Group 5

B2B
×
(Ball to Basket)

申奥 朱子琦 高庆霖

CONTENT

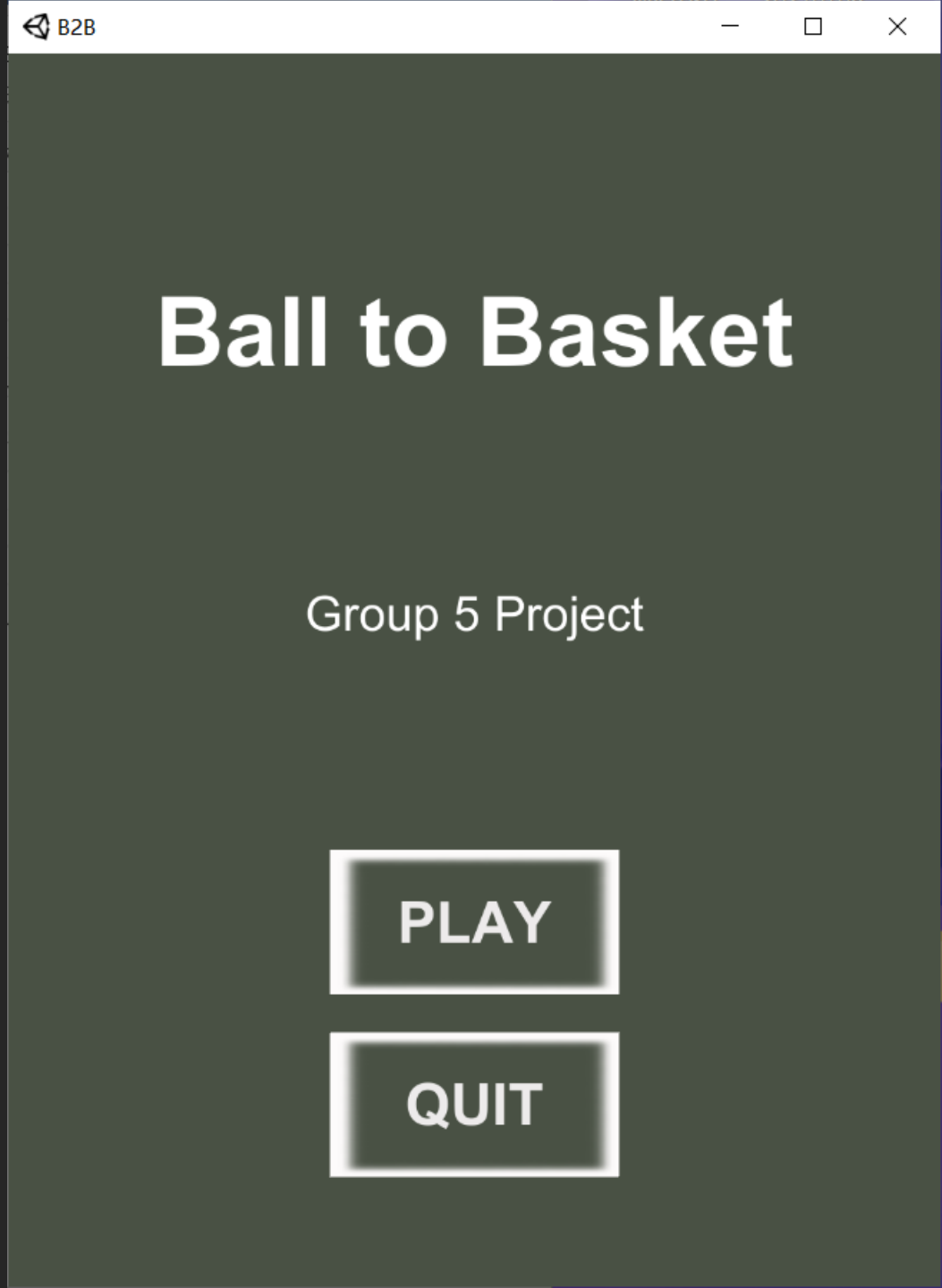

01
Test

Problem Statement

Problem Statement

一款基于物理引擎设计的益智类游戏

- ✓ 2D游戏开发
- ✓ 益智类游戏
- ✓ 物理引擎

