Code Examples

Asteroid

```
1.  using UnityEngine;
2.
3.  public class Asteroid : MonoBehaviour
4.  {
5.      Rigidbody2D rb;
6.      [HideInInspector] public int pointsValue;
7.      float defaultScale;
8.
9.      private void Awake()
10.     {
11.         rb = GetComponent<Rigidbody2D>();
12.         defaultScale = transform.localScale.x;
13.     }
14.
15.     private void OnEnable()
16.     {
17.         transform.localScale = transform.localScale * Random.Range(0.5f, 1.5f);
18.         pointsValue = Mathf.RoundToInt(100 * transform.localScale.x);
19.
20.         transform.position = Random.insideUnitCircle.normalized * 50;
21.         rb.AddForce(Random.insideUnitCircle, ForceMode2D.Impulse);
22.         rb.AddTorque(Random.Range(-10, 10));
23.     }
24.
```

```
    private void OnDisable()
    {
        transform.localScale = Vector3.one * defaultScale;
    }
}
```

Audio Manager

```
using UnityEngine;

public class AudioManager : MonoBehaviour
{
    public static AudioManager instance;

    AudioSource audioSource;

    private void Awake()
    {
        if (instance != null && instance != this)
        {
            Destroy(this);
            return;
        }

        instance = this;
        audioSource = GetComponent<AudioSource>();
    }

```

```
    public void PlaySFX(AudioClip clip, float volumeScale)
    {
        audioSource.PlayOneShot(clip, volumeScale);
    }
}
```

Damageable

```
using UnityEngine;

public class Damageable : MonoBehaviour
{
    float hp;

    private void OnEnable()
    {
        hp = 100;
    }

    public bool Damage(float dmg)
    {
        if (!isActiveAndEnabled)
        {
            return false;
        }

        hp -= dmg;
```

```
        if (hp < 0)
        {
            GameObject explosion = GameManager.instance.explosionPool.GetObject();
            explosion.transform.position = transform.position;
            explosion.transform.localScale = transform.localScale;
            explosion.SetActive(true);
            gameObject.SetActive(false);
            return true;
        }

        return false;
    }
}
```

Explosion

```
using System.Collections;
using UnityEngine;

public class Explosion : MonoBehaviour
{
    [SerializeField] float duration = 0.5f;
    [SerializeField] AnimationCurve curve;
    [SerializeField] AudioClip explosionSFX;

    private void OnEnable()
    {
        StartCoroutine(ShrinkScale());
```

```
    }

    IEnumerator ShrinkScale()
    {
        AudioManager.instance.PlaySFX(explosionSFX, 1);
        float startScale = transform.localScale.x;
        float scale;
        float timer = 0;

        while (timer < duration)
        {
            float t = timer / duration;
            scale = Mathf.Lerp(startScale, 0, curve.Evaluate(t));
            transform.localScale = Vector3.one * scale;
            timer += Time.deltaTime;
            yield return null;
        }

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

Game Manager

```
using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;
```

```
4.
5.  public class GameManager : MonoBehaviour
6.  {
7.      public static GameManager instance;
8.
9.      public ObjectPool explosionPool;
10.     [SerializeField] ObjectPool[] asteroidPools;
11.     public ObjectPool[] projectilePools;
12.
13.     [SerializeField] float reSpawnDuration = 2;
14.
15.     bool gameOver;
16.
17.     [SerializeField] PlayerUI[] playerUI;
18.     int[] scores = new int[2];
19.
20.     [SerializeField] bool twoPlayerGame = true;
21.     [SerializeField] float positionOffset = 3;
22.     [SerializeField] GameObject playerObject;
23.     [SerializeField] Color[] playerColors;
24.     [SerializeField] GameObject[] winScreens;
25.
26.     private void Awake()
27.     {
28.         if (instance != null && instance != this)
29.         {
```

```
30.         Destroy(this);
31.         return;
32.     }
33.
34.     instance = this;
35. }
36.
37. IEnumerator Start()
38. {
39.     // Set up the players
40.
41.     Player.totalPlayers = 0;
42.     int numPlayers = twoPlayerGame ? 2 : 1;
43.
44.     for (int i = 0; i < numPlayers; i++)
45.     {
46.         GameObject player = Instantiate(playerObject);
47.
48.         if (twoPlayerGame)
49.         {
50.             float xOffset = (i == 0) ? -positionOffset : positionOffset;
51.             player.transform.position = Vector3.right * xOffset;
52.         }
53.         else {
54.             player.transform.position = Vector3.zero;
55.         }
```

```
56.
57.         player.GetComponentInChildren<SpriteRenderer>().color = playerColors[i];
58.         playerUI[i].SetPlayerColor(playerColors[i]);
59.         playerUI[i].gameObject.SetActive(true);
60.     }
61.
62.     while (!gameOver)
63.     {
64.         SpawnAsteroid();
65.         yield return new WaitForSeconds(Random.Range(1, 5));
66.     }
67.
68.     // end the game
69.
70.     if (twoPlayerGame)
71.     {
72.         int highestScore = 0;
73.         int winningPlayer = 0;
74.
75.         for (int i = 0; i < numPlayers; i++)
76.         {
77.             if (scores[i] > highestScore)
78.             {
79.                 highestScore = scores[i];
80.                 winningPlayer = i;
81.             }
```

