Code Examples

Random Size

```
1. using UnityEngine;
2.
3. public class RandomSize : MonoBehaviour
4. {
5.     float defaultScale;
6.     [SerializeField] float minScale = 0.5f;
7.     [SerializeField] float maxScale = 1.5f;
8.
9.     private void Awake()
10.    {
11.        defaultScale = transform.localScale.x;
12.    }
13.
14.    private void OnEnable()
15.    {
16.        transform.localScale = transform.localScale * Random.Range(minScale,
       maxScale);
17.    }
18. }
```

Floating Movement

```
1. using UnityEngine;
2.
3. public class FloatingMovement : MonoBehaviour
4. {
```

```
    Rigidbody2D rb;

    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
    }

    private void OnEnable()
    {
        transform.position = Random.insideUnitCircle.normalized * 50;
        rb.AddForce(Random.insideUnitCircle, ForceMode2D.Impulse);
        rb.AddTorque(Random.Range(-10, 10));
    }
}
```

Points

```
using UnityEngine;
public class Points : MonoBehaviour
{
    [SerializeField] int basePoints 100;
    [SerializeField] bool scalePointsBySize;

    public void ScorePoints(int playerNumber)
    {

        int points = basePoints;
        if (scalePointsBySize)
```

```
12.     {
13.         points = Mathf.RoundToInt(basePoints * transform.localScale.x);
14.     }
15.
16.     GameManager.instance.UpdateScore(points, playerNumber);
17. }
18. }
```

Movement

```
1.  using UnityEngine;
2.  using UnityEngine.InputSystem;
3.  using UnityEngine.Events;
4.
5.  public class Movement : MonoBehaviour
6.  {
7.      Rigidbody2D rb;
8.
9.      [SerializeField] float moveSpeed = 5;
10.     [SerializeField] float turnSpeed = 3;
11.     [SerializeField] UnityEvent<float> OnThrust;
12.
13.     float moveAxis;
14.     float turnAxis;
15.
16.     private void Awake()
17.     {
18.         rb = GetComponent<Rigidbody2D>();
```

```
19.  }
20.
21.  void OnMove(InputValue value)
22.  {
23.      moveAxis = value.Get<float>() * moveSpeed;
24.      OnThrust.Invoke(value.Get<float>());
25.  }
26.
27.  void OnTurn(InputValue value)
28.  {
29.      turnAxis = value.Get<float>() * turnSpeed;
30.  }
31.
32.  private void FixedUpdate()
33.  {
34.      rb.AddForce(transform.up * moveAxis);
35.      rb.AddTorque(turnAxis);
36.  }
37.
38.  private void OnDisable()
39.  {
40.      moveAxis = 0;
41.      turnAxis = 0;
42.      rb.velocity = Vector2.zero;
43.  }
44. }
```

Fade Sprite

```
1.  using UnityEngine;
2.
3.  public class FadeSprite : MonoBehaviour
4.  {
5.      [SerializeField] SpriteRenderer sprite;
6.      [SerializeField] float fadeSpeed = 5;
7.      [Tooltip("Automatically turn off the scripts if the sprite's transparency is zero.")]
8.      [SerializeField] bool autoOff = true;
9.
10.     float target;
11.
12.     private void Update()
13.     {
14.         float newValue = Mathf.MoveTowards(sprite.color.a, target, fadeSpeed *
        Time.deltaTime);
15.         SetTransparency(newValue);
16.
17.         if (autoOff && sprite.color.a == 0)
18.         {
19.             enabled = false;
20.         }
21.     }
22.
23.     public void SetSpriteTransparencyValue(float value)
24.     {
```

```
25.     if (autoOff && enabled == false)
26.     {
27.         enabled = true;
28.     }
29.
30.     SetTransparency(value);
31.   }
32.
33.   public void SetSpriteTransparencyTarget(float value)
34.   {
35.
36.     if (autoOff && enabled == false)
37.     {
38.         enabled = true;
39.     }
40.
41.     target = value;
42.   }
43.
44.   void SetTransparency(float value)
45.   {
46.     sprite.color = new Color(sprite.color.r, sprite.color.g, sprite.color.b, value);
47.   }
48.
49.   void OnDisable()
50.   {
```

```
51.     SetTransparency(0);
52.   }
53. }
```

