C 语言项目报告

项目名称：____B2B（Ball to Basket）___

项目成员：_____申奥、朱子琦、高庆霖_____

填写日期：_____2021.07.09_____

1 摘要（Abstract）

B2B（Ball to Basket）游戏项目是一款应用 Unity 的物理引擎的多种功能的休闲益智类游戏，指能在游戏中锻炼人的逻辑力与敏捷力的一系列需利用部分物理知识才能实现胜利的游戏。
Unity 作为一个强大的游戏开发引擎，具有较为完善的有关物体碰撞等一系列游戏所需要的函数，以及较为便捷的 UI 设计方法。故我们选择使用 unity 开发一个有关小球运动与碰撞的物理休闲益智类游戏，并在其中加入诸如混色、叠加等设计增加其可玩性，再配以设计的烧脑关卡达到寓教于乐、在游戏中锻炼逻辑的目的，取名为 B2B（Ball to Basket）。

2 问题描述（Problem Statement）

2.1 项目背景

近年来，随着经济科技的发展、生活节奏的加快，人们对休闲类游戏的需求越来越大。物理休闲益智类游戏使人们在休闲放松的同时能够锻炼自己的逻辑力与敏捷力，是一项很好的放松选择。现在市面上有许多物理休闲益智类小游戏，如割绳子、弹珠等，相关游戏设计思路与设计方法均有较强的借鉴意义。本游戏就是在这种背景下被开发出来的，我们致力于实现游戏放松身心目的的同时，能对玩家的大脑及能力实现一定的锻炼与提高。对于游戏设计环境的选择，我们参考了班内大多数同学的使用 Unity 的现状，基于多交流多学习共同进步的目的和 Unity 作为一个强大的游戏开发引擎，具有较为完善的有关物体碰撞等一系列游戏所需要的函数，以及较为便捷的 UI 设计方法的重要优势，我们选择了 Unity 作为游戏制作引擎。

2.2 实际问题

但在实际工作中我们遇到了游戏物体与交互道具绘制来源、**vs 函数的关联、多颜色小球的实现与交互、物理引擎的使用、胜利失败条件的判定、游戏体验的丰富、益智烧脑关卡设计、开局界面的设计等诸多细节问题。**通过灵活使用图片处理类软件和编程软件，以及利用的游戏引擎中的部分函数，并查找资料，参考学习源代码，灵活使用算法，我们成功解决了上述问题，并且通过开关、加速器、翻斗、篮子堆叠、染色器、混色器等多种游戏道具实现丰富的游戏关卡，使项目可玩性和完整度大大提高。

3 组内分工（Group Division）

名字	学号	学院	工作百分比
申奥	09201231	生物	35%
朱子琦	13200303	生物	30%
高庆霖	22299221	生物	35%

3.1 申奥

申奥是小组的组长，进行了项目的选题，安排了项目的分工，建立了项目的框架。申奥主要负责这个程序中的物理碰撞的实现，开关、加速器和翻斗三个道具的功能实现和预制，以及第 3、5、6、7、8、9 关卡的制作。

3.2 朱子琦

朱子琦主要负责 UI 布局与通用预制的生成，篮子堆叠、染色器和混色器三个道具的

功能实现和预制，以及第 2、4、10、11、12 关卡的制作。

3.3 高庆霖

高庆霖辅助进行了项目的选题和游戏制作引擎的选择。高庆霖主要负责了游戏胜利失败的判定，界面功能按钮的实现和开局界面与选关界面的实现。最后，他整合了其他小组成员编写的项目并生成了游戏应用程序。

3.4 文档工作

朱子琦负责了开题、中期和期末答辩 PPT 的制作和开题项目开发设计书；
高庆霖负责了中期的接口设计文档和软件需求文档；
申奥负责了中期的详细设计文档和期末的项目报告。

4 分析（Analysis）

4.1 游戏逻辑与游戏规则

首先，我们需要确定游戏逻辑与游戏规则。游戏的目标十分简单，即玩家通过在合适的入口按照合适的顺序放置带有颜色的小球，使小球在物理引擎的模拟下能够掉落入对应颜色的篮子里，即“Ball to Basket”。

概括游戏规则如下:

(1)拖动颜色小球按钮到入口区生成物理小球

(2)玩家需要使带颜色的小球进入对应篮子中赢得胜利

(3)篮子中有多于一个小球或落入小球颜色错误则游戏失败

(4)在可能短的时间内完成关卡从而得到更高的星级和金钱奖励

4.2 界面建立与游戏按钮与路径绘制

建立游戏界面、开局界面、选关界面、胜利界面、两个失败界面，绘制各种游戏按钮与路径。

4.3 通用功能

对每一关均需要成立的功能需求如下：暂停功能、重玩功能、小球生成功能、小球拖拽功能、小球销毁功能、生成物理小球功能、小球轨道设置功能、篮子承接功能、胜利失败判定功能、计时器与积分功能。

4.4 特有道具功能

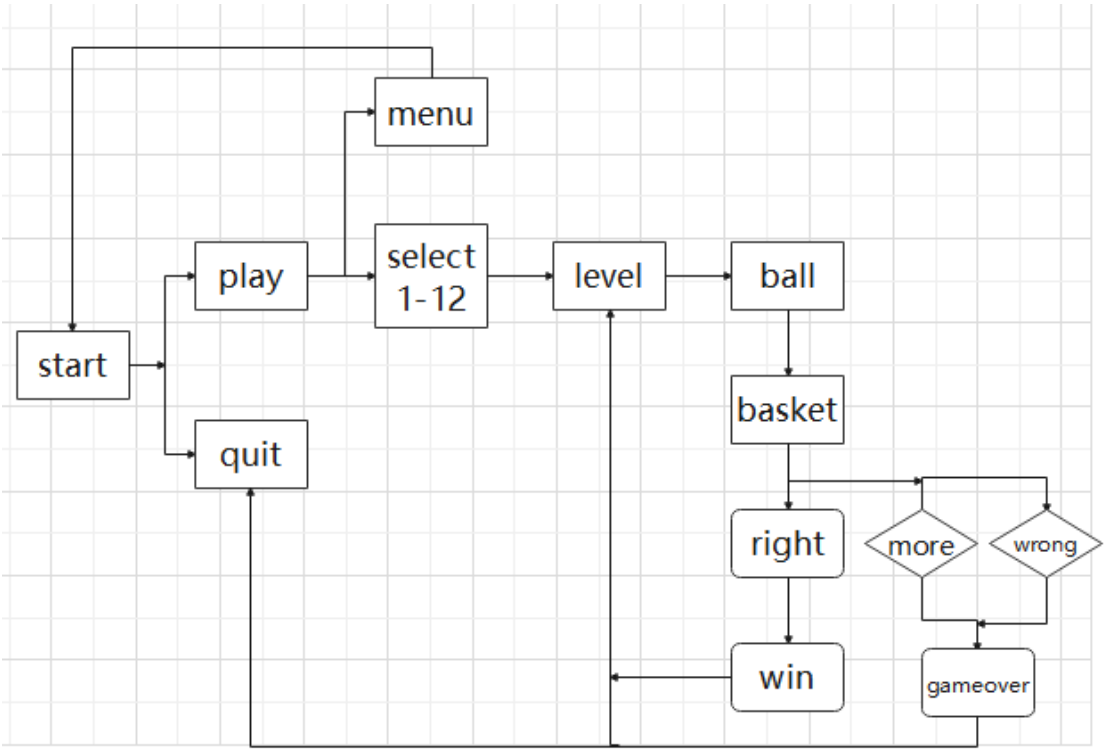
对特定关卡新引入的道具的功能需求如下：开关功能、篮子堆叠功能、翻斗功能、染色器与混色器功能、加速器功能。

