GameManager_1.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager_1 : MonoBehaviour
6  {
7      [SerializeField] GameObject BallPrefab;
8      [SerializeField] int BallInstanceCount;
9
10     // Start is called before the first frame update
11     void Start()
12     {
13         Spawn();
14     }
15
16     // Update is called once per frame
17     void Update()
18     {
19
20     }
21
22     public void Spawn()
23     {
24         //コルーチンでボールを生成する
25         StartCoroutine(CreateBall(BallInstanceCount));
26     }
27
28     public IEnumerator CreateBall(int count)
29     {
30         //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
31         //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
32         //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
33         for (int i = 0; i < count; i++)
34         {
35             Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
36             Instantiate(BallPrefab, pos, Quaternion.identity);
37             yield return new WaitForSeconds(0.04f);
```

```
38      }
39    }
40  }
41
```

GameManager_2.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager_2 : MonoBehaviour
6  {
7      [SerializeField] GameObject BallPrefab;
8      [SerializeField] int BallInstanceCount;
9      [SerializeField] Sprite[] BallSprites;
10
11
12     // Start is called before the first frame update
13     void Start()
14     {
15         Spawn();
16     }
17
18     // Update is called once per frame
19     void Update()
20     {
21
22     }
23
24     public void Spawn()
25     {
26         //コルーチンでボールを生成する
27         StartCoroutine(CreateBall(BallInstanceCount));
28     }
29
30     public IEnumerator CreateBall(int count)
31     {
32         //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
33         //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
34         //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
35         for (int i = 0; i < count; i++)
36         {
37             Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
```

```
38          GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
39
40          //ランダムでボールを設定
41          int ballID = Random.Range(0, BallSprites.Length);
42          ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
43          yield return new WaitForSeconds(0.04f);
44      }
45    }
46  }
47
```

Ball.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Ball : MonoBehaviour
6  {
7      public int BallID;
8  }
9
```

GameManager_3.cs

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

public class GameManager_3 : MonoBehaviour
{
    [SerializeField] GameObject BallPrefab;
    [SerializeField] int BallInstanceCount;
    [SerializeField] Sprite[] BallSprites;


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

    // Update is called once per frame
    void Update()
    {

    }

    public void Spawn()
    {
        //コルーチンでボールを生成する
        StartCoroutine(CreateBall(BallInstanceCount));
    }

    public IEnumerator CreateBall(int count)
    {
        //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
        //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
        //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
        for (int i = 0; i < count; i++)
        {
            Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
```

```
38          GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
39
40          //ランダムでボールを設定
41          int ballID = Random.Range(0, BallSprites.Length);
42          ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
43          ball.GetComponent<Ball>().BallID = ballID;
44          yield return new WaitForSeconds(0.04f);
45      }
46    }
47  }
48
```

GameManager_4.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager_4 : MonoBehaviour
6  {
7      [SerializeField] GameObject BallPrefab;
8      [SerializeField] int BallInstanceCount;
9      [SerializeField] Sprite[] BallSprites;
10
11     private bool isDragging;
12
13     // Start is called before the first frame update
14     void Start()
15     {
16         Spawn();
17     }
18
19     // Update is called once per frame
20     void Update()
21     {
22         if (Input.GetMouseButtonDown(0))
23         {
24             OnDragBigin();
25         }
26         else if (Input.GetMouseButtonUp(0))
27         {
28             OnDragEnd();
29         }
30         else if (isDragging)
31         {
32             OnDragging();
33         }
34     }
35
36     void OnDragBigin()
37     {
```

```
38      Debug.Log("ドラッグ開始");
39
40      //座標変換
41      Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
42      RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
43
44      //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
45      if (hit && hit.collider.GetComponent<Ball>())
46      {
47          Debug.Log("---Ballにhit---");
48      }
49  }
50  void OnDragging()
51  {
52
53  }
54  void OnDragEnd()
55  {
56      Debug.Log("ドラッグ終了");
57  }
58
59  public void Spawn()
60  {
61      //コルーチンでボールを生成する
62      StartCoroutine(CreateBall(BallInstanceCount));
63  }
64
65  public IEnumerator CreateBall(int count)
66  {
67      //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
68      //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
69      //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
70      for (int i = 0; i < count; i++)
71      {
72          Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
73          GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
74
75          //ランダムでボールを設定
76          int ballID = Random.Range(0, BallSprites.Length);
```