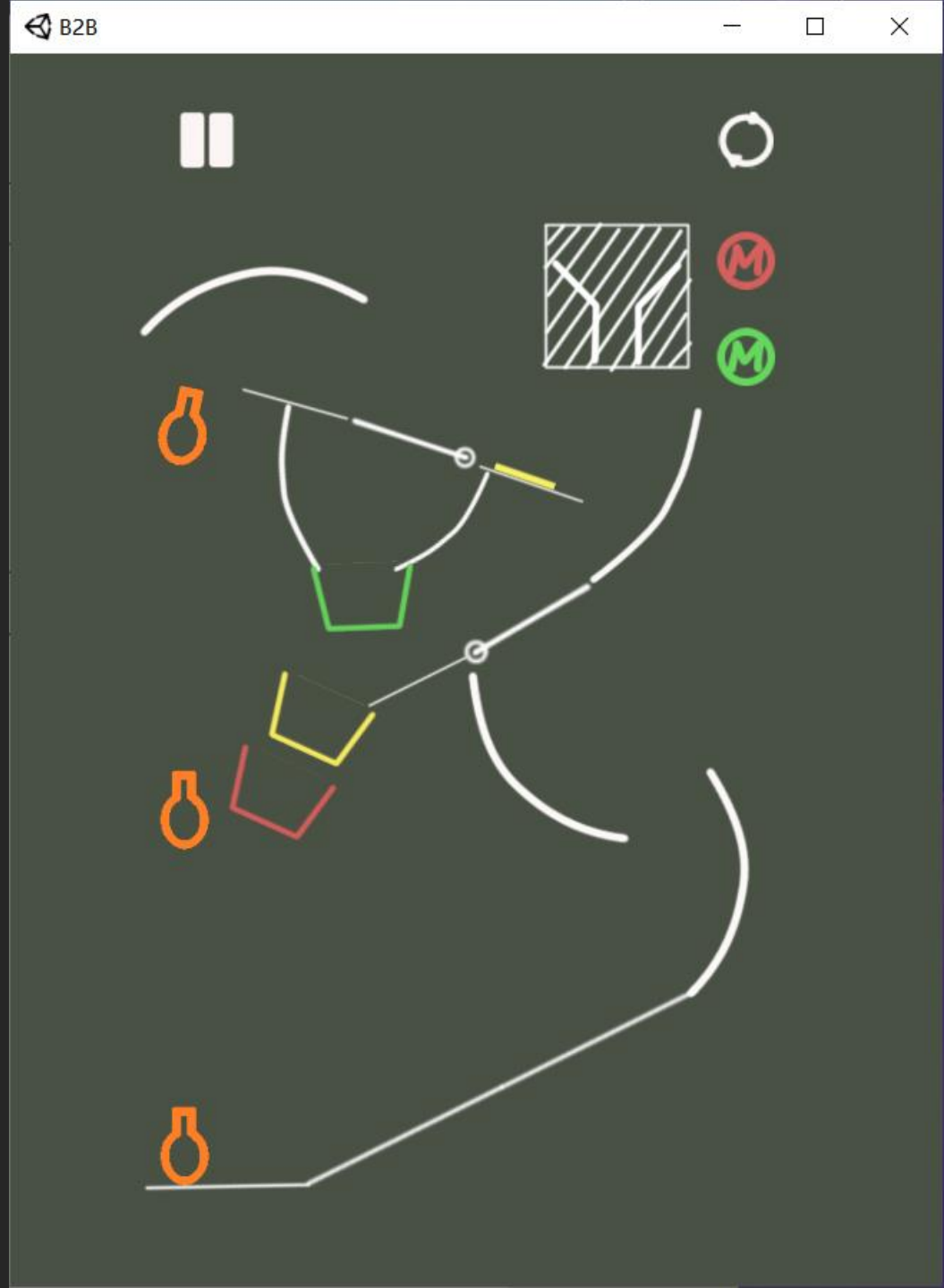



02
Analysis&
Design

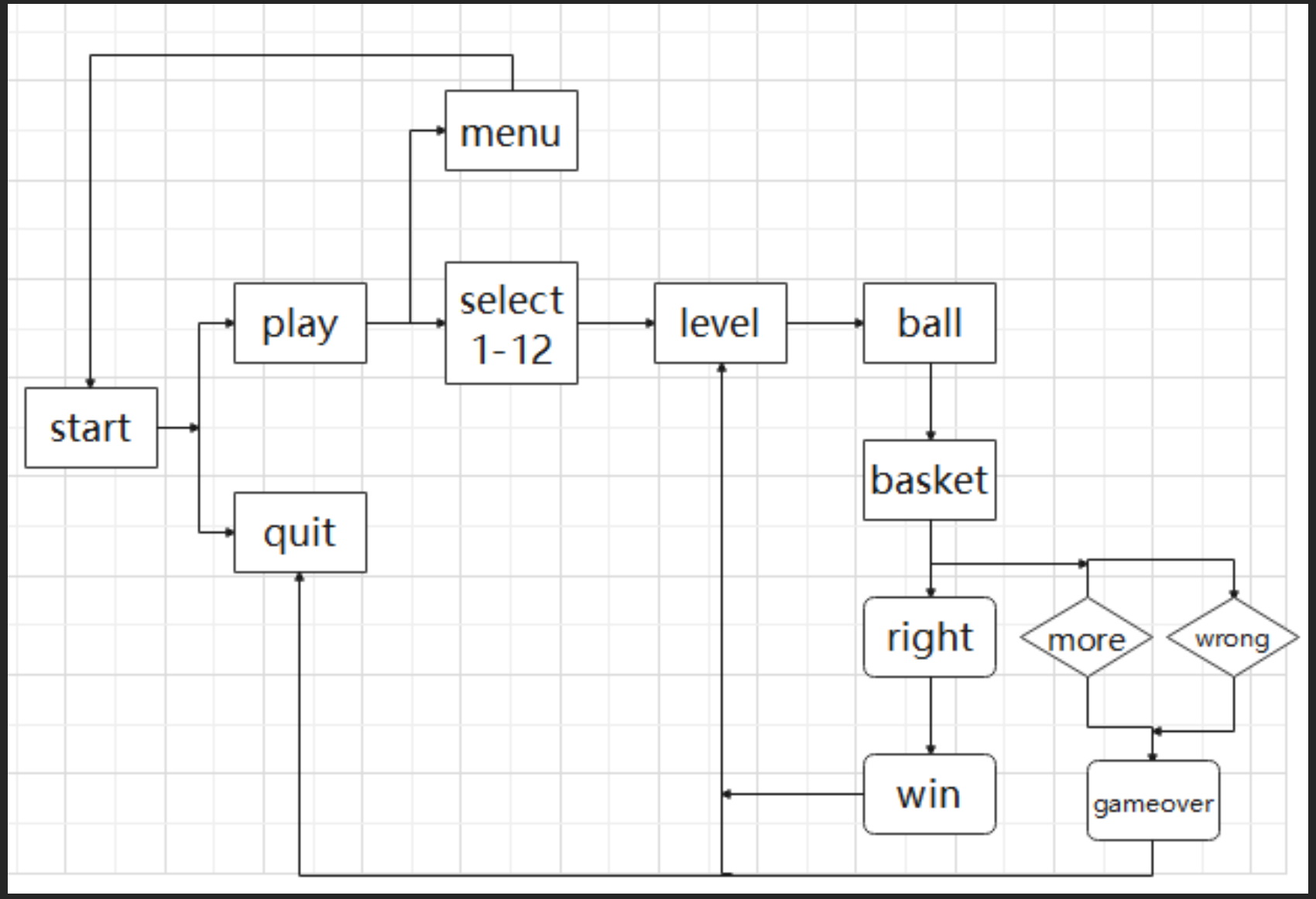
Analysis

休闲益智类游戏

- ✓ 休闲放松
- ✓ 益智
- ✓ 单机



Design


menu
play
select
1-12
level
ball
