rigidBody.mass * Time.deltaTime * 500f));
                if (!flameParticleSystem.isPlaying)
                        flameParticleSystem.Play ();
        }
        else if (flameParticleSystem.isPlaying)
                flameParticleSystem.Stop ();

        if (Input.GetKey(KeyCode.LeftArrow))
                rigidBody.AddTorque(-transform.up * (rigidBody.mass * Time.deltaTime * 500f));
        else if (Input.GetKey(KeyCode.RightArrow))
                rigidBody.AddTorque(transform.up * (rigidBody.mass * Time.deltaTime * 500f));

        // firing is only allowed at most once per 0.25 seconds
        if (Input.GetKey (KeyCode.Space) && lastFiredTime + 0.25f <= Time.time) {
                lastFiredTime = Time.time;
                FireBullet ();
        }
    }

    void OnDestroy() {
        if (GameManager.lives > 1) {
            GameManager.CreatePlayerSpaceship ();
            GameManager.SetLives (GameManager.lives - 1);
        }
        else
            GameManager.SetGameState (GameManager.GAMESTATES.MENU);
    }

    private void FireBullet() {
```

```
        GameObject go = Instantiate (bulletPrefab) as GameObject;
        go.transform.position = transform.position + transform.forward*2.5f;
        go.transform.rotation = transform.rotation;
    }
}
```

```
public class SpeedLimiter : MonoBehaviour {

    // inspector settings
    public Rigidbody rigid;
    public float speedLimit = 5f;

    void FixedUpdate () {
        float spd = rigid.velocity.magnitude;
        if (spd > speedLimit)
            rigid.velocity *= speedLimit / spd;
    }
}
```

```
public class Bullet : MonoBehaviour {

    // inspector settings
    public Rigidbody rigid;

    void Start () {
        rigid.velocity = transform.forward * 30f;
    }
}
```

```
public class ScreenEdgeChecker : MonoBehaviour {

    // inspector settings
    public Rigidbody rigidBody;
    public bool destroyWhenOffscreen = false;

    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) {
            if (destroyWhenOffscreen)
                Destroy (this.gameObject);
            else
                transform.position = new Vector3 (pos.x + xTeleport, 0f, pos.z + zTeleport);
        }

    }
}
```