文件: c:\Users\Administrator\mathpractice\client\Assets\Scripts\cameractrl.cs

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

public class cameractrl : MonoBehaviour
{
public float distance = 3;

private float currentX = 0.0f; // 当前水平旋转角度
private float currentY = 0.0f; // 当前垂直旋转角度
// Start is called before the first frame update
void Start()
{
// 初始化旋转角度
Vector3 angles = transform.eulerAngles;
currentX = angles.y;
currentY = angles.x;

}

// Update is called once per frame
void Update()
{
Vector3 direction = new Vector3(0, 0, -distance);
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\cameractrl.cs

```
Quaternion rotation = Quaternion.Euler(currentY, currentX, 0);


// ■■■■■■■■

Vector3 targetPosition = new Vector3(0, 0, 0);// target.position + Vector3.up * height;

Vector3 cameraPosition = targetPosition + rotation * direction;


// ■■■■■■■■■

transform.position = cameraPosition;

transform.LookAt(targetPosition);

}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\DispMsg.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class DispMsg : MonoBehaviour
{
public string msg = "event_";
// Start is called before the first frame update
void Start()
{
GetComponent<Button>().onClick.AddListener(() =>
{
Main.DispEvent(msg);
});
}

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

}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\extends.cs

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

public static class extends
{
// Start is called before the first frame update
public static void Clear(this Transform trans)
{
while (trans.childCount > 0)
{
var c = trans.GetChild(0);
c.transform.SetParent(null);
if (Application.isPlaying)
{
GameObject.Destroy(c.gameObject);
}
else
{
GameObject.DestroyImmediate(c.gameObject);
}
}
}
public static T GetOrAddComponent<T>(this GameObject go) where T : Component
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\extends.cs

```
{
var c = go.GetComponent<T>();
if (c == null)
{
c = go.AddComponent<T>();
}
return c;
}
public static string ReplaceAll(this string str,string p1,string p2)
{
string ret = str.Replace(p1, p2);
while (ret.IndexOf(p1) > -1)
{
ret = ret.Replace(p1, p2);
}
return ret;
}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frmbase.cs

using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class frmbase : MonoBehaviour

{

// Start is called before the first frame update

private void Awake()

{


}

void Start()

{


}


// Update is called once per frame

void Update()

{


}

Transform gb

{
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frmbase.cs

```
get
{
return transform.Find("gb");
}
}
public void show()
{
gb.gameObject.SetActive(true);
OnShow();
}
protected virtual void OnShow()
{

}
protected virtual void OnHide()
{

}
public void hide()
{
gb.gameObject.SetActive(false);
OnHide();
}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\levelmgr.cs

```
■using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class levelmgr : MonoBehaviour

{

public static int level {

get

{

return PlayerPrefs.GetInt("level", 1);

}

set

{

PlayerPrefs.SetInt("level", value);

}

}


internal static void init()

{

level = 1;

}

/**

* ■■■■■■
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\levelmgr.cs

* @param level ■■

* @return ■■

* ■■■■1■100■

*

*/

internal static float getTimeAll(int level)

{

float t = level / (float)realmaxlevel;

return Mathf.Lerp(30, 100, t);

}

/**

* ■■■■■■■

* */

internal static int getMin(int level)

{

return level;

}

static int maxlevel = 50;

public static int maxcount = 30;


public static int realmaxlevel

{

get

{
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\levelmgr.cs
return maxlevel-maxcount;
}
}
/**
* ■■■■■■
* */
internal static int getCount(int level)
{
float t= level /(float)maxlevel;
return (int)Mathf.Lerp(15,realmaxlevel,t);
}
internal static int getWid(int level_playing)
{
float t = level_playing / (float)(realmaxlevel);
if (t < 0.2f)
{
return 4;
}
else if (t < 0.6f)
{
return 6;
}
else
{
```