start
basket
quit
right
more
wrong
win
gameover

基本功能的实现

游戏界面与预制的实现

物理碰撞的实现

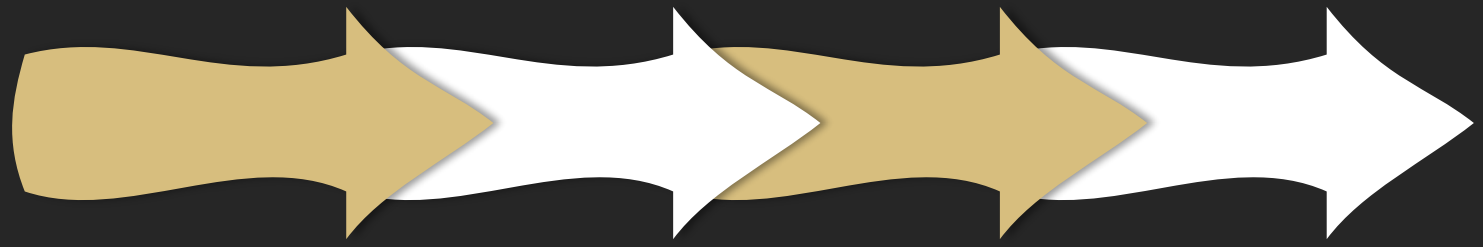
胜利失败界面的实现

附加零件增加可玩性

开关 加速器 翻斗

篮子堆叠

染色器与混色器



更多关卡的设计

益智关卡的设计与实现....