Damageable

```
1.  using UnityEngine;
2.  using UnityEngine.Events;
3.
4.  public class Damageable : MonoBehaviour
5.  {
6.      float hp;
7.      [HideInInspector]
8.      [SerializeField] AudioClip explosionSFX;
9.      [HideInInspector]
10.     [SerializeField] UnityEvent<int> OnDestroyedByPlayer;
11.
12.     private void OnEnable()
13.     {
14.         hp = 100;
15.     }
16.
17.     public bool Damage(float dmg, int attackingPlayer)
18.     {
19.         if (Damage(dmg) == true)
20.         {
21.             OnDestroyedByPlayer.Invoke(attackingPlayer);
22.             return true;
```

```
23.     }
24.
25.     return false;
26.   }
27.
28.   public bool Damage(float dmg)
29.   {
30.     if (!isActiveAndEnabled)
31.     {
32.       return false;
33.     }
34.
35.     hp -= dmg;
36.
37.     if (hp < 0)
38.     {
39.       ExplosionManager.GenerateExplosion(transform.position,
    transform.localScale.x);
40.       AudioManager.instance.PlaySFX(explosionSFX, 1);
41.       gameObject.SetActive(false);
42.       return true;
43.     }
44.
45.     return false;
46.   }
47. }
```

Explosion

```
1.  using System.Collections;
2.  using UnityEngine;
3.
4.  public class Explosion : MonoBehaviour
5.  {
6.      [SerializeField] float duration = 0.5f;
7.      [SerializeField] AnimationCurve curve;
8.
9.      private void OnEnable()
10.     {
11.         StartCoroutine(ShrinkScale());
12.     }
13.
14.     IEnumerator ShrinkScale()
15.     {
16.         float startScale = transform.localScale.x;
17.         float scale;
18.         float timer = 0;
19.
20.         while (timer < duration)
21.         {
22.             float t = timer / duration;
23.             scale = Mathf.Lerp(startScale, 0, curve.Evaluate(t));
24.             transform.localScale = Vector3.one * scale;
25.             timer += Time.deltaTime;
```

```
            yield return null;
        }

        transform.localScale = Vector3.zero;
        gameObject.SetActive(false);
    }
}
```

Weapon Base Class

```
using UnityEngine;

public abstract class Weapon : MonoBehaviour
{
    protected Player player;
    public string weaponName;

    protected virtual void Awake()
    {
        player = transform.root.GetComponentInChildren<Player>();
    }

    public abstract void Fire(Transform origin);
}
```

Object Pool

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

```
4.
5.  public class ObjectPool : MonoBehaviour
6.  {
7.      [SerializeField] GameObject poolObject;
8.      [SerializeField] int numToCreate = 20;
9.      List<GameObject> createdObjects = new();
10.     int index;
11.
12.     protected virtual void Awake()
13.     {
14.         for (int i = 0; i < numToCreate; i++)
15.         {
16.             CreateObject();
17.         }
18.     }
19.
20.     public void SetPool(GameObject poolObject, int numToCreate)
21.     {
22.         this.poolObject = poolObject;
23.         this.numToCreate = numToCreate;
24.     }
25.
26.     protected virtual GameObject CreateObject()
27.     {
28.         GameObject createdObject = Instantiate(poolObject, transform);
29.         createdObjects.Add(createdObject);
```

```
30.      return createdObject;
31.  }
32.
33.  public GameObject GetObject()
34.  {
35.      for (int i = 0; i < createdObjects.Count; i++)
36.      {
37.          if (createdObjects[index].activeInHierarchy == false);
38.          {
39.              return createdObjects[index];
40.          }
41.
42.          index++;
43.          if (index >= createdObjects.Count)
44.          {
45.              index = 0;
46.          }
47.      }
48.
49.      return CreateObject();
50.  }
51. }
```