4.5 游戏整体功能

对关卡游戏的连续性与完整性的功能需求如下：退出游戏功能、切换界面功能。

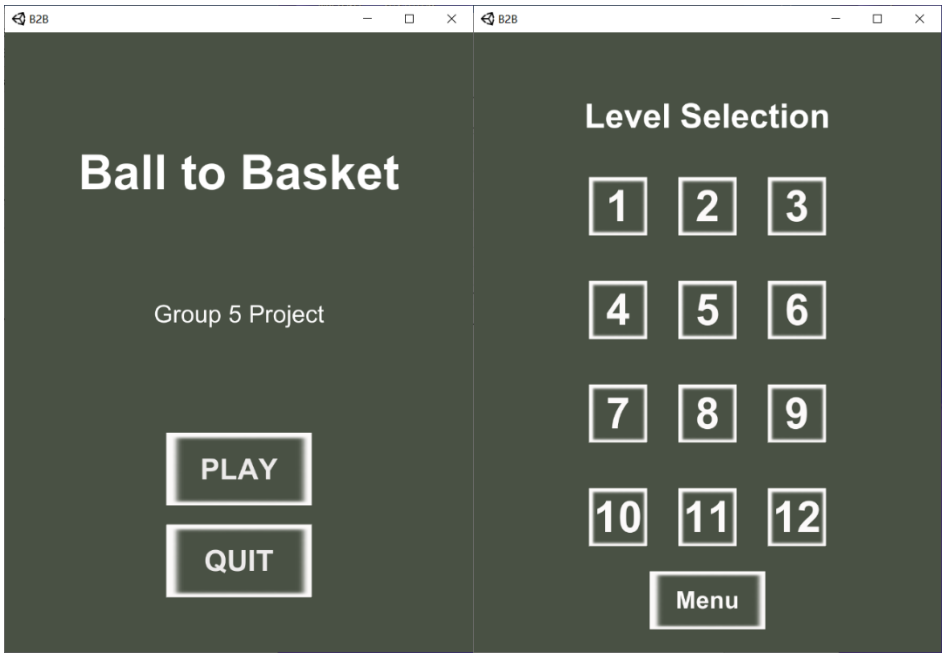
5 设计（Design）

5.1 游戏逻辑与游戏规则



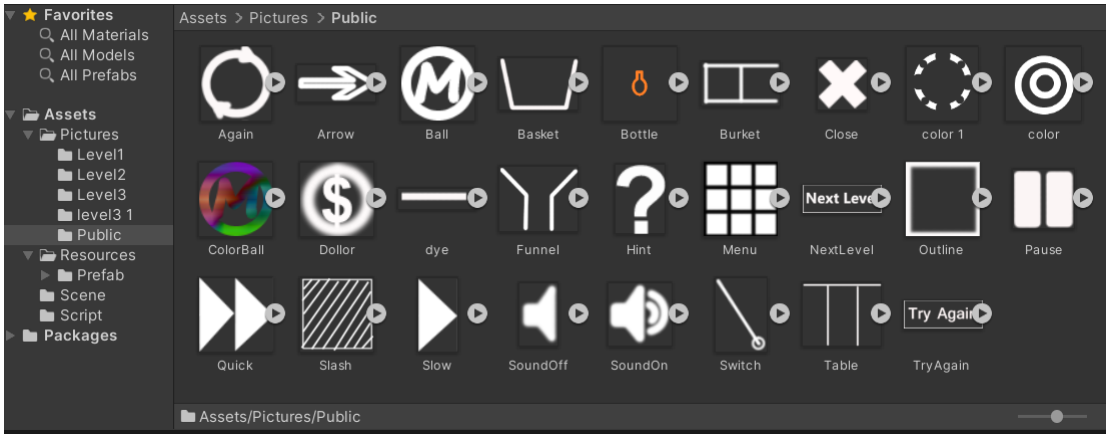
图一

界面建立与游戏按钮与路径绘制

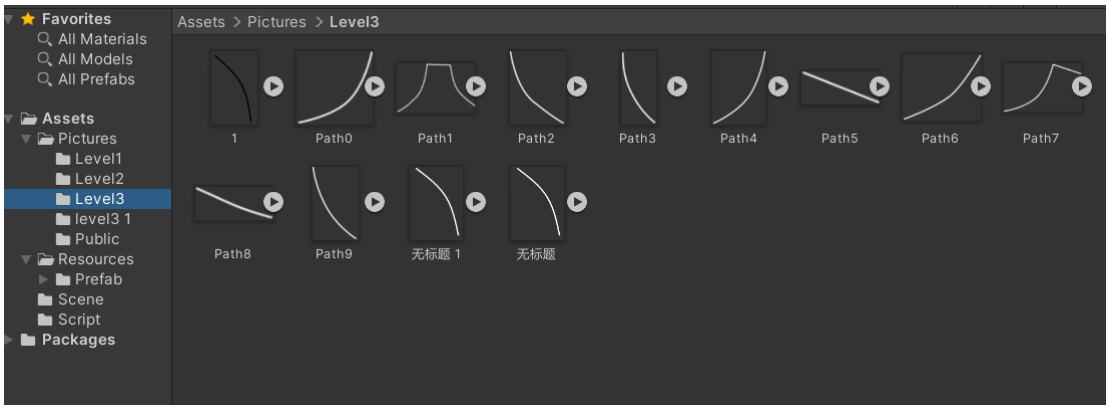


图二 开局界面

图三 选关界面



图四 游戏道具



图五 游戏路径

5.2 通用功能

暂停功能：由暂停按钮实现游戏中计时器的暂停与开始

重玩功能：由重玩按钮实现游戏中当前关卡的重玩

小球生成功能：由小球按钮实现生成对应颜色的小球

小球拖拽功能：由鼠标位置检测函数实现生成小球随鼠标移动的拖拽效果

小球销毁功能：由鼠标事件检测函数实现鼠标左键松开时生成小球的销毁

生成物理小球功能：由 3D/2D 射线碰撞判定实现在入口区生成物理小球

小球轨道设置功能：由刚体组件属性和物理碰撞检测实现小球沿设计轨道运动的物理碰撞过程

篮子承接功能：边碰撞器或多边形碰撞器检测实现小球在篮子底部停止

胜利失败判定功能：由胜利失败情景枚举函数与物理属性检测器实现游戏关卡胜利或失败的判定

计时器与评分功能：由总体控制脚本中的二维数组判定每关通关时间规定星级标准，由胜利界面脚本中的内置函数计时

5.3 特有道具功能

开关功能：由位置差异的组合体与碰撞函数实现开关切换功能

篮子堆叠功能：由篮子组碰撞函数与篮子碰撞体实现篮子对不同小球经过时的穿过判定

翻斗功能：由刚体旋转函数与翻斗碰撞体实现翻斗对小球的承接与倒出

染色器与混色器功能：由颜色索引函数与转换混合函数实现对小球颜色的改变

加速器功能：由位置差异的组合体与加速函数实现对小球速度与方向的改变

5.4 游戏整体功能

退出游戏功能：由退出游戏按钮实现退出游戏
切换界面功能：由场景索引函数与场景按钮实现切换场景

6 实施（Implementation）

6.1 Basket 脚本：实现篮子颜色设置功能、篮子堆叠功能、篮子承接小球功能、篮子状态函数赋值

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Basket : MonoBehaviour
6   {
7       //篮子颜色索引
8       public int colorIndex;
9
10      //篮子状态，0为空，1为颜色对，-1为颜色错，-2为超过一个
11      public int basketState;
12
13      //篮子的堆叠的层索引
14      public int pileIndex;
15
16      //穿过该篮子的小球的数目
17      private int exitBallNum;
18
19      //自身的多边形碰撞体组件
20      private PolygonCollider2D selfPolyCollider2D;
21
22      // Start is called before the first frame update
23      void Start()
24      {
25          //初始时，篮子为空
26          basketState = 0;
27
28          //初始时，离开的小球数为0
29          exitBallNum = 0;
30
31          //自身的多边形碰撞体组件
32          selfPolyCollider2D = this.GetComponent<PolygonCollider2D>();
33
34          //根据离开小球数和堆叠层索引判断多边形碰撞体组件的激活与禁用状态
35          selfPolyCollider2D.enabled = (exitBallNum >= pileIndex);
36      }
37
38      // Update is called once per frame
39      void Update()
40      {
```

```
41
42      }
43
44      //小球进入
45      private void OnTriggerEnter2D(Collider2D coll)
46      {
47          //如果小球进入
48          if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall") &&
49              selfPolyCollider2D.enabled == true)
50          {   //如果篮子空
51              if (basketState == 0)
52              {
53                  //获取目标小球physicsball组件
54                  PhysicsBall tempPhysicsBall = coll.transform.GetComponent<PhysicsBall>();
55                  //如果颜色对
56                  if (this.colorIndex == tempPhysicsBall.colorIndex)
57                  {
58                      basketState = 1;
59
60                  }
61                  else
62                  {
63                  basketState = -1;
64                  }
65              }
66              else
67              {
68                  basketState = -2;
69              }
70
71              //更新篮子状态
72              LevelController.Instance.GameStateRefresh();
73          }
74      }
75      //当其他的碰撞体离开该物体时，执行的操作
76      void OnTriggerExit2D(Collider2D coll)
77      {
78          //如果进入的碰撞体是物理小球
79          if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
80          {

81              //离开小球数+1
82              exitBallNum++;
83
84              //根据离开小球数和堆叠层索引判断多边形碰撞体组件的激活与禁用状态
85              selfPolyCollider2D.enabled = (exitBallNum >= pileIndex);
86          }
87      }
88  }
89
```

6.2 Bottle 脚本：实现瓶子状态切换功能

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

public class Bottle : MonoBehaviour
{
    //瓶口朝向的角度数组
    public float[] angleArray;

    //当前瓶口朝向的角度在角度数组中的索引
    private int currentAngleIndex;

    //2D射线的碰撞信息
    private RaycastHit2D hitInfo2D;

    //射线碰撞的层标记
    private int hitMask;

    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
    {
    }

    // Use this for initialization
    void Start()
    {
        //初始时，角度索引为0
        currentAngleIndex = 0;

        //射线碰撞的层标记
        hitMask += 1 << LayerMask.NameToLayer("Bottle");
    }

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

```
    {
        //如果游戏处于Playing状态，并且鼠标按下
        if (Myclass.currentGameState == GameState.Playing && Input.GetMouseButtonDown(0))
        {
            //获得鼠标位置
            Vector2 mousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);

            //如果鼠标位置在瓶子的碰撞体的范围之内
            if (this.GetComponent<BoxCollider2D>().OverlapPoint(mousePosition))
            {
                //发射2D射线
                hitInfo2D = Physics2D.Raycast(mousePosition, Vector3.forward, Camera.main.farClipPlane, hitMask);

                //如果射线碰撞到了瓶子
                if (hitInfo2D)
                {
                    //角度索引更新
                    currentAngleIndex = (currentAngleIndex + 1) % angleArray.Length;

                    //改变瓶子朝向
                    this.transform.eulerAngles = Vector3.forward * angleArray[currentAngleIndex];
                }
            }
        }
    }

    //当该脚本组件不可用时
    void OnDisable()
    {
    }
}

```

6.3 Bottle Trigger 脚本：实现瓶子对小球运动方向和速度的改变功能

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

public class BottleTrigger : MonoBehaviour
{
    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
    {
    }

    // Use this for initialization
    void Start()
    {
    }

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

    //当该脚本组件不可用时
    void OnDisable()
    {
    }

    //当其他的碰撞体进入该物体时，执行的操作
    void OnTriggerEnter2D(Collider2D coll)
    {
        //如果进入的碰撞体是物理小球
        if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
        {
            //位置修正
            coll.transform.position = this.transform.position;

            //速度修正
            coll.attachedRigidbody.velocity = 10 * this.transform.up;
        }
    }
}
```

6.4 Bucket 脚本：实现翻斗状态的赋予功能、进入翻斗小球的计数功能、翻斗旋转方向与速度的设定功能、倒出小球重力系数修正功能

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

public class Bucket : MonoBehaviour
{
    //小球倒出之前的角速度
    public float trippingAngleSpeed;

    //小球倒出之后的角速度
    public float returnAngleSpeed;

    //翻斗的容量
    public int bucketCapacity;

    //bool值，指示翻斗是否进入旋转状态
    private bool rotateEnable;