```
return 8;
}
}

internal static int getHei(int level_playing)
{
float t = level_playing / (float)realmaxlevel;
if (t < 0.2f)
{
return 8;
}
else if (t < 0.6f)
{
return 10;
}
else
{
return 12;
}
}

internal static float getSource(int level_playing)
{
float t = level_playing / (float)realmaxlevel;
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\levelmgr.cs

```
return Mathf.Lerp(1000, 10000000, t);

}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Main.cs

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

public class Main : MonoBehaviour
{
public delegate object registfun(object parm);
// Start is called before the first frame update
public static Main inst;
private void Awake()
{
inst = this;
}
void Start()
{
float bl = ((float)Screen.width) / Screen.height;
Screen.SetResolution((int)(1920*bl),1920,false);
DispEvent("gamebegin");
}
static Dictionary<string, List<registfun>> evs = new();
public static object DispEvent(string ev, object parm = null)
{
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Main.cs
if (evs.ContainsKey(ev))
{
for (int i = 0; i < evs[ev].Count; i++)
{
var p = evs[ev][i](parm);
if (p != null)
{
return p;
}
}
Debug.LogError("■■■■■■" + ev);
return null;
}
else
{
return null;
}
}
public static void RegistEvent(string ev, registfun fun)
{
if (evs.ContainsKey(ev))
{
//Debug.LogError("■■■■■■" + ev);
}
```

```
else
{
evs[ev] = new List<registfun>();
}
evs[ev].Add(fun);
}
public static void RemoveEvent(string ev, registfun fun)
{
if (evs.ContainsKey(ev))
{
evs[ev].Remove(fun);
}
}
// Update is called once per frame
void Update()
{

}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\musicmgr.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class musicmgr : MonoBehaviour
{
public AudioClip main, play,other,win,faild;
public AudioSource source
{
get
{
return GetComponent<AudioSource>();
}
}
// Start is called before the first frame update
void Awake()
{
Main.RegistEvent("gamebegin", (x) =>
{
source.clip = main;
source.Play();
return null;
});
Main.RegistEvent("event_play", (x) =>
{
source.clip = play;
source.Play();
return null;
});
Main.RegistEvent("game_win", (x) =>
{
source.clip = win;
source.Play();
return null;
});
Main.RegistEvent("game_lose", (x) =>
{
source.clip = faild;
source.Play();
return null;
});
Main.RegistEvent("event_music", (a) =>
{
source.mute = !bMusicEnable;
return null;
});
}
private void Start()
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\musicmgr.cs

```
{
source.mute = !bMusicEnable;
}
bool bMusicEnable
{
get
{
return PlayerPrefs.GetInt("music", 1) == 1;
}
}
// Update is called once per frame
void Update()
{

}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
■using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using DG.Tweening;
using System;

public class playctrl : MonoBehaviour
{
public Material mat1;
public Material mat2,mat3,mat4,mat5,mat6,mat7,mat8,mat9,mat10;
public GameObject prefab;
private void Start()
{

}

/// <summary>
/// ■■■■Box■■■■■■■■■■■■■■■■■■■■■■
/// </summary>
public void PlayOnePlusOne(int v, int v1)
{
if (_play != null)
{
StopCoroutine(_play);
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs
}



for (int i = 0; i < alst.Count; i++)
{
GameObject.Destroy(alst[i]);
}
alst.Clear();
for(int i = 0; i < blst.Count; i++)
{
GameObject.Destroy(blst[i]);
}
blst.Clear();
clst.Clear();

_play = play(v, v1);

StartCoroutine(_play);
}
IEnumerator _play;
internal void PlayOneSubOne(int v, int v1)
{
if (_play != null)
{
```

```
StopCoroutine(_play);

}




for (int i = 0; i < alst.Count; i++)

{

GameObject.Destroy(alst[i]);

}

alst.Clear();

for (int i = 0; i < blst.Count; i++)

{

GameObject.Destroy(blst[i]);

}

blst.Clear();

clst.Clear();


_play = playsub(v, v1);


StartCoroutine(_play);