关卡的积累与升级

各关卡进行关联

开局界面


04
Group Division

Group Division

申 奥： 物理碰撞的实现（35%） 开关 加速器 翻斗关卡的制作

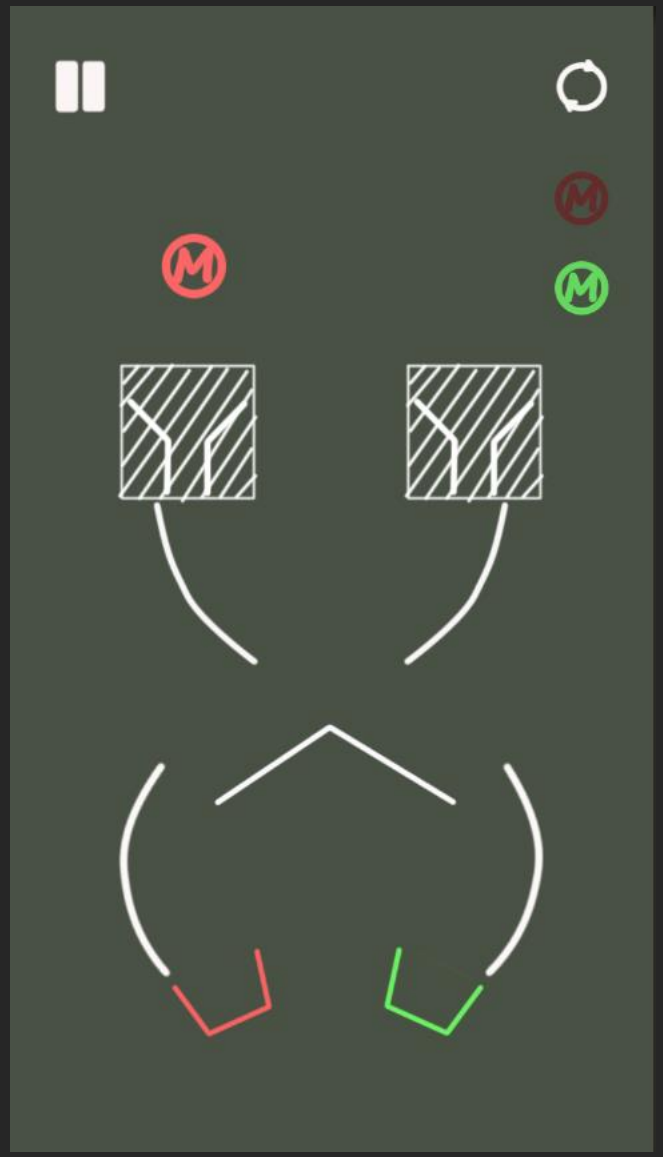
朱子琦：UI布局与预制生成（30%）篮子堆叠 染色器 混色器关卡的制作

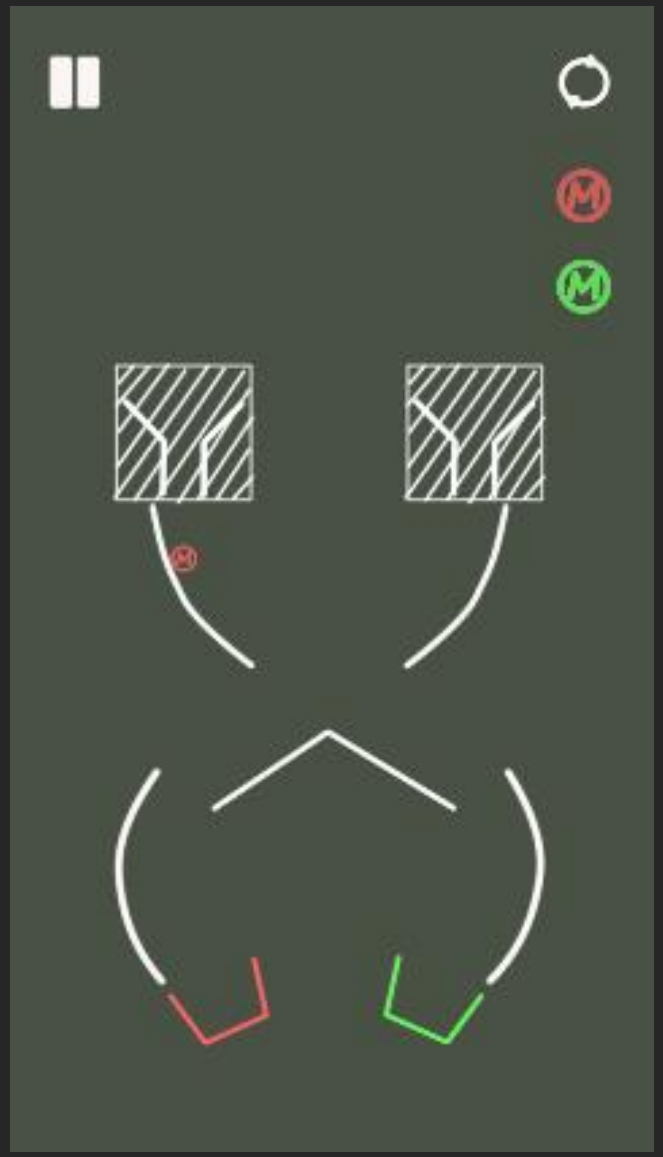
高庆霖：胜利失败的判定与实现（35%）关卡的积累与升级

05

Implementation

小球的产生与射线检测





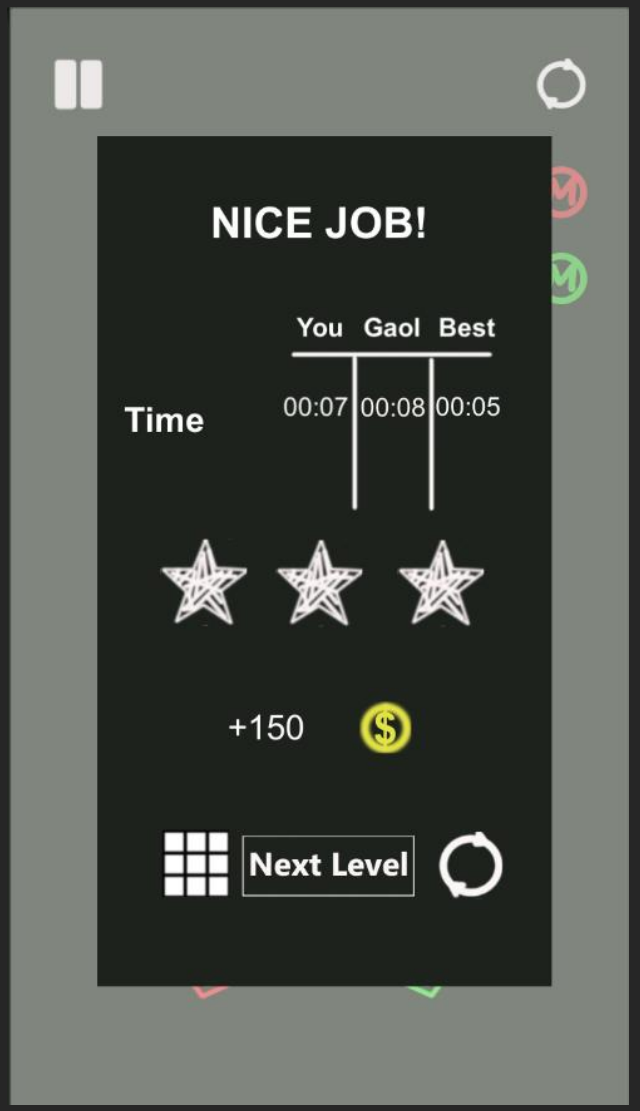
```
public class DragBall : MonoBehaviour
{
    //鼠标位置
    private Vector3 currentMousePositi
    // Start is called before the firs
    0 个引用
    void Start()
    {

    }

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

        //小球跟随鼠标
        this.transform.position = curr
    }
```