    //翻斗的旋转类型，0表示小球倒出之前的旋转，1表示小球倒出之后的旋转
    private int rotateState;

    //翻斗所转过的总角度的计数
    private float angleCount;

    //进入小球的计数
    private int enterBallNum;

    //离开小球的计数
    private int exitBallNum;

    //翻斗的初始旋转
    private Vector3 originalEulerAngle;

    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
```

```
41      {
42      }
43
44      // Use this for initialization
45      void Start()
46      {
47          //初始时，翻斗不旋转
48          rotateEnable = false;
49
50          //初始时，角度计数为0
51          angleCount = 0;
52
53          //初始时，进入和离开小球数均为0
54          enterBallNum = 0;
55          exitBallNum = 0;
56
57          //记录初始欧拉角
58          originalEulerAngle = this.transform.eulerAngles;
59      }
60
61      // Update is called once per frame
62      void Update()
63      {
64          //如果翻斗进入旋转状态
65          if (rotateEnable)
66          {
67              //如果小球尚未倒出
68              if (rotateState == 0)
69              {
70                  //旋转
71                  this.transform.Rotate(Vector3.forward, trippingAngleSpeed * Time.deltaTime);
72
73                  //角度计数递增
74                  angleCount += Mathf.Abs(trippingAngleSpeed * Time.deltaTime);
75              }
76
77              //如果小球已经倒出
78              if (rotateState == 1)
79              {
80                  //旋转
```

```
                this.transform.Rotate(Vector3.forward, returnAngleSpeed * Time.deltaTime);

                //角度计数递减
                angleCount -= Mathf.Abs(returnAngleSpeed * Time.deltaTime);

                //如果角度达到0
                if (angleCount <= 0)
                {
                    //退出旋转状态
                    rotateEnable = false;

                    //修正旋转为初始旋转
                    this.transform.eulerAngles = originalEulerAngle;

                    //进入、离开小球数清0
                    enterBallNum = 0;
                    exitBallNum = 0;

                    //角度计数归0
                    angleCount = 0;
                }
            }
        }
    }

    //当该脚本组件不可用时
    void OnDisable()
    {
    }

    //当别的碰撞体进入该触发器时
    void OnTriggerEnter2D(Collider2D col1)
    {
        //如果进入的物体是物理小球
        if (col1.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
        {
            //进入小球计数
            enterBallNum++;

```

```
            //如果进入小球数达到翻斗容量
            if (enterBallNum == bucketCapacity)
            {
                //设置该小球的重力系数
                coll.attachedRigidbody.gravityScale = 1.7F;

                //翻斗进入旋转状态
                rotateEnable = true;

                //小球倒出之前的旋转
                rotateState = 0;
            }
        }
    }

    //当别的碰撞体离开该触发器时
    void OnTriggerExit2D(Collider2D coll)
    {
        //如果离开的物体是物理小球
        if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
        {
            //离开小球计数
            exitBallNum++;

            //如果离开小球数达到翻斗容量
            if (exitBallNum == bucketCapacity)
            {
                //翻斗进入旋转状态
                rotateEnable = true;

                //小球倒出之后的旋转
                rotateState = 1;
            }
        }
    }
}
```

6.5 Color Mixer 脚本：实现获取进入混色器小球颜色索引功能、进入小球数量计数功能、混色后新小球颜色赋予功能、小球暂停与下落功能

```
using UnityEngine;
using System.Collections;

public class ColorMixer : MonoBehaviour
{
    //进入该混合器的两个小球
    private GameObject firstBall;
    private GameObject secondBall;

    //两个小球的颜色索引
    private int firstColorIndex;
    private int secondColorIndex;

    //混合之后的颜色索引
    private int mixedColorIndex;

    //进入的小球计数器
    private int enterBallNum;

    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
    {
    }

    // Use this for initialization
    void Start()
    {
        //初始时，进入小球数为0
        enterBallNum = 0;
    }

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

```
41
42          //当该脚本组件不可用时
43          void OnDisable()
44          {
45          }
46
47          //当别的碰撞体进入混合器的触发器时
48          void OnTriggerEnter2D(Collider2D coll)
49          {
50              //如果进入的物体是物理小球
51              if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
52              {
53                  //小球计数
54                  enterBallNum++;
55
56                  //如果进入的是第一个小球
57                  if (enterBallNum == 1)
58                  {
59                      //速度为0
60                      coll.attachedRigidbody.velocity = Vector2.zero;
61
62                      //重力系数为0
63                      coll.attachedRigidbody.gravityScale = 0;
64
65                      //位置修正
66                      coll.transform.position = this.transform.position;
67
68                      //记录该小球
69                      firstBall = coll.gameObject;
70
71                      //获取该小球的颜色索引
72                      firstColorIndex = firstBall.GetComponent<PhysicsBall>().colorIndex;
73                  }
74
75                  //如果进入的是第二个小球
76                  if (enterBallNum == 2)
77                  {
78                      //速度为0
79                      coll.attachedRigidbody.velocity = Vector2.zero;
80
81                      //重力系数为0
82                      coll.attachedRigidbody.gravityScale = 0;
83
84                      //位置修正
85                      coll.transform.position = this.transform.position;
86
87                      //记录该小球
88                      secondBall = coll.gameObject;
89
90                      //获取该小球的颜色索引
91                      secondColorIndex = secondBall.GetComponent<PhysicsBall>().colorIndex;
92
93                      //计算混合之后的小球的颜色索引
94                      mixedColorIndex = Myclass.ColorMixingTable[firstColorIndex, secondColorIndex];
95
96                      //销毁第二个小球
97                      Destroy(secondBall);
98
99                      //修改第一个小球的颜色索引
100                     firstBall.GetComponent<PhysicsBall>().colorIndex = mixedColorIndex;
101
102                     //修改第一个小球的图片颜色
103                     firstBall.GetComponent<SpriteRenderer>().color = Resources.Load<SpriteRenderer>("Prefab/PhysicsBall/" + mixedColorIndex).color;
104
105                     //第一个小球重力系数恢复
106                     firstBall.GetComponent<Rigidbody2D>().gravityScale = 1;
107
108                     //进入小球计数清0
109                     enterBallNum = 0;
110                 }
111             }
112         }
113     }
114
```

6.6 Defeat Interface 脚本：实现颜色错误信息显示功能、书面错误信息

显示功能、失败后重玩按钮重玩功能

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

public class DefeatInterface : MonoBehaviour
{
    public Text defeatInfo;

    private void OnEnable()
    {
        //如果颜色错误
        if (Myclass.currentGameState == GameState.ColorError)
        {
            //显示失败信息
            defeatInfo.text = "Mismatched \n Colors";
        }
        //如果数目错误
        if (Myclass.currentGameState == GameState.NumError)
        {
            //显示失败信息
            defeatInfo.text = "Too Many \n Marbles";
        }

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

    }

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

    }

    //点重玩后
    public void TryAgainButtonClick()
    {
        //重新加载关卡
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }
}

```

6.7 Drag Ball 脚本：实现鼠标点击拖动小球图标功能

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

public class DragBall : MonoBehaviour
{
    //鼠标位置
    private Vector3 currentMousePosition;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        //计算鼠标当前坐标
        currentMousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        currentMousePosition.z = 0;

        //小球跟随鼠标
        this.transform.position = currentMousePosition;
    }
}

```

6.8 Dye 脚本：实现改变物理小球颜色功能

```
using UnityEngine;
using System.Collections;

public class Dye : MonoBehaviour
{
    //染色器的颜色索引
    public int colorIndex;

    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
    {
    }

    // Use this for initialization
    void Start()
    {
    }

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

    //当该脚本组件不可用时
    void OnDisable()
    {
    }

    //当别的碰撞体进入染色器的触发器时
    void OnTriggerEnter2D(Collider2D coll)
    {
        //如果进入的物体是物理小球
        if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
        {
            //修改物理小球的颜色索引
```

```
            coll.gameObject.GetComponent<PhysicsBall>().colorIndex = this.colorIndex;

            //修改物理小球的图片颜色
            coll.gameObject.GetComponent<SpriteRenderer>().color = this.GetComponent<SpriteRenderer>().color;
        }
    }
}
```

6.9 Function Button 脚本：实现游戏中重玩按钮重玩功能

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

public class FunctionButton : MonoBehaviour
{
    public void PlayAgainButtonClick()
    {
        //重新加载关卡
        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
    }

}
```

6.10 Level Choose 脚本：实现选关界面选关按钮选关功能

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


public class LevelChoose : MonoBehaviour
{



    public void LoadScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }


}
```

6.11 Level Controller 脚本：实现胜利判定与界面显示、数目错误判定与界面显示、颜色错误判定与界面显示功能

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

public class LevelController : MonoBehaviour
{
    //单例
    public static LevelController Instance;

    public int levelnum;

    //失败界面
    public GameObject defeatInterface;

    //胜利界面
    public GameObject winInterface;

    //本局游戏时间
    public float playTime;

    //储存篮子信息
    private Basket[] basketArray;
    private void Awake()
    {   //单例赋值
        Instance = this;
    }
    // Start is called before the first frame update
    void Start()
    {
        //游戏状态为playing
        Myclass.currentGameState = GameState.Playing;

        //计算归零
        playTime = 0;

        //获取篮子数
        basketArray = GameObject.FindObjectsOfType<Basket>();
    }
```

```
41
42      // Update is called once per frame
43      void Update()
44      {
45          //如果游戏处于play状态
46          if (Myclass.currentGameState == GameState.Playing)
47          {
48              //计算递增
49              playTime += Time.deltaTime;
50          }
51
52      }
53
54      //游戏状态更新
55      public void GameStateRefresh()
56      {
57          //如果为play状态
58          if (Myclass.currentGameState == GameState.Playing)
59          {
60              //如果颜色错
61              if (ColorErrorJudge())
62              {
63                  Myclass.currentGameState = GameState.ColorError;
64                  //失败界面激活
65                  defeatInterface.SetActive(true);
66              }
67              //如果数目错
68              else if (NumErrorJudge())
69              {
70                  Myclass.currentGameState = GameState.NumError;
71                  //失败界面激活
72                  defeatInterface.SetActive(true);
73              }
74              else if(WinJudge())
75              {
76                  Myclass.currentGameState = GameState.Win;
77                  //胜利界面激活
78                  winInterface.SetActive(true);
79              }
80          }
```