}


cameractrl _ctrl;


cameractrl ctrl
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
{
get
{
if (_ctrl == null)
{
_ctrl = Camera.main.gameObject.GetComponent<cameractrl>();//.GetComponent<cameractrl>;
}
return _ctrl;
}
}
List<GameObject> alst = new List<GameObject>();
List<GameObject> blst = new List<GameObject>();
List<GameObject> clst = new List<GameObject>();
void brushctrl()
{
int hei = 0;
hei = Mathf.Max(hei, alst.Count);
hei = Mathf.Max(hei, blst.Count);
hei = Mathf.Max(hei, clst.Count);
//■■dotween■ctrl■distance■■■■■hei*2
DOTween.To(() => ctrl.distance, x => ctrl.distance = x,10/*■■■■*/+ hei * 4, 0.5f).SetEase(Ease.OutQuad)
}


/**
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs
* ■■■■
* */
IEnumerator playsub(int a, int b)
{
//■■■■■
for (int i = 0; i < a; i++)
{
//yield return new WaitForSeconds(1);
GameObject boxObject1 = GameObject.Instantiate(prefab);
boxObject1.transform.position = new Vector3(-3, i * 2, 0);

alst.Add(boxObject1);
brushctrl();
}
//■■■■
//for (int i = 0; i < b; i++)
//{
//    //yield return new WaitForSeconds(1);
//    GameObject boxObject2 = GameObject.Instantiate(prefab);
//    boxObject2.transform.position = new Vector3(3, i * 2, 0);
//    blst.Add(boxObject2);
//    brushctrl();
//}
brushcolor();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
yield return new WaitForSeconds(1);

//■■■■■■■■■■■
for (int i = 0; i < alst.Count; i++)
{
Vector3 targetPosition = new Vector3(0, i * 2, 0);
alst[i].transform.DOMove(targetPosition, 1.0f).SetEase(Ease.OutQuad);
}

brushcolor();

yield return new WaitForSeconds(1);

//■■■■■■■■■■■■■■■■■■■■■■
//■■■■■■■■■■■■■■■■■■■■■■■■
for(int i = 0; i < b;i++)
{
clst.Add(alst[0]);
alst.RemoveAt(0);
}


//■■■■■■■■■■■■■■■■■■■
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
for (int i = 0; i < clst.Count; i++)
{
Vector3 targetPosition = new Vector3(3, i * 2, 0);
clst[i].transform.DOMove(targetPosition, 1.0f).SetEase(Ease.OutQuad);
}
yield return new WaitForSeconds(0.2f);
brushcolor();
yield return new WaitForSeconds(1);

//■■■■■■■■■■■■■■
//for (int i = 0; i < blst.Count; i++)
//{
//     GameObject.Destroy(blst[i]);
//}

////■■blst■■■■■■■■■■
//blst.Clear();
//■clist■■■■■■
for (int i = 0; i < clst.Count; i++)
{
clst[i].GetComponent<MeshRenderer>().material.DOFloat(1, "_Float", 1f).SetEase(Ease.OutQuad);
}
yield return new WaitForSeconds(1);
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
for (int i = 0; i < clst.Count; i++)

{

GameObject.Destroy(clst[i]);

}

clst.Clear();




//■■■■■■■■■■■■■

for (int i = 0; i < alst.Count; i++)

{

Vector3 targetPosition = new Vector3(0, i * 2, 0);

alst[i].transform.DOMove(targetPosition, 1.0f).SetEase(Ease.OutBounce);

}


brushcolor();

brushctrl();


yield return new WaitForSeconds(3);

Main.DispEvent("event_backfromplay");

_play = null;

}


IEnumerator play(int a, int b)

{
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
for(int i = 0; i < a; i++)

{

yield return new WaitForSeconds(1);

GameObject boxObject1 = GameObject.Instantiate(prefab);

boxObject1.transform.position = new Vector3(-3, i*2, 0);


alst.Add(boxObject1);

brushcolor();

brushctrl();