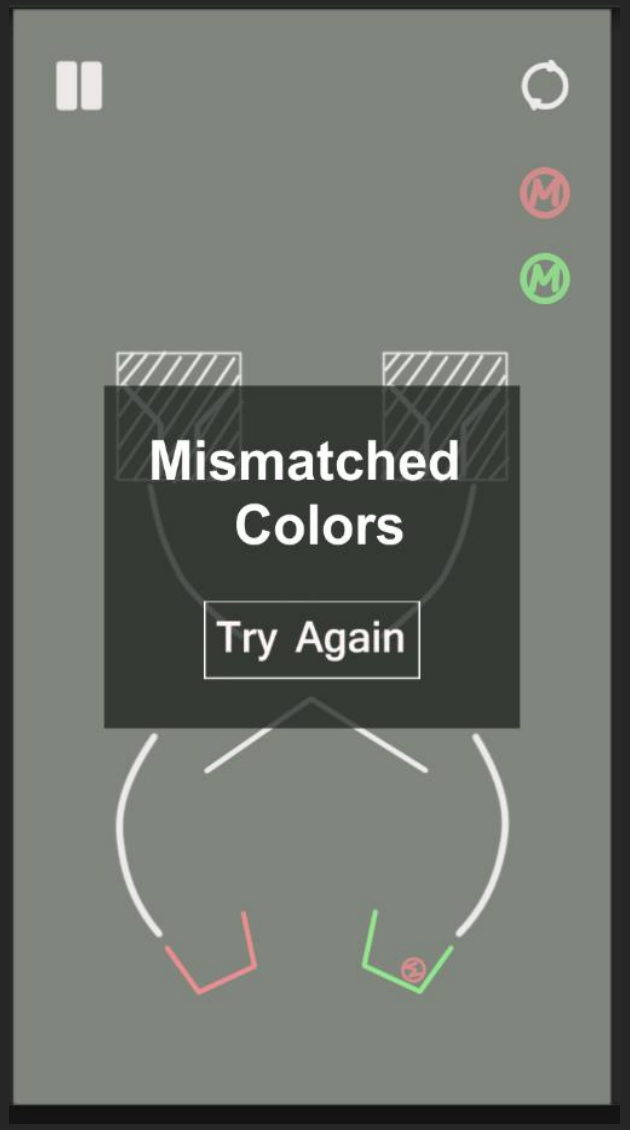
```
public void MarbleButtonDown()
{
    //计算鼠标当前坐标
    currentMousePosition = Camera.main.ScreenToWorldPoint(Input.mousePosition);
    currentMousePosition.z = 0;

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

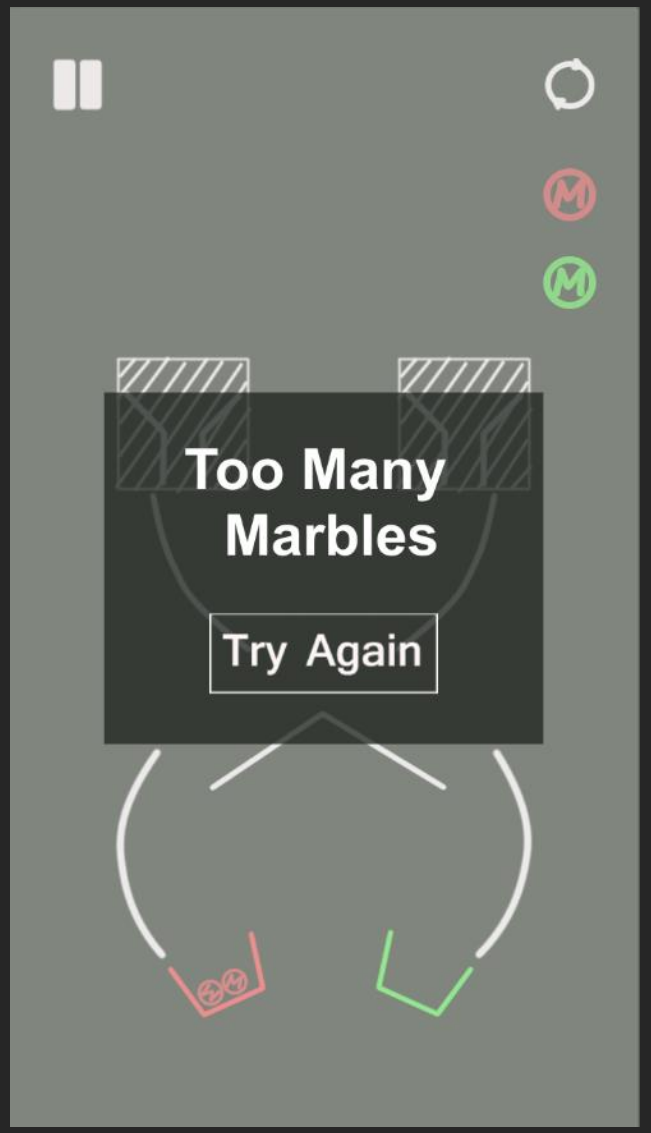
```
//  执行鼠标松开
) 个引用
ublic void MarbleButtonUp()
{
    //销毁临时小球
    Destroy(tempDragBall);