```
    }

    //颜色错误判断
    private bool ColorErrorJudge()
    {
        //遍历篮子
        for (int i = 0; i < basketArray.Length; i++)
        {
            if(basketArray[i].basketState ==-1)
            {
                //返回
                return true;
            }
        }
        return false;
    }
    //数目错误判断
    private bool NumErrorJudge()
    {
        //遍历篮子
        for (int i = 0; i < basketArray.Length; i++)
        {
            if (basketArray[i].basketState == -2)
            {
                //返回
                return true;
            }
        }
        return false;
    }

    //颜色错误判断
    private bool WinJudge()
    {
        //遍历篮子
        for (int i = 0; i < basketArray.Length; i++)
        {
            if (basketArray[i].basketState != 1)
            {
                //尚未胜利
                return false;
            }
        }
        return true;
    }


}
```

6.12 Main Menu 脚本：实现开始界面的进入与退出功能

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

public class MainMenu : MonoBehaviour
{
    public void PlayButtonClick()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void QuitButtonClick()
    {

        Application.Quit();
    }

}

```

6.13 Marble Button 脚本：实现对应颜色拖动小球的生成与销毁、鼠标位置与状态的检测、对应颜色物理小球的生成功能

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

public class MarbleButton : MonoBehaviour
{
    //该小球的颜色索引
    public int colorIndex;

    //生成拖动小球
    private GameObject tempDragBall;

    //鼠标位置
    private Vector3 currentMousePosition;

    //3D射线碰撞信息
    private RaycastHit hitInfo3D;

    //射线层标记
    private int hitMask;

    // Start is called before the first frame update
    void Start()
    {
        //计算射线碰撞层标记
        hitMask += 1 << LayerMask.NameToLayer("Entrance");
    }

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

    }

     //  执行鼠标按下
     public void MarbleButtonDown()
    {
        //计算鼠标当前坐标
        currentMousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
        currentMousePosition.z = 0;


        //生成对应拖动小球
        tempDragBall = Instantiate(Resources.Load<GameObject>("Prefab/DragBall/" + colorIndex),
                                   currentMousePosition,
                                   Quaternion.identity) as GameObject;

    }

    //  执行鼠标松开
    public void MarbleButtonUp()
    {
        //销毁临时小球
        Destroy(tempDragBall);

        //生成3D射线
        Ray testRay = Camera.main.ScreenPointToRay(Input.mousePosition);

        //发射射线
       bool hitresult= Physics.Raycast(testRay, out hitInfo3D, Camera.main.farClipPlane,hitMask);

        //射线碰到
        if (hitresult)
        {
            //在入口生成小球
            Instantiate(Resources.Load<GameObject>("Prefab/PhysicsBall/" + colorIndex),
                        hitInfo3D.transform.position, Quaternion.identity);

        }
    }
}
```

6.14 My class 脚本：实现游戏状态的赋予、关卡时间目标的设定、小球混

色原理设定功能

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



 //游戏状态
public enum GameState
 {
        Load,
        Playing,
        Win,
        ColorError,
        NumError
 }
public class Myclass
 {   //当前状态
     public static GameState currentGameState = GameState.Load;

     //当前关卡索引
     public static int currentLevelIndex = 3;


     //游戏每关时间分界
     public static float[,] levelThresholdTime ={
                                                 {8,15 },
                                                 {10,15 },
                                                 {15,20 },
                                                 {10,15 },
                                                 {15,20 },
                                                 {15,20 },
                                                 {15,20 },
                                                 {15,20},
                                                 {25,35 },
                                                 {20,25 },
                                                 {25,35 },
                                                 {25,35 },



     };

     //玩家总金币
     public static int totalCoinNum = 0;

     //游戏中各个小球颜色混合表,0红，1绿，2蓝，3黄，4橙，5紫，6灰
     public static int[,] ColorMixingTable = {
                                                  {0,6,5,4,6,6,6},
                                                  {6,1,6,6,6,6,6},
                                                  {5,6,2,6,6,6,6},
                                                  {4,6,6,3,6,6,6},
                                                  {6,6,6,6,4,6,6},
                                                  {6,6,6,6,6,5,6},
                                                  {6,6,6,6,6,6,6}
                                             };
 }
```

6.15 Pause Interface 脚本：实现暂停界面的激活与关闭、控制游戏时间的流动和停止功能

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

public class PauseInterface : MonoBehaviour
{
    public static bool GameIsPaused = false;

    public GameObject pauseInterface;

    // Update is called once per frame

    public void Resume()
    {   //关闭暂停界面
        pauseInterface.SetActive(false);
        //计时恢复
        Time.timeScale = 1f;
    }

    public void Pause()
    {   //激活暂停界面
        pauseInterface.SetActive(true);
        //计时停止
        Time.timeScale = 0f;

    }

    public void LoadScene(string sceneName)
    {   //场景切换函数
        SceneManager.LoadScene(sceneName);
        Time.timeScale = 1f;
    }

    public void QuitButtonClick()
    {
        //退出程序
        Application.Quit();
    }
}

```

6.16 Physics Ball 脚本：实现物理小球状态的赋予功能

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

public class PhysicsBall : MonoBehaviour
{
    //该小球的颜色索引
    public int colorIndex;

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

    }

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

    }
}

```

6.17 Switch 脚本：实现开关状态切换功能、物理小球路径检测功能

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

public class Switch : MonoBehaviour
{
    //两个子物体的引用
    private GameObject child0;
    private GameObject child1;

    //唤醒
    void Awake()
    {
    }

    //当该脚本组件可用时
    void OnEnable()
    {
    }

    // Use this for initialization
    void Start()
    {
        //获取子物体的引用
        child0 = this.transform.GetChild(0).gameObject;
        child1 = this.transform.GetChild(1).gameObject;
    }

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

    //当该脚本组件不可用时
    void OnDisable()
    {
    }

    //当别的碰撞体离开该触发器时，执行操作
    void OnTriggerExit2D(Collider2D coll)
    {
        //离开的物体是物理小球
        if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
        {
            //子物体的激活状态置反
            child0.SetActive(!child0.activeSelf);
            child1.SetActive(!child1.activeSelf);
        }
    }
}
```

6.18 Win Interface 脚本：实现关卡计时、时间等级赋予、星星等级判定与显示、最佳时间的更新与显示、金币获取与累计、胜利界面关卡重玩功能

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

public class WinInterface : MonoBehaviour
{

    //本局真实，目标，最佳时间
    public Text youTimeText;
    public Text goalTimeText;
    public Text bestTimeText;

    //星级
    private int starLevel;

    //星星的图片组件
    public Image firstStar;
    public Image secondStar;
    public Image thirdStar;

    //实心星星和空心星星
    private Sprite emptyStar;
    private Sprite filledStar;

    //金币文本
    public Text coinNumText;

    private void OnEnable()
    {
        //显示真实时间
        DisplayTime(LevelController.Instance.playTime, youTimeText);

        //显示目标时间
        DisplayTime(Myclass.levelThresholdTime[Myclass.currentLevelIndex, 0], goalTimeText);

        //读取最佳时间
        float tempBestTime = PlayerPrefs.GetFloat("bestTime" + Myclass.currentLevelIndex, float.MaxValue);
        if(LevelController.Instance.playTime< tempBestTime)
        {
            tempBestTime = LevelController.Instance.playTime;

            //记录新的最佳时间
            PlayerPrefs.SetFloat("bestTime" + Myclass.currentLevelIndex, tempBestTime);
        }
        //显示最佳时间
        DisplayTime(tempBestTime, bestTimeText);

        //获取实心空心星星图片
        emptyStar = Resources.Load<Sprite>("Prefab/Star/EmptyStar");
        filledStar = Resources.Load<Sprite>("Prefab/Star/FilledStar");

        //计算星级
        CaclculateStarLevel();

        //显示星级
        DisplayStarLevel();

        //显示金币数
        DisplayCoin();
    }

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

    }

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

    }
    //将float格式时间转为“00：00”
    private void DisplayTime(float targetTime, Text targetText)
    {
        TimeSpan tempTimeSpan = new TimeSpan(0, 0, (int)targetTime);

```

```
            targetText.text = tempTimeSpan.Minutes.ToString("00") + ":" + tempTimeSpan.Seconds.ToString("00");

        }

        //计算本关星级
        private void CaclculateStarLevel()
        {
            //3星
            if (LevelController.Instance.playTime <= Myclass.levelThresholdTime[Myclass.currentLevelIndex, 0])
            {
                starLevel = 3;
            }
            //2星
            else if(LevelController.Instance.playTime<= Myclass.levelThresholdTime[Myclass.currentLevelIndex, 1])
            {
                starLevel = 2;
            }
            else
            {
                starLevel = 1;
            }
        }

        //显示本关星级
        private void DisplayStarLevel()
        {
            //3星
            if (starLevel == 3)
            {
                firstStar.sprite = filledStar;
                secondStar.sprite = filledStar;
                thirdStar.sprite = filledStar;
            }
            //2星
            else if (starLevel == 2)
            {
                firstStar.sprite = filledStar;
                secondStar.sprite = filledStar;
                thirdStar.sprite = emptyStar;
            }
            else if (starLevel == 1)
            {
                firstStar.sprite = filledStar;
                secondStar.sprite = emptyStar;
                thirdStar.sprite = emptyStar;
            }
        }

        //计算本关金币
        private void DisplayCoin()
        {
            //本局获得
            int tempCoin = 50 * starLevel;
            //显示
            coinNumText.text = "+" + tempCoin;

            //读取总金币
            Myclass.totalCoinNum = PlayerPrefs.GetInt("totalCoinNum", 0);
            //更新金币
            Myclass.totalCoinNum += tempCoin;
            //重新存入
            PlayerPrefs.SetInt("totalCoinNum", Myclass.totalCoinNum);
        }

        public void PlayAgainButtonClick()
        {
            //重新加载关卡
            SceneManager.LoadScene(SceneManager.GetActiveScene().name);
        }
        public void LoadScene(string sceneName)
        {
            SceneManager.LoadScene(sceneName);
        }

        public void NextLevel()
        {
            SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
        }

    }