}




for(int i = 0; i < b; i++)

{

yield return new WaitForSeconds(1);

GameObject boxObject2 = GameObject.Instantiate(prefab);

boxObject2.transform.position = new Vector3(3, i*2, 0);

blst.Add(boxObject2);

brushcolor();

brushctrl();

}


yield return new WaitForSeconds(1);
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
for(int i = 0; i < alst.Count; i++)
{
AnimateBoxes(i, alst[i].transform);
}
for(int i = 0; i < blst.Count; i++)
{
AnimateBoxes(i+alst.Count, blst[i].transform);
}
brushctrl();

yield return new WaitForSeconds(3);
Main.DispEvent("event_backfromplay");
_play = null;
}
Material getmat(int count,int index=1)
{
if (count == 1)
{
return mat1;
}
else if (count == 2)
{
return mat2;
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
else if (count == 3)
{
return mat3;
}
else if (count == 4)
{
return mat4;
}
else if (count == 5)
{
return mat5;
}
else if (count == 6)
{
return mat6;
}
else if(count == 7)
{
if (index == 0) return mat1;
if (index == 1) return mat2;
if (index == 2) return mat3;
if (index == 3) return mat4;
if (index == 4) return mat5;
if (index == 5) return mat6;
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

```
return mat7;
}
else if (count == 8)
{
return mat8;
}
else if (count == 9)
{
if (!mat9list.ContainsKey(index))
{
mat9list[index] = Instantiate(mat9);
mat9list[index].color = Color.Lerp(new Color(0.9f, 0.9f, 0.9f), new Color(0.5f, 0.5f, 0.5f), index / 8f);
}
return mat9list[index];
}
else
{
return mat10;
}
}
Dictionary<int, Material> mat9list = new Dictionary<int, Material>();
void brushcolor()
{
for(int i = 0; i < alst.Count; i++)
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs
{
alst[i].GetComponent<MeshRenderer>().material = getmat(alst.Count,i);
}
for (int i = 0; i < blst.Count; i++)
{
blst[i].GetComponent<MeshRenderer>().material = getmat(blst.Count,i);
}

for (int i = 0; i < clst.Count; i++)
{
clst[i].GetComponent<MeshRenderer>().material = getmat(clst.Count,i);
}
}
void AnimateBoxes(int y,Transform box2 )
{
clst.Add(box2.gameObject);
// ■■■■■■■■■■■■■■■■■■■■■■■■
Vector3 targetPosition = new Vector3(0, y*2, 0); // 1.5f■■■■■■■■■■■1■■■■■■■■■■■■1.0f■■■■1.5f

// ■■DOTween■■■■■■
// ■■.DOJump■■■■■■
box2.DOJump(targetPosition, 1.0f, 1, 1.0f) // ■■■■1.0f■1■■■■■■1.0■
.SetEase(Ease.OutQuad)
.OnComplete(() => {
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\playctrl.cs

// ■■■■■■■■■■■■■■■

//box1.GetComponent<Renderer>().material = mat2;

//box2.GetComponent<Renderer>().material = mat2;

brushcolor();

Debug.Log("■■■■■■■■■■■■■■■■■■■■■■■■■■");

});


}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\rot.cs

```
using System.Collections;

using System.Collections.Generic;

using UnityEngine;


public class rot : MonoBehaviour

{

public float speed = 40;

// Start is called before the first frame update

void Start()

{


}


// Update is called once per frame

void Update()

{

transform.localRotation = Quaternion.Euler(0, 0, Time.time * speed);

}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\soundmgr.cs

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

public class soundmgr : MonoBehaviour
{
// Start is called before the first frame update
public AudioSource peng, click,slider;
void Start()
{
Main.RegistEvent("event_lian", (parm) =>
{
if (bSoundEnable)
{
peng.Play();
}

return 1;
});
Main.RegistEvent("event_click", (parm) =>
{
if (bSoundEnable)
{
click.Play();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\soundmgr.cs

```
}
return 1;
});
Main.RegistEvent("event_sound", (parm) =>
{

return null;
});

Main.RegistEvent("event_slider", (parm) =>
{
if (bSoundEnable)
{
slider.Play();
}
return null;
});
}
bool bSoundEnable
{
get
{
return PlayerPrefs.GetInt("sound", 1) == 1;
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\soundmgr.cs