    //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),
```

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

胜利与失败


NICE JOB!
You Gaol Best
Time
00:07 00:08 00:05
+150
Next Level


Mismatched Colors
Try Again

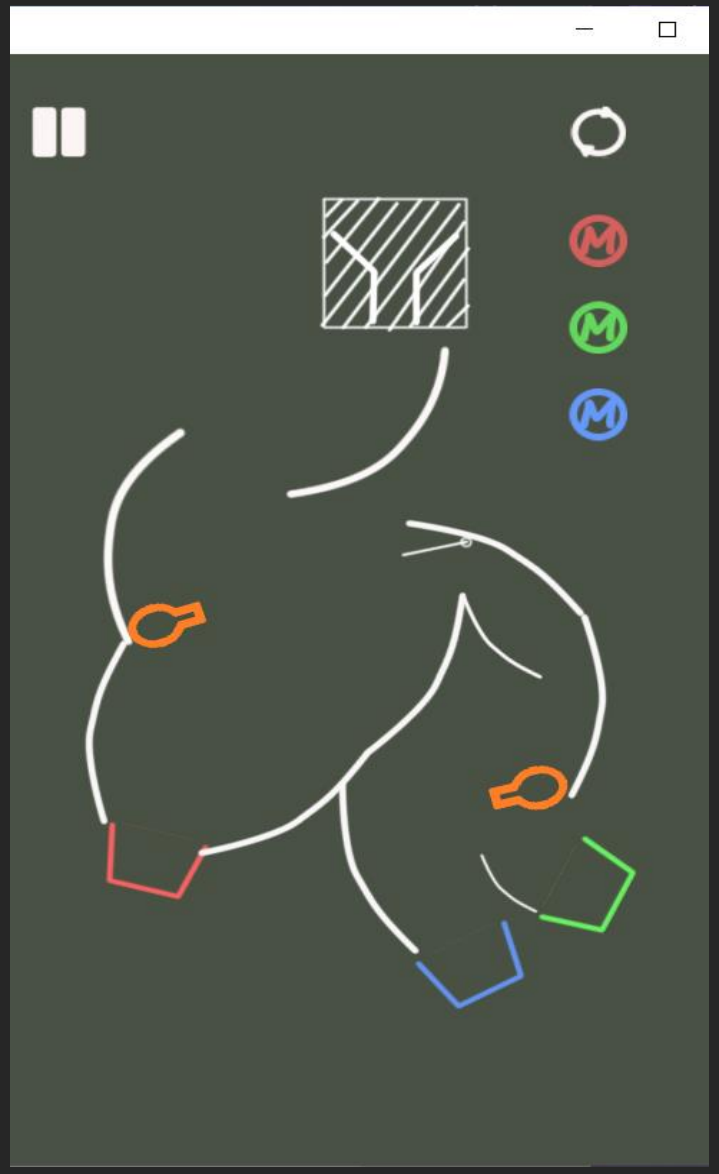

Too Many Marbles
Try Again

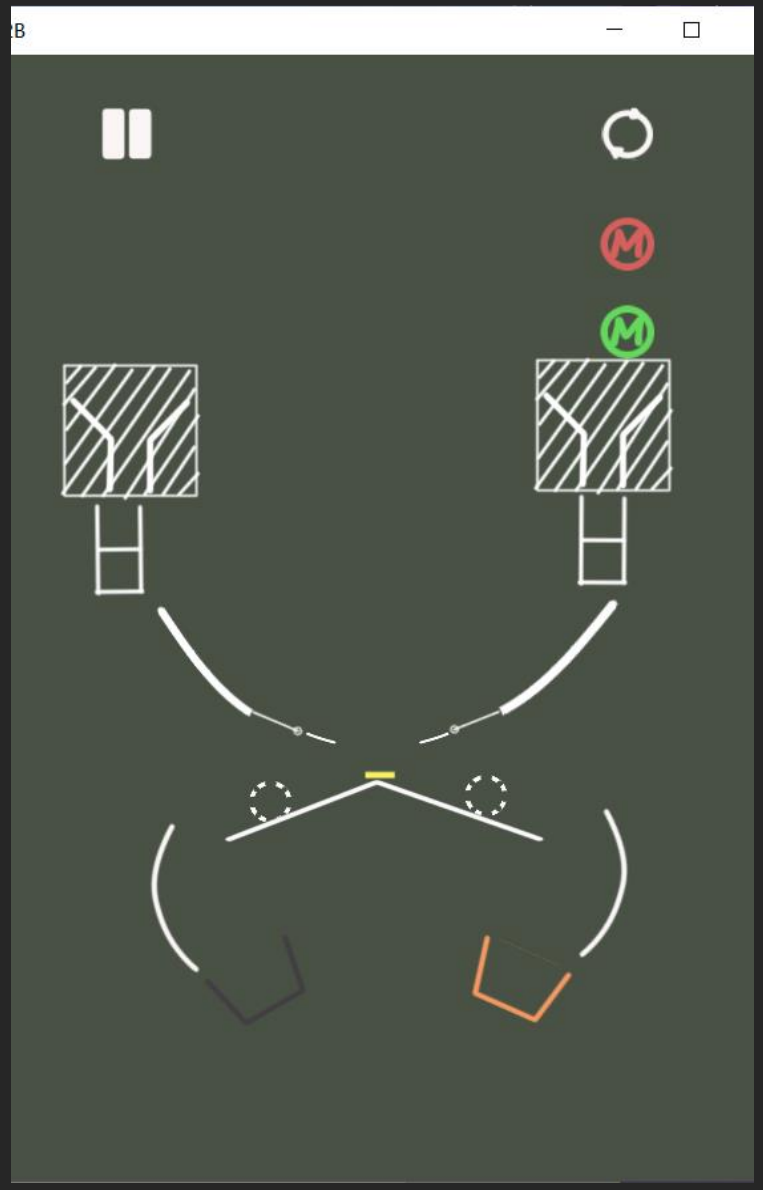
```
//如果小球进入
if (coll.gameObject.layer == Lay
    selfPolyCollider2D.enabled =
{   //如果篮子空
    if (basketState == 0)
    {
        //获取目标小球physicsbal
        PhysicsBall tempPhysicsB
        //如果颜色对
        if (this.colorIndex == t
        {
            basketState = 1;
        }
        else
        {
        basketState = -1;
        }
    }
    else
    {
        basketState = -2;
    }
```