```
77            ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
78            ball.GetComponent<Ball>().BallID = ballID;
79            yield return new WaitForSeconds(0.04f);
80        }
81    }
82 }
83
```

GameManager_5.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager_5 : MonoBehaviour
6  {
7      [SerializeField] GameObject BallPrefab;
8      [SerializeField] int BallInstanceCount;
9      [SerializeField] Sprite[] BallSprites;
10     [SerializeField] List<Ball> removeBalls = new List<Ball>();
11
12     private bool isDragging;
13
14     // Start is called before the first frame update
15     void Start()
16     {
17         Spawn();
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23         if (Input.GetMouseButtonDown(0))
24         {
25             OnDragBigin();
26         }
27         else if (Input.GetMouseButtonUp(0))
28         {
29             OnDragEnd();
30         }
31         else if (isDragging)
32         {
33             OnDragging();
34         }
35     }
36
37     void OnDragBigin()
```

```
38    {
39        Debug.Log("ドラッグ開始");
40
41        //座標変換
42        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
43        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
44
45        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
46        if (hit && hit.collider.GetComponent<Ball>())
47        {
48            Debug.Log("---Ballにhit---");
49
50            //Listに登録する
51            Ball ball = hit.collider.GetComponent<Ball>();
52            AddRemoveBall(ball);
53            isDragging = true;
54        }
55    }
56    void OnDragging()
57    {
58        //座標変換
59        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
60        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
61
62        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
63        if (hit && hit.collider.GetComponent<Ball>())
64        {
65            Debug.Log("Ballにhitしたよ!");
66
67            //Listに登録する
68            Ball ball = hit.collider.GetComponent<Ball>();
69            AddRemoveBall(ball);
70        }
71    }
72    void OnDragEnd()
73    {
74        //リストの要素数分、ループ
75        int removeCount = removeBalls.Count;
76        for (int i = 0; i < removeCount; i++)
```

```
77      {
78          //中身を削除
79          Destroy(removeBalls[i].gameObject);
80      }
81
82      //要素自体を削除
83      removeBalls.Clear();
84      Debug.Log("ドラッグ終了");
85      isDragging = false;
86  }
87
88  void AddRemoveBall(Ball ball)
89  {
90      //既に登録されているGameObjectを除き、新たに登録する
91      if (removeBalls.Contains(ball) == false)
92      {
93          //登録
94          removeBalls.Add(ball);
95      }
96  }
97
98  public void Spawn()
99  {
100     //コルーチンでボールを生成する
101     StartCoroutine(CreateBall(BallInstanceCount));
102  }
103
104  public IEnumerator CreateBall(int count)
105  {
106     //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
107     //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
108     //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
109     for (int i = 0; i < count; i++)
110     {
111         Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
112         GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
113
114         //ランダムでボールを設定
115         int ballID = Random.Range(0, BallSprites.Length);
```

```
116             ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
117             ball.GetComponent<Ball>().BallID = ballID;
118             yield return new WaitForSeconds(0.04f);
119         }
120     }
121 }
122
```

GameManager_6.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class GameManager_6 : MonoBehaviour
6  {
7      [SerializeField] GameObject BallPrefab;
8      [SerializeField] int BallInstanceCount;
9      [SerializeField] Sprite[] BallSprites;
10      [SerializeField] List<Ball> removeBalls = new List<Ball>();
11
12     private bool isDragging;
13     private Ball currentDraggingBall;
14
15     // Start is called before the first frame update
16     void Start()
17     {
18         Spawn();
19     }
20
21     // Update is called once per frame
22     void Update()
23     {
24         if (Input.GetMouseButtonDown(0))
25         {
26             OnDragBigin();
27         }
28         else if (Input.GetMouseButtonUp(0))
29         {
30             OnDragEnd();
31         }
32         else if (isDragging)
33         {
34             OnDragging();
35         }
36     }
37
```

```
38    void OnDragBigin()
39    {
40        Debug.Log("ドラッグ開始");
41
42        //座標変換
43        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
44        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
45
46        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
47        if (hit && hit.collider.GetComponent<Ball>())
48        {
49            Debug.Log("---Ballにhit---");
50
51            //Listに登録する
52            Ball ball = hit.collider.GetComponent<Ball>();
53            AddRemoveBall(ball);
54            isDragging = true;
55
56            //最後に触ったボールを記憶
57            currentDraggingBall = ball;
58        }
59    }
60    void OnDragging()
61    {
62        //座標変換
63        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
64        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
65
66        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
67        if (hit && hit.collider.GetComponent<Ball>())
68        {
69            Ball ball = hit.collider.GetComponent<Ball>();
70
71            //最後に触ったボールと今触ったボールが同じIDであるか判別
72            if (ball.BallID == currentDraggingBall.BallID)
73            {
74                //2点の距離が遠くないか判別
75                float distance = Vector2.Distance(ball.transform.position,currentDraggingBall.transform.position);
76                if (distance < 1.5)
```