```
}

// Update is called once per frame

void Update()

{


}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Tools.cs

```
using System;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;


public class Tools : MonoBehaviour

{

internal int times;

public UnityEngine.UI.Button.ButtonClickedEvent eventx;


// Start is called before the first frame update

void Start()

{

GetComponent<Button>().onClick.AddListener(() =>

{

eventx.Invoke();

Main.DispEvent("event_click");

});

}


// Update is called once per frame

void Update()

{
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Tools.cs

```
}

internal void reset()
{
times = 1;
gameObject.SetActive(true);
}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Editor\frmbaseeditor.cs

```
■using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEditor;
[CustomEditor(typeof(frmbase),true)]
public class frmbaseeditor : Editor
{
// Start is called before the first frame update
public override void OnInspectorGUI()
{
base.OnInspectorGUI();
if (GUILayout.Button("■■"))
{
frm.show();
}
if (GUILayout.Button("■■"))
{
frm.hide();
}
}
frmbase frm
{
get
{
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Editor\frmbaseeditor.cs

```
return target as frmbase;

}

}

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Editor\NewFile.cs

```
using UnityEngine;

using UnityEditor;

using System;


public class PlayerPrefsTools : EditorWindow

{

[MenuItem("Tools/PlayerPrefs/Delete All PlayerPrefs")]

public static void DeleteAllPlayerPrefs()

{

PlayerPrefs.DeleteAll();

PlayerPrefs.Save();


Debug.Log("■■ PlayerPrefs ■■■");

}


[MenuItem("Tools/PlayerPrefs/Open PlayerPrefs Window")]

public static void ShowWindow()

{

GetWindow<PlayerPrefsTools>("PlayerPrefs ■■");

}


void OnGUI()

{

GUILayout.Label("PlayerPrefs ■■", EditorStyles.boldLabel);
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\Editor\NewFile.cs

```
if (GUILayout.Button("■■■■"))
{
OpenPlayerPrefsDirectory();
}
if (GUILayout.Button("■■■■ PlayerPrefs"))
{
DeleteAllPlayerPrefs();
}

if (GUILayout.Button("■■■■■■"))
{
Debug.Log("■■■■: " + PlayerPrefs.GetInt("level", 1));
}
}

private void OpenPlayerPrefsDirectory()
{
//■■■■■■■
UnityEditor.EditorUtility.OpenWithDefaultApp(Application.persistentDataPath);
}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_chengjiu.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
public class frm_chengjiu : frmbase
{
public Transform content;
public TextMeshProUGUI additionAccuracyText; // ■■■■■■■
public TextMeshProUGUI subtractionAccuracyText; // ■■■■■■■
public TextMeshProUGUI overallAccuracyText; // ■■■■■■■
public TextMeshProUGUI mainall;
public TextMeshProUGUI details;
/// <summary>
/// ■■■■■■■■■
/// </summary>
void UpdateStatisticsDisplay()
{
// ■■■■■■
int totalAddition = PlayerPrefs.GetInt("TotalAdditionQuestions", 0);
int correctAddition = PlayerPrefs.GetInt("CorrectAdditionQuestions", 0);
int totalSubtraction = PlayerPrefs.GetInt("TotalSubtractionQuestions", 0);
int correctSubtraction = PlayerPrefs.GetInt("CorrectSubtractionQuestions", 0);
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_chengjiu.cs

// ■■■■■

float additionAccuracy = totalAddition > 0 ? (float)correctAddition / totalAddition * 100 : 0;

float subtractionAccuracy = totalSubtraction > 0 ? (float)correctSubtraction / totalSubtraction * 100 : 0;