```
private void OnEnable()
{
    //如果颜色错误
    if (Myclass.currentGameState == GameState.ColorError)
    {
        //显示失败信息
        defeatI
    }
    //如果数目错
    if (Myclass.
    {
        //显示失
        defeatI
    }
```

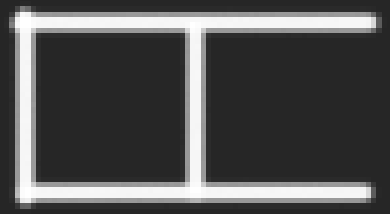
```
private void DisplayCoin()
{
    //本局获得
    int tempCoin = 50 * starLevel;
    //显示
    coinNumText.text = "+" + tempCoin;

    //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);
}
```

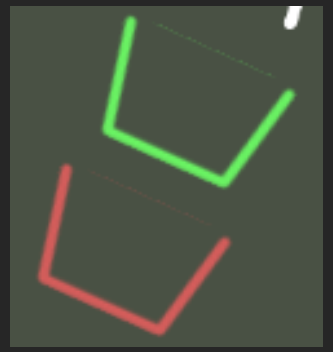
特殊道具





特殊道具

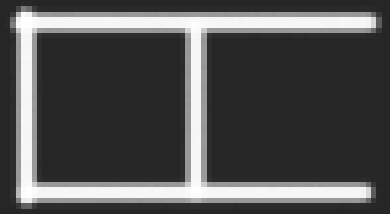




开关（switch）

```
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);
    }
```

翻斗（bucket）



```
void Start()
{
    //初始时，翻斗不
    rotateEnable = fa
    //衫
    ang
    //衫
    ent
    exi
    //i
    ori
}
```

```
void Update()
{
    //如果翻斗进入
    if (rotateEnab
    {
        //如果小球
        if (rotate
        {
            //旋转
            this.t

            //角度
            angleC
        }
```

```
//如果小球已经倒出
if (rotateState == 1)
{
    //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;
    }
```

加速瓶（bottle）

```
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];
            }
        }
```

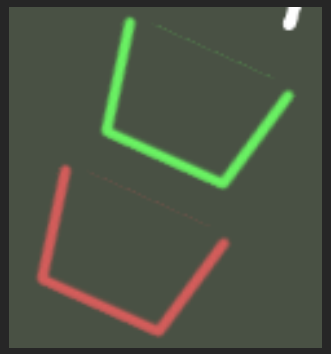
混色器（colormixer）

```
//游戏中
public s
//小
ente
//如
if (
{
}
//如果进入的物体是物理小球
if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
{
    //如果进入的是第二个小球
    if (enterBallNum == 2)
    {
        //速度为0
        coll.attachedRigidbody.velocity = Vector2.zero;

        //重力系数为0
        coll.attachedRigidbody.gravityScale = 0;

        //位置修正
        coll.transform.position = this.transform.position;

        //记录该小球
        secondBall = coll.gameObject;

        //获取该小球的颜色索引
        secondColorIndex = secondBall.GetComponent<PhysicsBall>().colorIndex;