```

7 测试（Test）

7.1 功能测试

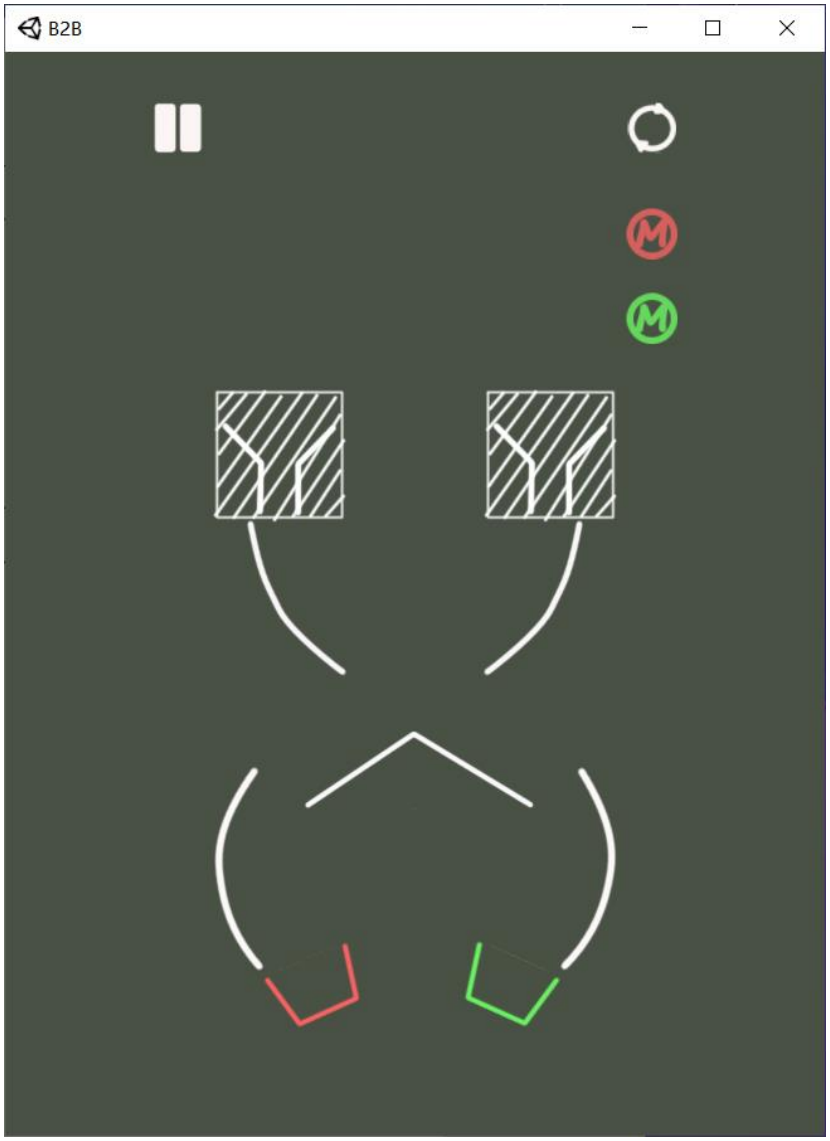
7.1.1 颜色失败判定

7.1.2 数量失败判定

7.1.3 胜利判定

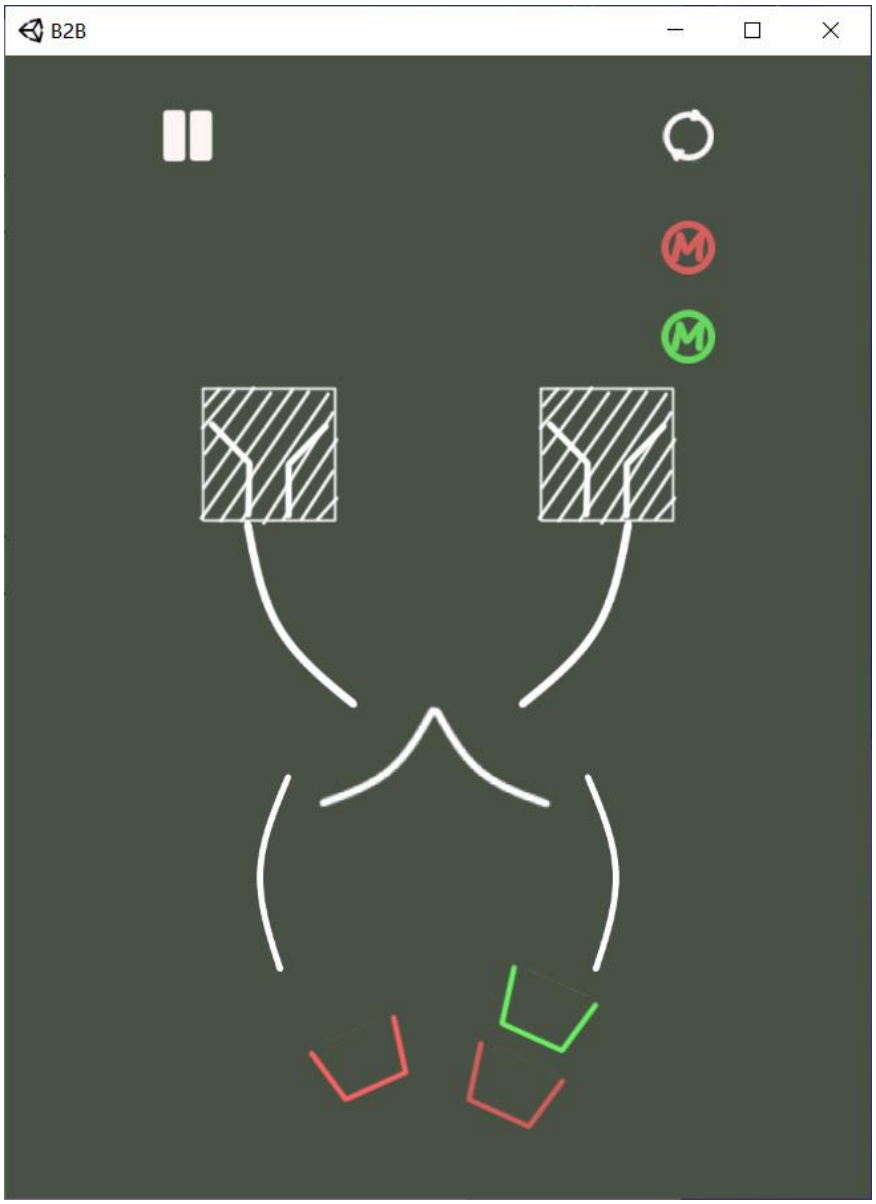
7.2 关卡展示与攻略

7.2.1 Level 1



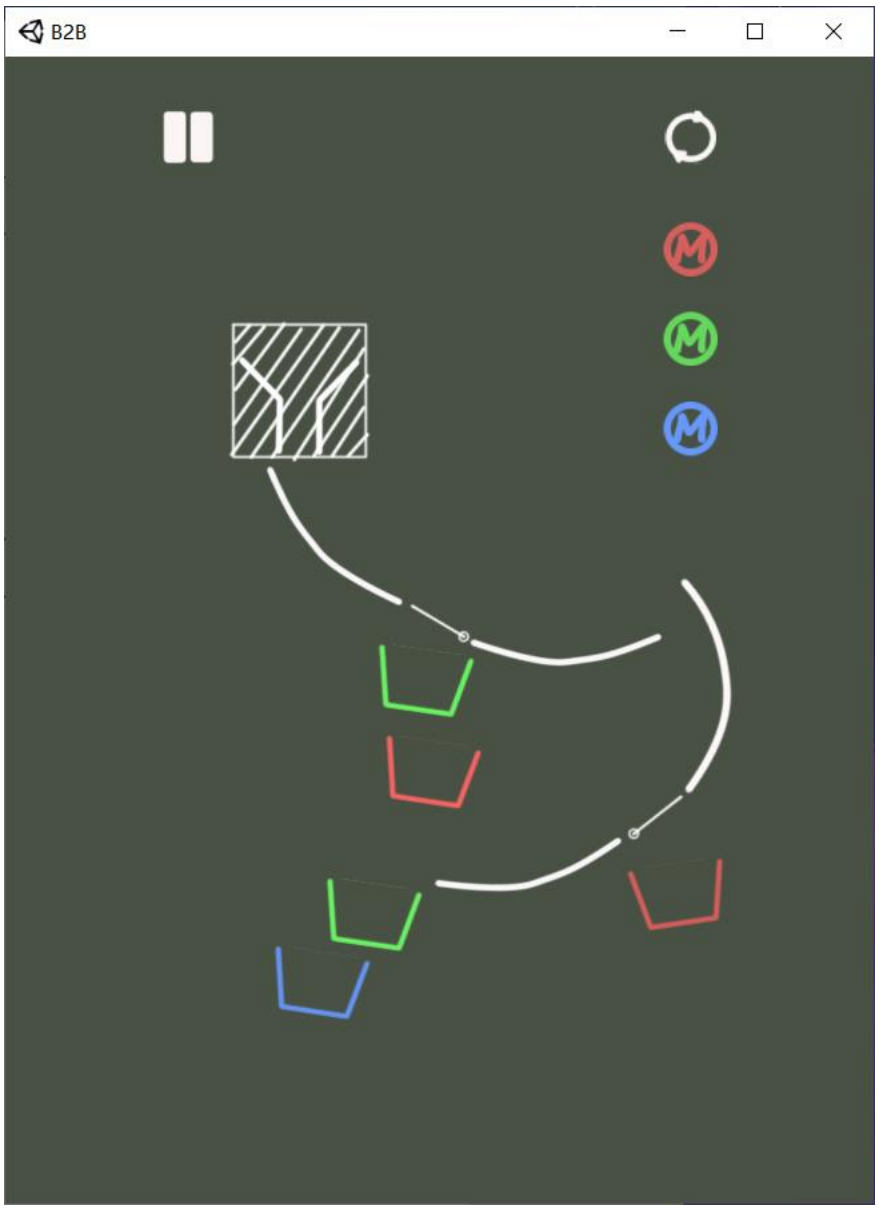
关卡攻略：左入口：绿→右入口：红。

7.2.2 Level 2



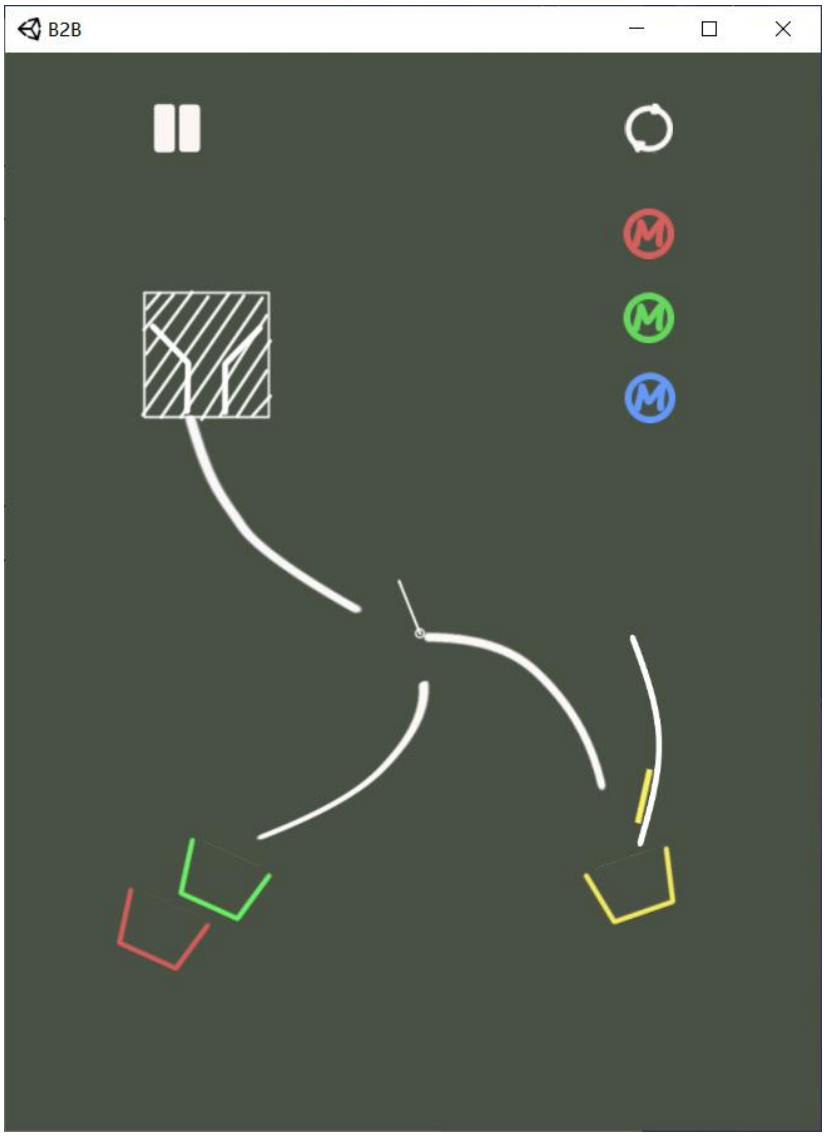
关卡攻略：左入口：红→右入口：红→右入口：绿。

7.2.3 Level 3



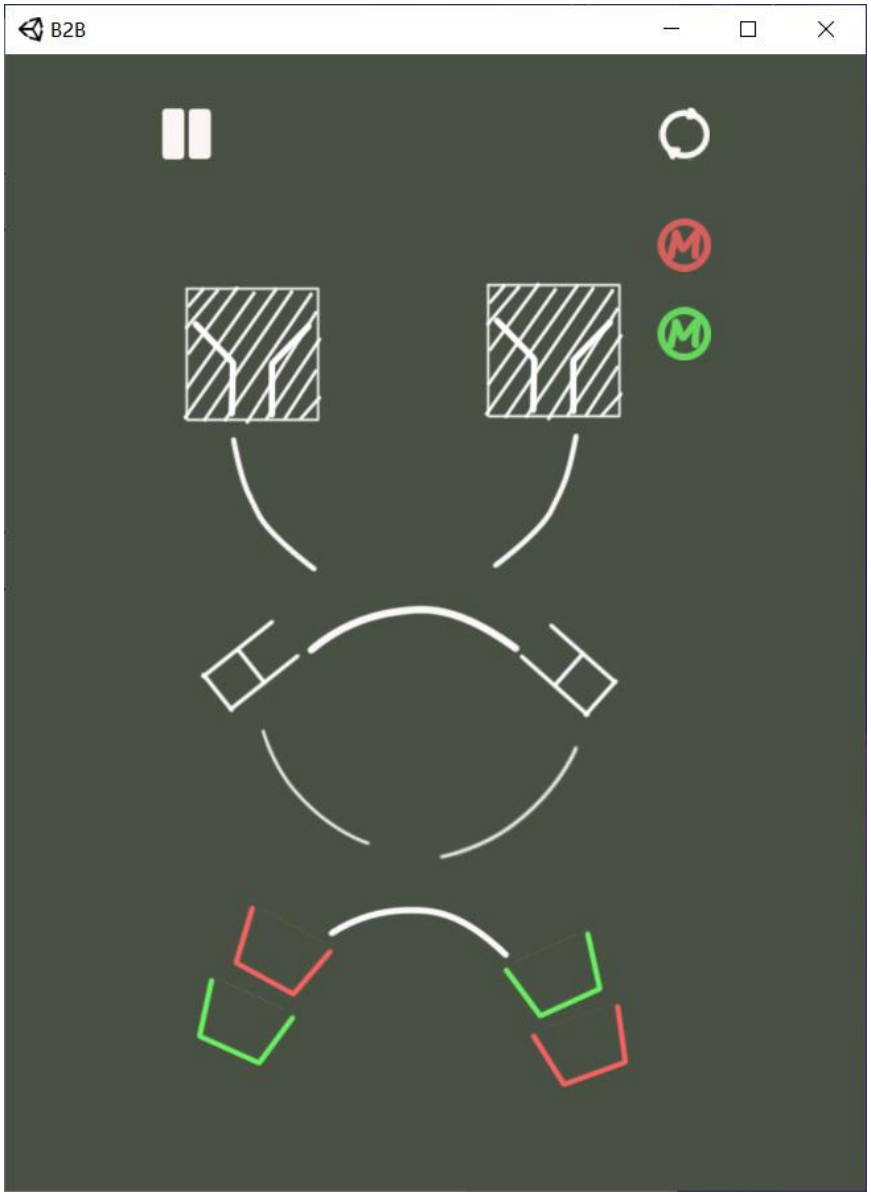
游戏攻略：入口：蓝→入口：红→入口：红→入口：绿
→入口：绿

7.2.4 Level 4