Projectile Pool

```
1.  using UnityEngine;
2.
3.  public class ProjectilePool : ObjectPool
```

```
4.  {
5.      [HideInInspector]
6.      public int playerNumber;
7.
8.      protected override GameObject CreateObject()
9.      {
10.         GameObject createdObject = base.CreateObject();
11.         if ((createdObject.TryGetComponent(out Projectile projectile)))
12.         {
13.             projectile.playerNumber = playerNumber;
14.         }
15.         return createdObject;
16.     }
17.
18.     public void GenerateProjectile(Vector3 position, Quaternion rotation)
19.     {
20.         GameObject newProjectile = GetObject();
21.         newProjectile.transform.position = position;
22.         newProjectile.transform.rotation = rotation;
23.         newProjectile.SetActive(true);
24.     }
25. }
```

Projectile

```
1.  using System.Collections;
2.  using System.Collections.Generic;
3.  using UnityEngine;
```

```
4.
5.  public class Projectile : MonoBehaviour
6.  {
7.      public int playerNumber;
8.
9.      [SerializeField] float damage = 30;
10.     [SerializeField] float speed = 30;
11.     [SerializeField] float duration = 5;
12.
13.     [SerializeField] AudioClip fireClip;
14.     [SerializeField] AudioClip hitClip;
15.
16.     Collider2D[] hitColliders = new Collider2D[5];
17.
18.     float timer;
19.
20.     private void OnEnable()
21.     {
22.         AudioManager.instance.PlaySFX(fireClip, 1);
23.     }
24.
25.     void Update()
26.     {
27.         // move
28.         transform.position += transform.up * speed * Time.deltaTime;
29.
```

```
30.     // check if hit
31.     int numHit = Physics2D.OverlapBoxNonAlloc(transform.position,
    transform.localScale, transform.rotation.z, hitColliders);
32.
33.     if (numHit > 0)
34.     {
35.         for (int i = 0; i < numHit; i++)
36.         {
37.             if (hitColliders[i].TryGetComponent(out Damageable damageable))
38.             {
39.                 damageable.Damage(damage, playerNumber);
40.
41.                 ExplosionManager.GenerateExplosion(transform.position,
    transform.localScale.x);
42.                 AudioManager.instance.PlaySFX(hitClip, 1);
43.                 gameObject.SetActive(false);
44.                 break;
45.             }
46.         }
47.     }
48.
49.     // advance the timer
50.     timer += Time.deltaTime;
51.
52.     // disable
53.     if (timer > duration)
54.     {
```

```
            gameObject.SetActive(false);
        }
    }

    private void OnDisable()
    {
        timer = 0;
    }
}
```

Projectile Weapon

```
using UnityEngine;

public class ProjectileWeapon : Weapon
{
    [SerializeField] GameObject projectile;
    ProjectilePool projectiles;

    [SerializeField] float offset = 0.75f;

    protected override void Awake()
    {
        base.Awake();
        GameObject projectileContainer = new GameObject(projectile.name + " Pool");
        projectileContainer.transform.parent = transform.root;
        projectileContainer.SetActive(false);
        projectiles = projectileContainer.AddComponent<ProjectilePool>();
```

```
17.       projectiles.SetPool(projectile, 30);
18.       projectiles.playerNumber = player.Number;
19.       projectileContainer.SetActive(true);
20.   }
21.
22.   public override void Fire(Transform origin)
23.   {
24.       projectiles.GenerateProjectile(origin.position + (origin.transform.up * offset),
      origin.rotation);
25.   }
26. }
```