```
77              {
78                  AddRemoveBall(ball);
79
80                  //最後に触ったボールを記憶
81                  currentDraggingBall = ball;
82              }
83          }
84      }
85  }
86  void OnDragEnd()
87  {
88      //リストの要素数分、ループ
89      int removeCount = removeBalls.Count;
90
91      //3個以上じゃないと消さない
92      if (removeCount >= 3)
93      {
94          for (int i = 0; i < removeCount; i++)
95          {
96              Destroy(removeBalls[i].gameObject);
97          }
98      }
99
100     //要素自体を削除
101     removeBalls.Clear();
102     Debug.Log("ドラッグ終了");
103     isDragging = false;
104 }
105
106 void AddRemoveBall(Ball ball)
107 {
108     //既に登録されているGameObjectを除き、新たに登録する
109     if (removeBalls.Contains(ball) == false)
110     {
111         //登録
112         removeBalls.Add(ball);
113     }
114 }
115
```

```
116    public void Spawn()
117    {
118        //コルーチンでボールを生成する
119        StartCoroutine(CreateBall(BallInstanceCount));
120    }
121
122    public IEnumerator CreateBall(int count)
123    {
124        //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
125        //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
126        //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
127        for (int i = 0; i < count; i++)
128        {
129            Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
130            GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
131
132            //ランダムでボールを設定
133            int ballID = Random.Range(0, BallSprites.Length);
134            ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
135            ball.GetComponent<Ball>().BallID = ballID;
136            yield return new WaitForSeconds(0.04f);
137        }
138    }
139 }
140
```

GameManager_7.cs

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

public class GameManager_7 : MonoBehaviour
{
    [SerializeField] GameObject BallPrefab;
    [SerializeField] int BallInstanceCount;
    [SerializeField] Sprite[] BallSprites;
     [SerializeField] List<Ball> removeBalls = new List<Ball>();

    private bool isDragging;
    private Ball currentDraggingBall;

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

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            OnDragBigin();
        }
        else if (Input.GetMouseButtonUp(0))
        {
            OnDragEnd();
        }
        else if (isDragging)
        {
            OnDragging();
        }
    }

```

```
38    void OnDragBigin()
39    {
40        Debug.Log("ドラッグ開始");
41
42        //座標変換
43        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
44        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
45
46        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
47        if (hit && hit.collider.GetComponent<Ball>())
48        {
49            Debug.Log("---Ballにhit---");
50
51            //Listに登録する
52            Ball ball = hit.collider.GetComponent<Ball>();
53            AddRemoveBall(ball);
54            isDragging = true;
55
56            //最後に触ったボールを記憶
57            currentDraggingBall = ball;
58        }
59    }
60    void OnDragging()
61    {
62        //座標変換
63        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
64        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
65
66        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
67        if (hit && hit.collider.GetComponent<Ball>())
68        {
69            Ball ball = hit.collider.GetComponent<Ball>();
70
71            //最後に触ったボールと今触ったボールが同じIDであるか判別
72            if (ball.BallID == currentDraggingBall.BallID)
73            {
74                //2点の距離が遠くないか判別
75                float distance = Vector2.Distance(ball.transform.position,currentDraggingBall.transform.position);
76                if (distance < 1.5)
```

```
77              {
78                  AddRemoveBall(ball);
79
80                  //最後に触ったボールを記憶
81                  currentDraggingBall = ball;
82              }
83          }
84      }
85  }
86  void OnDragEnd()
87  {
88      //リストの要素数分、ループ
89      int removeCount = removeBalls.Count;
90
91      //3個以上じゃないと消さない
92      if (removeCount >= 3)
93      {
94          for (int i = 0; i < removeCount; i++)
95          {
96              Destroy(removeBalls[i].gameObject);
97          }
98          StartCoroutine(CreateBall(removeCount));
99      }
100
101     //要素自体を削除
102     removeBalls.Clear();
103     Debug.Log("ドラッグ終了");
104     isDragging = false;
105 }
106
107 void AddRemoveBall(Ball ball)
108 {
109     //既に登録されているGameObjectを除き、新たに登録する
110     if (removeBalls.Contains(ball) == false)
111     {
112         //登録
113         removeBalls.Add(ball);
114     }
115 }
```

```
116
117     public void Spawn()
118     {
119         //コルーチンでボールを生成する
120         StartCoroutine(CreateBall(BallInstanceCount));
121     }
122
123     public IEnumerator CreateBall(int count)
124     {
125         //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
126         //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
127         //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
128         for (int i = 0; i < count; i++)
129         {
130             Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
131             GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
132
133             //ランダムでボールを設定
134             int ballID = Random.Range(0, BallSprites.Length);
135             ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
136             ball.GetComponent<Ball>().BallID = ballID;
137             yield return new WaitForSeconds(0.04f);
138         }
139     }
140 }
141
```