```
82.         }
83.
84.         winScreens[winningPlayer].GetComponent<UnityEngine.UI.Image>().color =
    playerColors[winningPlayer];
85.         winScreens[winningPlayer].SetActive(true);
86.     }
87.
88.     yield return new WaitForSeconds(5);
89.
90.     SceneManager.LoadScene(SceneManager.GetActiveScene().name);
91. }
92.
93. void SpawnAsteroid()
94. {
95.     int i = 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.                playerUI[playerNumber].UpdateLives(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.                {
117.                    gameOver = true;
118.                }
119.            }
120.        }
121.
122.        IEnumerator ReEnablePlayer(GameObject player)
123.        {
124.            yield return new WaitForSeconds(reSpawnDuration);
125.            player.transform.position = Vector3.zero;
126.            player.SetActive(true);
127.        }
128.
129.        public void UpdateScore(int pointsToAdd, int playerNumber)
130.        {
131.            scores[playerNumber] += pointsToAdd;
132.            playerUI[playerNumber].UpdateScore(scores[playerNumber]);
```

```
            }
        }
```

Infinite Bounds

```
using UnityEngine;

public class InfiniteBounds : MonoBehaviour
{
    Rigidbody2D rb;
    Camera cam;

    private void Awake()
    {
        rb = GetComponent<Rigidbody2D>();
        cam = Camera.main;
    }

    private void FixedUpdate()
    {
        float sizeBuffer = transform.localScale.x + 0.5f;
        Vector3 topRightCorner = cam.ViewportToWorldPoint(new Vector3(1, 1, 0));
        Vector3 bottomLeftCorner = cam.ViewportToWorldPoint(new Vector3(0, 0, 0));

        bool moveObject = false;
        Vector3 newPosition = rb.position;

        if (newPosition.x > (topRightCorner.x + sizeBuffer))
```

```
24.     {
25.         newPosition.x = bottomLeftCorner.x - sizeBuffer;
26.         moveObject = true;
27.     }
28.
29.     if (newPosition.x < (bottomLeftCorner.x - sizeBuffer))
30.     {
31.         newPosition.x = topRightCorner.x + sizeBuffer;
32.         moveObject = true;
33.     }
34.
35.     if (newPosition.y > (topRightCorner.y + sizeBuffer))
36.     {
37.         newPosition.y = bottomLeftCorner.y - sizeBuffer;
38.         moveObject = true;
39.     }
40.
41.     if (newPosition.y < (bottomLeftCorner.y - sizeBuffer))
42.     {
43.         newPosition.y = topRightCorner.y + sizeBuffer;
44.         moveObject = true;
45.     }
46.
47.     if (moveObject)
48.     {
49.         rb.position = newPosition;
```

```
    }
}
}
```

Invincible On Enable

```
using System.Collections;
using UnityEngine;

public class InvincibleOnEnable : MonoBehaviour
{
    [SerializeField] float duration = 3;
    [SerializeField] float flashInterval = 0.25f;
    [SerializeField] SpriteRenderer spriteRenderer;
    [SerializeField] Damageable damageable;

    bool firstEnable = true;

    private void OnEnable()
    {
        if (firstEnable)
        {
            firstEnable = false;
            return;
        }

        StartCoroutine(Invincible());
    }
```

```
23.
24.  IEnumerator Invincible()
25.  {
26.      damageable.enabled = false;
27.      float timer = 0;
28.      float interval = 0;
29.      bool visible = false;
30.
31.      while (timer < duration)
32.      {
33.          if (interval > flashInterval)
34.          {
35.              spriteRenderer.enabled = visible;
36.              visible = !visible;
37.              interval -= flashInterval;
38.          }
39.
40.          interval += Time.deltaTime;
41.          timer += Time.deltaTime;
42.          yield return null;
43.      }
44.
45.      spriteRenderer.enabled = true;
46.      damageable.enabled = true;
47.  }
48.
```

```
    private void OnDisable()
    {
        spriteRenderer.enabled = true;
        damageable.enabled = true;
    }
}
```

Movement

```
using UnityEngine;
using UnityEngine.InputSystem;

public class Movement : MonoBehaviour
{
    Rigidbody2D rb;
    [SerializeField] SpriteRenderer fireSprite;

    [SerializeField] float moveSpeed = 5;
    [SerializeField] float turnSpeed = 3;

    float moveAxis;
    float turnAxis;

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

```

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

```
45.     if (fireSprite)
46.     {
47.         fireSprite.color = new Color(fireSprite.color.r, fireSprite.color.g,
        fireSprite.color.b, 0);
48.     }
49.   }
50. }
```

Object Pool