// ■■■■■■■

int totalAll = totalAddition + totalSubtraction;

int correctAll = correctAddition + correctSubtraction;

float overallAccuracy = totalAll > 0 ? (float)correctAll / totalAll * 100 : 0;


// ■■UI■■

if (additionAccuracyText != null)

{

additionAccuracyText.text = $"{additionAccuracy}%";//({correctAddition}/{totalAddition})";

}


if (subtractionAccuracyText != null)

{

subtractionAccuracyText.text = $"{subtractionAccuracy}%";// ({correctSubtraction}/{totalSubtraction})";

}


if (overallAccuracyText != null)

{

overallAccuracyText.text = $"{overallAccuracy}%";// ({correctAll}/{totalAll})";

}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_chengjiu.cs

details.text = $"■■■:{totalAddition+totalSubtraction},■■:{correctAddition+correctSubtraction}";

mainall.text = $"{overallAccuracy}%";

Debug.Log($"■■■■■■ - ■■: {additionAccuracy:F1}%, ■■: {subtractionAccuracy:F1}%, ■■: {overallAccu

}

protected override void OnShow()

{

base.OnShow();

ShowStatistics();

}

/// <summary>

/// ■■■■■■

/// </summary>

void ShowStatistics()

{

UpdateStatisticsDisplay();

}

private void Awake()

{

var x = content.GetComponentsInChildren<Image>();

foreach(var item in x)

{

item.raycastTarget = false;

}

var fx = content.GetComponentsInChildren<TextMeshProUGUI>();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_chengjiu.cs

```
foreach (var item in fx)
{
item.raycastTarget = false;
}
Main.RegistEvent("event_chengjiu", (object parm) =>
{
show();
return null;
});
Main.RegistEvent("gamebegin", (object parm) =>
{
hide();
return null;
});
Main.RegistEvent("event_mix", (object parm) =>
{
hide();
return null;
});
}
// Start is called before the first frame update
void Start()
{
var ls =content.GetComponentsInChildren<VerticalLayoutGroup>();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_chengjiu.cs

```
for(int i = 0; i < ls.Length; i++)
{
ls[i].enabled = false;
}
}

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

}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_main.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class frm_main : frmbase
{
public Button plus, subs, mix, setup;
private void Awake()
{
Main.RegistEvent("gamebegin", (object parm) =>
{
show();
return null;
});
plus.onClick.AddListener(() =>
{
Main.DispEvent("event_plus");
});
subs.onClick.AddListener(() =>
{
Main.DispEvent("event_subs");
});
mix.onClick.AddListener(() =>
{
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_main.cs

```
Main.DispEvent("event_mix");
});
setup.onClick.AddListener(() =>
{
Main.DispEvent("event_chengjiu");
});
Main.RegistEvent("event_mix", (x) => {
this.hide();
return null;
});
Main.RegistEvent("event_plus", (x) => {
this.hide();
return null;
});
Main.RegistEvent("event_subs", (x) => {
this.hide();
return null;
});
Main.RegistEvent("event_chengjiu", (x) => {
this.hide();
return null;
});
Main.RegistEvent("event_setup", (x) => {
this.hide();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_main.cs

```
return null;
});
Main.RegistEvent("event_back", (x) => {
this.show();
return null;
});

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

}

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

}
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
■using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;
using UnityEngine.UI;
using DG.Tweening;

public class frm_practice : frmbase
{
public TextMeshProUGUI title, ansure, progress, explan;
public Button[] buttons;/*■■0■8■■■1■9■■■10■0■■■11■■■■■■■12■■■*/
private int currentQuestion = 0; // ■■■■■■
private int totalQuestions = 10; // ■■■■
private int score = 0; // ■■■
private string currentAnswer = ""; // ■■■■
private int correctAnswer = 0; // ■■■■
private string questionText = ""; // ■■■■
public Image right, wrong;

public Button back, play;
private bool isProcessing = false; // ■■■■■■■■■■

private int totalAdditionQuestions = 0; // ■■■■■■
private int correctAdditionQuestions = 0; // ■■■■■■■
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

private int totalSubtractionQuestions = 0; // ■■■■■■

private int correctSubtractionQuestions = 0; // ■■■■■■■

private QuestionType currentQuestionType; // ■■■■■■

// ■■■■■■

public enum QuestionType

{

Addition,    // ■■

Subtraction, // ■■

//  Multiplication // ■■

Mix,//■■

}

QuestionType type_practice = QuestionType.Addition;


void Awake()