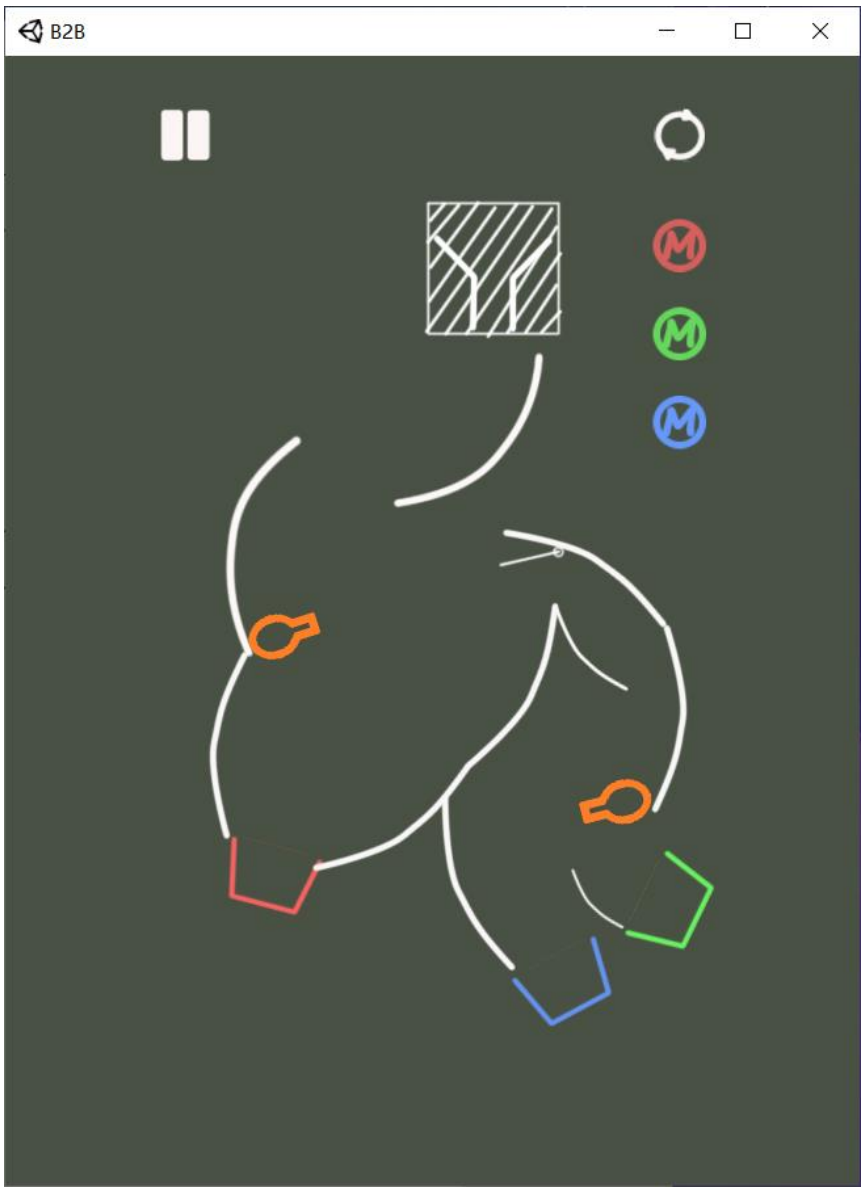
游戏攻略：入口：红→入口：黄→入口：绿

7.2.5 Level 5



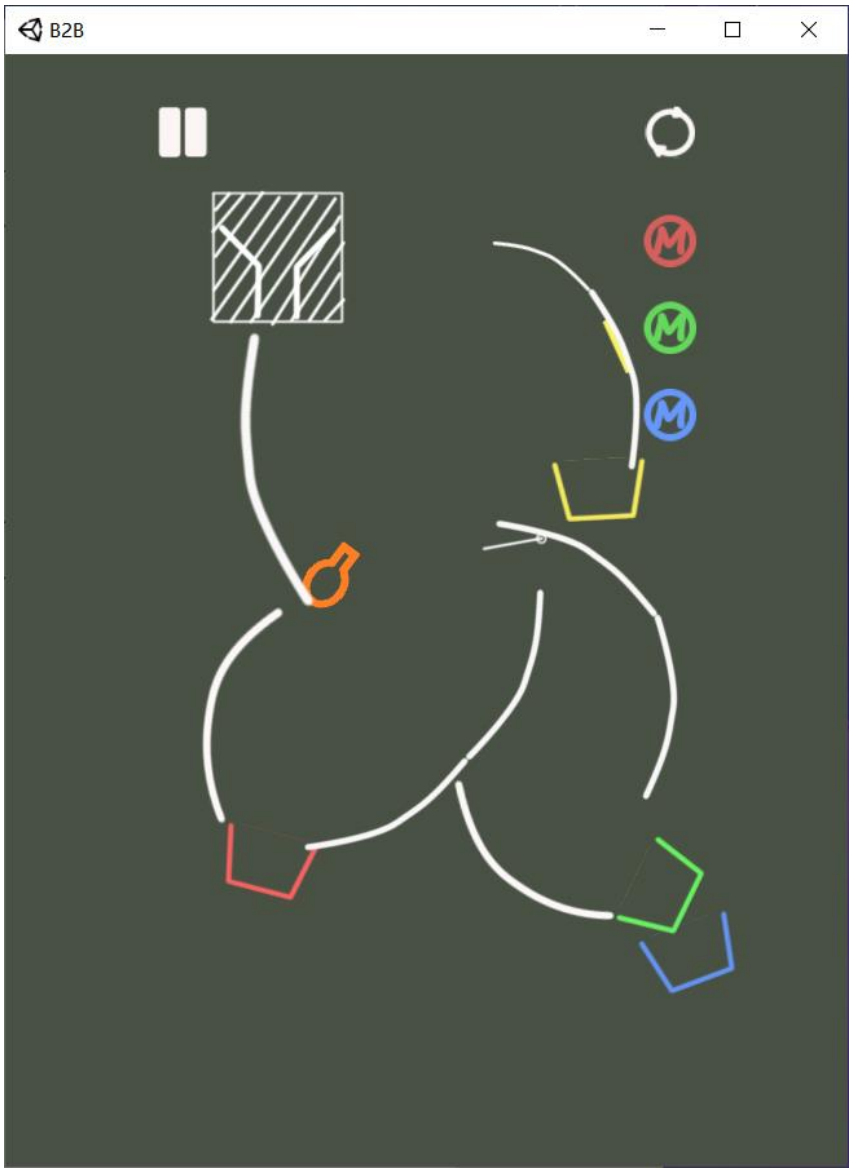
游戏攻略：左入口：红→左入口：绿→右入口：绿→右入口：红

7.2.6 Level 6



游戏攻略：入口：蓝（左加速瓶 1 右加速瓶 1）→入口：红（左加速瓶 1 右加速瓶 1）→入口：绿（左加速瓶 1 右加速瓶 1）

7.2.7 Level 7

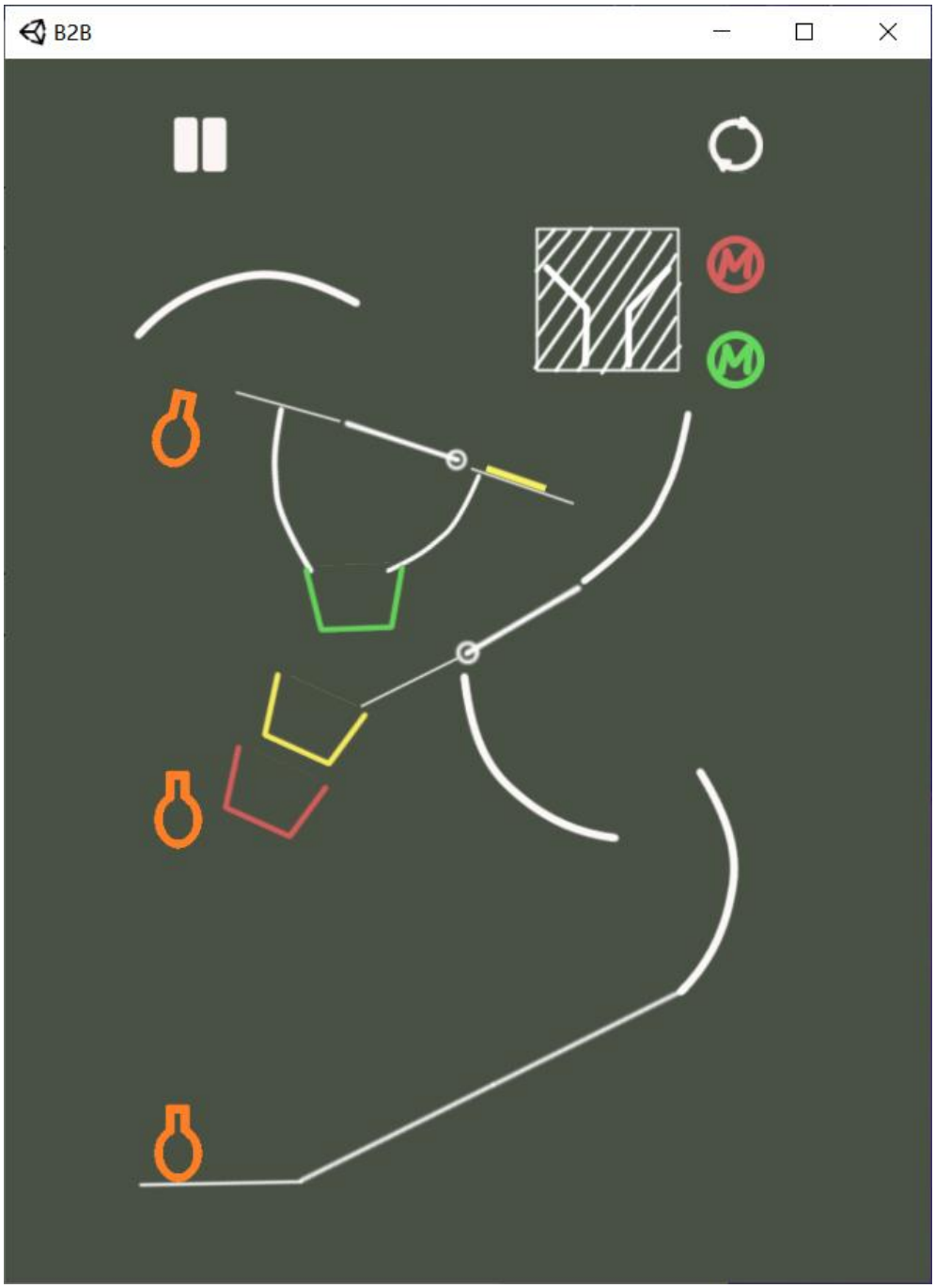


游戏攻略：入口：任意（加速瓶 1）→入口：蓝（加速瓶 2）→入口：红（左加速瓶 1 右加速瓶 1）

7.2.8 Level 8

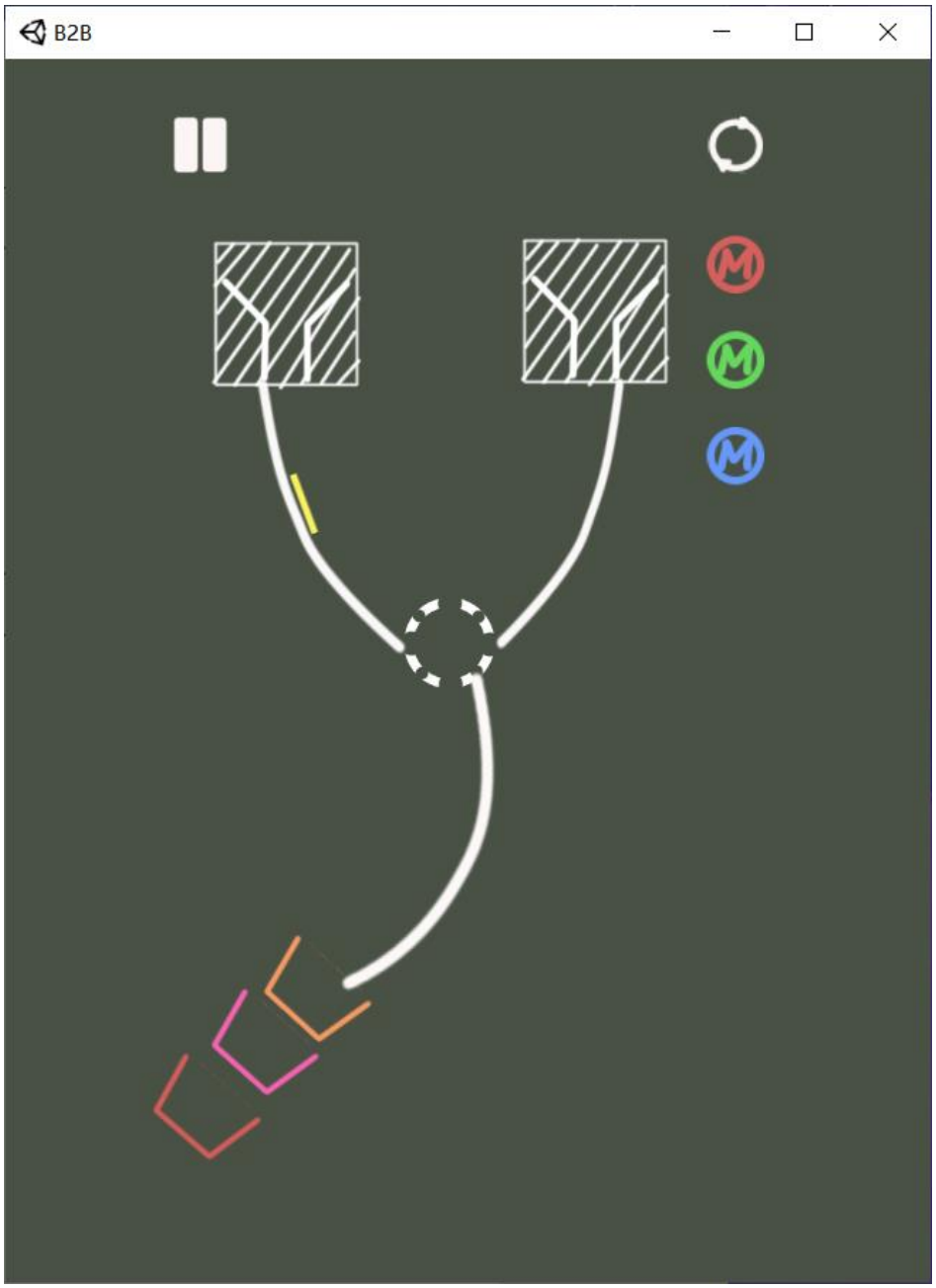
游戏攻略：入口：红（上加速瓶 1）→入口：蓝（加速瓶 2）→入口：红（左加速瓶 1 右加速瓶 1）

7.2.9 Level 9