Laser Weapon

```
1.  using UnityEngine;
2.  using UnityEngine.Events;
3.
4.  public class LaserWeapon : Weapon
5.  {
6.      [SerializeField] UnityEvent OnLaserFire;
7.
8.      [SerializeField] float damage = 10;
9.      [SerializeField] float offset = 0.75f;
10.
11.   [SerializeField] AudioClip fireClip;
12.   [SerializeField] AudioClip hitClip;
13.
14.   ContactFilter2D filter = new();
```

```
15.    RaycastHit2D[] hitColliders = new RaycastHit2D[5];
16.
17.    public override void Fire(Transform origin)
18.    {
19.        int numHit = Physics2D.Raycast(origin.position + (origin.up * offset), origin.up,
       filter.NoFilter(), hitColliders);
20.        OnLaserFire.Invoke();
21.        AudioManager.instance.PlaySFX(fireClip, 1);
22.
23.        if (numHit > 0)
24.        {
25.            for (int i = 0; i < numHit; i++)
26.            {
27.                if (hitColliders[i].collider.TryGetComponent(out Damageable damageable))
28.                {
29.                    damageable.Damage(damage, player.Number);
30.
31.                    ExplosionManager.GenerateExplosion(hitColliders[i].point, .2f);
32.                    AudioManager.instance.PlaySFX(hitClip, 1);
33.                    break;
34.                }
35.            }
36.        }
37.    }
38. }
```

Player Profile

```
1. using System.Collections;
2. using System.Collections.Generic;
3. using UnityEngine;
4.
5. [CreateAssetMenu]
6. public class PlayerProfile : ScriptableObject
7. {
8.     public GameObject playerObject;
9.     public Color playerColour;
10.    public string playerName;
11. }
```

Game Manager

```
1. using System.Collections;
2. using UnityEngine;
3. using UnityEngine.SceneManagement;
4. using System;
5.
6. public class GameManager : MonoBehaviour
7. {
8.     public static event Action<PlayerProfile> OnPlayerCreated;
9.     public static event Action<PlayerProfile, int> OnPlayerKilled;
10.    public static event Action<PlayerProfile, int> OnScoreChanged;
11.    public static event Action<PlayerProfile> OnPlayerWin;
12.
13.    public static GameManager instance;
14.
```

```
15.  [SerializeField] ObjectPool[] asteroidPools;
16.
17.  [SerializeField] float reSpawnDuration = 2;
18.
19.  bool gameOver;
20.
21.  int[] scores = new int[2];
22.
23.  [SerializeField] bool twoPlayerGame = true;
24.  [SerializeField] float positionOffset = 3;
25.
26.  [SerializeField] PlayerProfile[] players;
27.
28.  private void Awake()
29.  {
30.      if (instance != null && instance != this)
31.      {
32.          Destroy(this);
33.          return;
34.      }
35.
36.      instance = this;
37.  }
38.
39.  IEnumerator Start()
40.  {
```

```
41.     // Set up the players
42.
43.     Player.totalPlayers = 0;
44.     int numPlayers = twoPlayerGame ? 2 : 1;
45.
46.     for (int i = 0; i < numPlayers; i++)
47.     {
48.         GameObject player = Instantiate(players[i].playerObject);
49.
50.         if (twoPlayerGame)
51.         {
52.             float xOffset = (i == 0) ? -positionOffset : positionOffset;
53.             player.transform.position = Vector3.right * xOffset;
54.         }
55.         else {
56.             player.transform.position = Vector3.zero;
57.         }
58.
59.         player.GetComponentInChildren<SpriteRenderer>().color =
    players[i].playerColour;
60.         OnPlayerCreated?.Invoke(players[i]);
61.     }
62.
63.     while (!gameOver)
64.     {
65.         SpawnAsteroid();
```

```
66.         yield return new WaitForSeconds(UnityEngine.Random.Range(1, 5));
67.     }
68.
69.     // end the game
70.
71.     if (twoPlayerGame)
72.     {
73.         int highestScore = 0;
74.         int winningPlayer = 0;
75.
76.         for (int i = 0; i < numPlayers; i++)
77.         {
78.             if (scores[i] > highestScore)
79.             {
80.                 highestScore = scores[i];
81.                 winningPlayer = i;
82.             }
83.         }
84.
85.         OnPlayerWin?.Invoke(players[winningPlayer]);
86.     }
87.
88.     yield return new WaitForSeconds(5);
89.
90.     SceneManager.LoadScene(SceneManager.GetActiveScene().name);
91. }
```

```
92.
93.    void SpawnAsteroid()
94.    {
95.        int i = UnityEngine.Random.Range(0, asteroidPools.Length);
96.
97.        GameObject newAsteroid = asteroidPools[i].GetObject();
98.        newAsteroid.SetActive(true);
99.    }
100.
101.            public void ReportPlayerDeath(GameObject player, int playerNumber, int
    lives)
102.            {
103.                OnPlayerKilled?.Invoke(players[playerNumber], lives);
104.
105.                if (lives > 0)
106.                {
107.                    StartCoroutine(ReEnablePlayer(player));
108.                    return;
109.                }
110.
111.                if (lives <= 0)
112.                {
113.                    Player.totalPlayers--;
114.
115.                    if (Player.totalPlayers == 0)
116.                    {
```

```
                    gameOver = true;
                }
            }
        }

        IEnumerator ReEnablePlayer(GameObject player)
        {
            yield return new WaitForSeconds(reSpawnDuration);
            player.transform.position = Vector3.zero;
            player.SetActive(true);
        }

        public void UpdateScore(int pointsToAdd, int playerNumber)
        {
            scores[playerNumber] += pointsToAdd;
            OnScoreChanged?.Invoke(players[playerNumber], scores[playerNumber]);
        }
    }
```