GameManager_8.cs

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

public class GameManager_8 : MonoBehaviour
{
    [SerializeField] GameObject BallPrefab;
    [SerializeField] int BallInstanceCount;
    [SerializeField] Sprite[] BallSprites;
     [SerializeField] List<Ball> removeBalls = new List<Ball>();

    private bool isDragging;
    private Ball currentDraggingBall;

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

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            OnDragBigin();
        }
        else if (Input.GetMouseButtonUp(0))
        {
            OnDragEnd();
        }
        else if (isDragging)
        {
            OnDragging();
        }
    }

```

```
38    void OnDragBigin()
39    {
40        Debug.Log("ドラッグ開始");
41
42        //座標変換
43        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
44        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
45
46        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
47        if (hit && hit.collider.GetComponent<Ball>())
48        {
49            Debug.Log("---Ballにhit---");
50
51            //Listに登録する
52            Ball ball = hit.collider.GetComponent<Ball>();
53            AddRemoveBall(ball);
54            isDragging = true;
55
56            //最後に触ったボールを記憶
57            currentDraggingBall = ball;
58        }
59    }
60    void OnDragging()
61    {
62        //座標変換
63        Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
64        RaycastHit2D hit = Physics2D.Raycast(mousePosition, Vector2.zero);
65
66        //飛ばしたレイに対して、当たったGameObjectにBallクラスが付与されているか確認
67        if (hit && hit.collider.GetComponent<Ball>())
68        {
69            Ball ball = hit.collider.GetComponent<Ball>();
70
71            //最後に触ったボールと今触ったボールが同じIDであるか判別
72            if (ball.BallID == currentDraggingBall.BallID)
73            {
74                //2点の距離が遠くないか判別
75                float distance = Vector2.Distance(ball.transform.position,currentDraggingBall.transform.position);
76                if (distance < 1.5)
```

```
77              {
78                  AddRemoveBall(ball);
79
80                  //最後に触ったボールを記憶
81                  currentDraggingBall = ball;
82              }
83          }
84      }
85  }
86  void OnDragEnd()
87  {
88      //リストの要素数分、ループ
89      int removeCount = removeBalls.Count;
90
91      //3個以上じゃないと消さない
92      if (removeCount >= 3)
93      {
94          for (int i = 0; i < removeCount; i++)
95          {
96              Destroy(removeBalls[i].gameObject);
97          }
98          StartCoroutine(CreateBall(removeCount));
99      }
100
101     //ボールの大きさを戻す
102     for (int i = 0; i < removeCount; i++)
103     {
104         removeBalls[i].transform.localScale = Vector3.one;
105     }
106
107     //要素自体を削除
108     removeBalls.Clear();
109     Debug.Log("ドラッグ終了");
110     isDragging = false;
111 }
112
113 void AddRemoveBall(Ball ball)
114 {
115     //既に登録されているGameObjectを除き、新たに登録する
```

```
116         if (removeBalls.Contains(ball) == false)
117         {
118             //登録
119             removeBalls.Add(ball);
120 
121             //ボールを大きくする
122             ball.transform.localScale = Vector3.one * 1.4f;
123         }
124     }
125 
126     public void Spawn()
127     {
128         //コルーチンでボールを生成する
129         StartCoroutine(CreateBall(BallInstanceCount));
130     }
131 
132     public IEnumerator CreateBall(int count)
133     {
134         //ボールを0.04秒ごとに生成する。作成する数は引数のcount分。
135         //生成する際、X軸は+0.2から-0.2の範囲、Y軸は8固定とする
136         //角度の指定が出てきた場合は、Quaternion.identity(回転しない)でOK
137         for (int i = 0; i < count; i++)
138         {
139             Vector2 pos = new Vector2(Random.Range(-0.2f,0.2f),8f);
140             GameObject ball = Instantiate(BallPrefab, pos, Quaternion.identity);
141 
142             //ランダムでボールを設定
143             int ballID = Random.Range(0, BallSprites.Length);
144             ball.GetComponent<SpriteRenderer>().sprite = BallSprites[ballID];
145             ball.GetComponent<Ball>().BallID = ballID;
146             yield return new WaitForSeconds(0.04f);
147         }
148     }
149 }
150 
```