        //计算混合之后的小球的颜色索引
        mixedColorIndex = Myclass.ColorMixingTable[firstColorIndex, secondColorIndex];

        //销毁第二个小球
        Destroy(secondBall);

        //修改第一个小球的颜色索引
        firstBall.GetComponent<PhysicsBall>().colorIndex = mixedColorIndex;

        //修改第一个小球的图片颜色
        firstBall.GetComponent<SpriteRenderer>().color = Resources.Load<SpriteRenderer>("Prefab/PhysicsBa
```

染色器（dye）

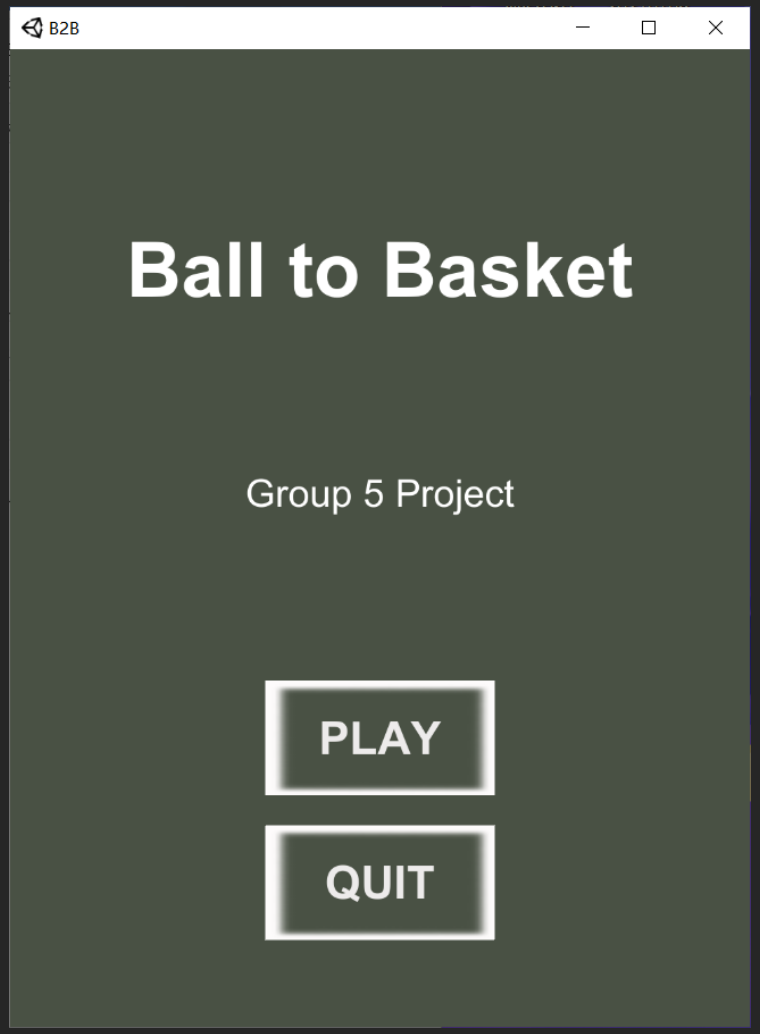
```
//当别的碰撞体进入染色器的触发器时
0 个引用
void OnTriggerEnter2D(Collider2D coll)
{
    //如果进入的物体是物理小球
    if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
    {
        //修改物理小球的颜色索引
        coll.gameObject.GetComponent<PhysicsBall>().colorIndex = this.colorIndex;

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

篮子堆叠

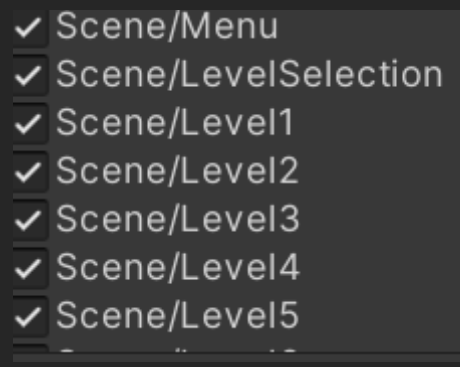


```
//当其他的碰撞体离开该物体时，执行的操作
0 个引用
void OnTriggerExit2D(Collider2D coll)
{
    //如果进入的碰撞体是物理小球
    if (coll.gameObject.layer == LayerMask.NameToLayer("PhysicsBall"))
    {
        //离开小球数+1
        exitBallNum++;

        //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
        selfPolyCollider2D.enabled = (exitBallNum >= pileIndex);
    }
```

关卡积累与开局界面


B2B
Ball to Basket
Group 5 Project
PLAY
QUIT


B2B
Level Selection
1
2
3
4
5
6
7
8
9
10
11
12
Menu




```
public class MainMenu : MonoBehaviour
{   //进入下一关
    0 个引用
    public void PlayButtonClick()
    {
        SceneManager.LoadScene(SceneMa
    }

    0 个引用
    public void QuitButtonClick()
    {   //退出程序

        Application.Quit();
    }
```

```
public class LevelChoose : MonoBehaviour
{


    //选关函数
    0 个引用
    public void LoadScene(string sceneName)
    {
        SceneManager.LoadScene(sceneName);
    }
```


ADD TITLE
GOOD
×
BYE
INPUT