```
1.  using System.Collections.Generic;
2.  using UnityEngine;
3.
4.  public class ObjectPool : MonoBehaviour
5.  {
6.      [SerializeField] GameObject poolObject;
7.      [SerializeField] int numToCreate = 20;
8.      List<GameObject> createdObjects = new();
9.      int index;
10.
11.     void Awake()
12.     {
13.         for (int i = 0; i < numToCreate; i++)
14.         {
15.             CreateObject();
16.         }
17.     }
18.
```

```
19.  GameObject CreateObject()
20.  {
21.      GameObject createdObject = Instantiate(poolObject, transform);
22.      createdObjects.Add(createdObject);
23.      return createdObject;
24.  }
25.
26.  public GameObject GetObject()
27.  {
28.      for (int i = 0; i < createdObjects.Count; i++)
29.      {
30.          if (createdObjects[index].activeInHierarchy == false)
31.          {
32.              return createdObjects[index];
33.          }
34.
35.          index++;
36.          if (index >= createdObjects.Count)
37.          {
38.              index = 0;
39.          }
40.      }
41.
42.      return CreateObject();
43.  }
44. }
```

Player

```
1.  using UnityEngine;
2.
3.  public class Player : MonoBehaviour
4.  {
5.      public static int totalPlayers;
6.      public int Number { get; private set; }
7.      int lives = 3;
8.
9.      Damageable damageable;
10.
11.     private void Awake()
12.     {
13.         Number = totalPlayers;
14.         totalPlayers++;
15.
16.         damageable = GetComponent<Damageable>();
17.     }
18.
19.     private void OnCollisionEnter2D(Collision2D collision)
20.     {
21.         if (collision.gameObject.CompareTag("Asteroid"))
22.         {
23.             if (damageable.Damage(200))
24.             {
25.                 lives--;
```

```
            GameManager.instance.ReportPlayerDeath(gameObject, Number, lives);
        }
    }
}
```

Player UI

```
using UnityEngine;
using UnityEngine.UI;

public class PlayerUI : MonoBehaviour
{
    [SerializeField] Text livesDisplay;
    [SerializeField] Text scoreDisplay;

    public void SetPlayerColor(Color playerColor)
    {
        livesDisplay.color = playerColor;
        scoreDisplay.color = playerColor;
    }

    public void UpdateLives(int lives)
    {
        switch (lives)
        {
            case 3:
                livesDisplay.text = "^ ^ ^";
```

```
            break;
        case 2:
            livesDisplay.text = "^ ^";
            break;
        case 1:
            livesDisplay.text = "^";
            break;
        default:
            livesDisplay.text = "";
            break;
    }
}

public void UpdateScore(int score)
{
    scoreDisplay.text = string.Format("{0:000000}", score);
}
}
```

Projectile

```
using UnityEngine;

public class Projectile : MonoBehaviour
{
    [SerializeField] int playerNumber;

    [SerializeField] float damage = 30;
```

```
8.      [SerializeField] float speed = 30;
9.      [SerializeField] float duration = 5;
10.
11.     [SerializeField] AudioClip fireClip;
12.     [SerializeField] AudioClip hitClip;
13.
14.     Collider2D[] hitColliders = new Collider2D[5];
15.
16.     float timer;
17.
18.     private void OnEnable()
19.     {
20.         AudioManager.instance.PlaySFX(fireClip, 1);
21.     }
22.
23.     void Update()
24.     {
25.         // move
26.         transform.position += transform.up * speed * Time.deltaTime;
27.
28.         // check if hit
29.         int numHit = Physics2D.OverlapBoxNonAlloc(transform.position,
    transform.localScale, transform.rotation.z, hitColliders);
30.
31.         if (numHit > 0)
32.         {
```

```
33.         for (int i = 0; i < numHit; i++)
34.         {
35.             if (hitColliders[i].TryGetComponent(out Damageable damageable))
36.             {
37.                 if (damageable.Damage(damage))
38.                 {
39.                     if (hitColliders[i].TryGetComponent(out Asteroid asteroid))
40.                     {
41.                         GameManager.instance.UpdateScore(asteroid.pointsValue,
    playerNumber);
42.                     }
43.                 }
44.
45.                 AudioManager.instance.PlaySFX(hitClip, 1);
46.                 gameObject.SetActive(false);
47.                 break;
48.             }
49.         }
50.     }
51.
52.     // advance the timer
53.     timer += Time.deltaTime;
54.
55.     // disable
56.     if (timer > duration)
57.     {
```

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

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

Weapon

```
using UnityEngine;

public class Weapon : MonoBehaviour
{
    int playerNumber;
    [SerializeField] float offset = 0.75f;

    private void Start()
    {
        playerNumber = GetComponent<Player>().Number;
    }

    void OnFire()
    {
        GameObject projectile =
    GameManager.instance.projectilePools[playerNumber].GetObject();
```

```
16.         projectile.transform.position = transform.position + (transform.up * offset);
17.         projectile.transform.rotation = transform.rotation;
18.         projectile.SetActive(true);
19.     }
20. }
```