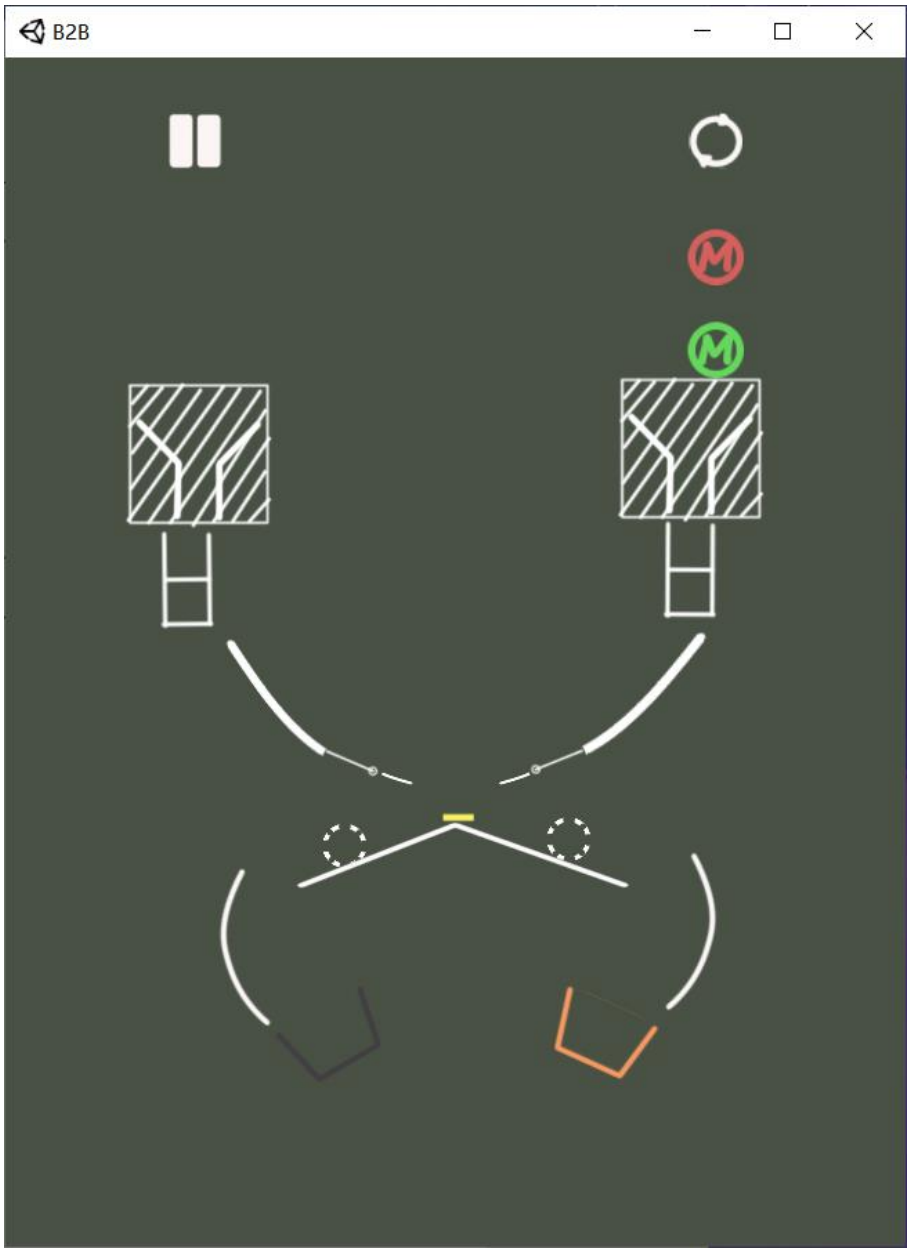
游戏攻略：入口：红→入口：任意→入口：绿

7.2.10 Level 10



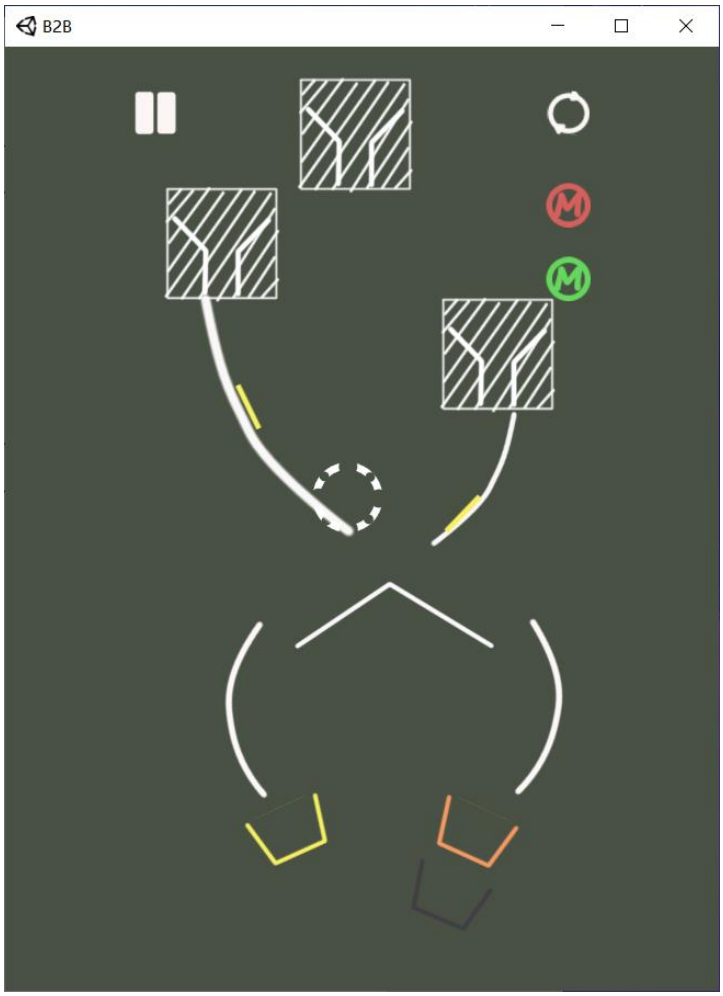
游戏攻略：右入口：红→右入口：红→右入口：红→右入口：蓝

7.2.11 Level 11



游戏攻略：左入口：绿→左入口：绿→右入口：红→右入口：红

7.2.12 Level 12



游戏攻略：右入口：任意→左入口：任意→中入口：绿→左入口：任意→中入口：红

8 总结（Conclusion）

在这段超过三个月的时间里，我们组的成员们都在这款游戏上花了很多时间和精力。我们在这个过程中一起进步，一起学习。从开始项目的选题到编译环境的选择，我们都给出了自己的意见，并且达成了共识。我们成功地实现了暂停、重玩、小球生成、小球拖拽、小球销毁、生成物理小球、小球轨道设置、篮子承接、胜利失败判定、计时器与积分、开关、篮子堆叠、翻斗、染色器与混色器、加速器、退出游戏和切换界面等功能。 从一开始对 Unity2D 的学习，再到后来对游戏界面的搭建，对游戏物体道具的增加，我们都遇到了很多大大小小的问题，但是我们耐心地一点点地调试与重整，最终完成了项目。我们能够在这么短的时间内取得进步，是因为我们经常在一起讨论和交流想法。在交流过程中不断更新和成熟我们的程序代码，使我们的游戏运行得更加顺利，内容更加丰富。我们使用 unity 开发了一个有关小球运动与碰撞的物理休闲益智类游戏，并在其中加入诸如混色、叠加等设计增加其可玩性，再配以设计的烧脑关卡达到寓教于乐、在游戏中锻炼逻辑的目的，取名为 B2B（Ball to Basket）。这款游戏的难度适中，轻松解压，对运行条件要求较低，适合大家在业余时间体验。总的来说，我们通过这一学期对 C 语言课程的学习和小组项目的制作，掌握了 C 语言和 C#语言的基本内容和语法，并能够根据个人需求灵活使用来实现预期效果，我们的编程能力和程序设计能力在课程与项目中大大提高！