Player UI

```
using UnityEngine;
using UnityEngine.UI;

public class PlayerUI : MonoBehaviour
{
    [SerializeField] PlayerProfile myPlayer;
```

```
7.    [SerializeField] Text livesDisplay;
8.    [SerializeField] Text scoreDisplay;
9.    [SerializeField] Image winScreen;
10.
11.   void SetUpUI(PlayerProfile playerProfile)
12.   {
13.      if (playerProfile != myPlayer)
14.      {
15.         return;
16.      }
17.
18.      SetPlayerColor(playerProfile.playerColour);
19.      ActivateUI(true);
20.   }
21.
22.   void ActivateUI(bool active)
23.   {
24.      livesDisplay.gameObject.SetActive(active);
25.      scoreDisplay.gameObject.SetActive(active);
26.   }
27.
28.   public void SetPlayerColor(Color playerColor)
29.   {
30.      livesDisplay.color = playerColor;
31.      scoreDisplay.color = playerColor;
32.      winScreen.color = playerColor;
```

```
33.  }
34.
35.  public void UpdateLives(PlayerProfile playerProfile, int lives)
36.  {
37.      if (playerProfile != myPlayer)
38.      {
39.          return;
40.      }
41.
42.      switch (lives)
43.      {
44.          case 3:
45.              livesDisplay.text = "^ ^ ^";
46.              break;
47.          case 2:
48.              livesDisplay.text = "^ ^";
49.              break;
50.          case 1:
51.              livesDisplay.text = "^";
52.              break;
53.          default:
54.              livesDisplay.text = "";
55.              break;
56.      }
57.  }
58.
```

```
59.  public void UpdateScore(PlayerProfile playerProfile, int score)
60.  {
61.      if (playerProfile != myPlayer)
62.      {
63.          return;
64.      }
65.
66.      scoreDisplay.text = string.Format("{0:000000}", score);
67.  }
68.
69.  void DisplayWinScreen(PlayerProfile playerProfile)
70.  {
71.      if (playerProfile != myPlayer)
72.      {
73.          return;
74.      }
75.
76.      winScreen.gameObject.SetActive(true);
77.  }
78.
79.  private void OnEnable()
80.  {
81.      ActivateUI(false);
82.      GameManager.OnPlayerCreated += SetUpUI;
83.      GameManager.OnScoreChanged += UpdateScore;
84.      GameManager.OnPlayerKilled += UpdateLives;
```

```
85.        GameManager.OnPlayerWin += DisplayWinScreen;
86.    }
87.
88.    private void OnDisable()
89.    {
90.        GameManager.OnPlayerCreated -= SetUpUI;
91.        GameManager.OnScoreChanged -= UpdateScore;
92.        GameManager.OnPlayerKilled -= UpdateLives;
93.        GameManager.OnPlayerWin -= DisplayWinScreen;
94.    }
95. }
```