{

back.onClick.AddListener(() =>

{

Main.DispEvent("event_back");

hide();

});

play.onClick.AddListener(() =>

{

Main.DispEvent("event_play", new int[] { para, parb, correctAnswer });

hide();
```

```
});
InitializeButtons();

Main.RegistEvent("event_begin", (x) =>
{
BeginPractice();
return null;
});
Main.RegistEvent("event_backfromplay", (x) =>
{
show();
return null;
});
Main.RegistEvent("gamebegin", (object parm) =>
{
hide();
return null;
});
Main.RegistEvent("event_plus", (x) =>
{

type_practice = QuestionType.Addition;
BeginPractice();
this.show();
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
return null;
});
Main.RegistEvent("event_subs", (x) =>
{

type_practice = QuestionType.Subtraction;
BeginPractice();
this.show();
return null;
});
Main.RegistEvent("event_mix", (x) =>
{
type_practice = QuestionType.Mix;
BeginPractice();
this.show();
return null;
});
Main.RegistEvent("event_chengjiu", (x) =>
{
this.hide();
return null;
});
// ■■■■■■■
if (right != null) right.gameObject.SetActive(false);
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
if (wrong != null) wrong.gameObject.SetActive(false);

}


/// <summary>

/// ■■■■■■■

/// </summary>

void InitializeButtons()

{

// ■■■■ 1-9

for (int i = 0; i < 9; i++)

{

int num = i + 1;

buttons[i].onClick.AddListener(() => OnNumberButtonPressed(num.ToString()));

}


// ■■■■ 0

buttons[9].onClick.AddListener(() => OnNumberButtonPressed("0"));


// ■■■■

buttons[10].onClick.AddListener(OnBackspaceButtonPressed);


// ■■■■

buttons[11].onClick.AddListener(OnConfirmButtonPressed);

}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
/// <summary>
/// ■■■■
/// </summary>
void BeginPractice()
{
currentQuestion = 0;
score = 0;
ResetCurrentStatistics(); // ■■■■■■■■■■■
GenerateNextQuestion();
}

/// <summary>
/// ■■■■■
/// </summary>
void GenerateNextQuestion()
{
if (currentQuestion >= totalQuestions)
{
SaveStatistics();
// ■■■■
EndPractice();
return;
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
// ■■■■■■

HideFeedbackIcons();


// ■■■■■■■■

QuestionType type = type_practice;// (QuestionType)Random.Range(0, 2);


// ■■■■■■■■■■

switch (type)

{

case QuestionType.Addition:

currentQuestionType = QuestionType.Addition;

totalAdditionQuestions++;

GenerateAdditionQuestion();

break;

case QuestionType.Subtraction:

currentQuestionType = QuestionType.Subtraction;

totalSubtractionQuestions++;

GenerateSubtractionQuestion();

break;

case QuestionType.Mix:

if (Random.Range(0, 2) == 0)

{

currentQuestionType = QuestionType.Addition;
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

totalAdditionQuestions++;

GenerateAdditionQuestion();

}

else

{

currentQuestionType = QuestionType.Subtraction;

totalSubtractionQuestions++;

GenerateSubtractionQuestion();

}

break;

}


// ■■■■■■

title.text = $"■{currentQuestion + 1}■: {questionText} = ?";

currentAnswer = "";

isProcessing = false;

currentQuestion++;

}

int para, parb;

/// <summary>

/// ■■■■■■

/// </summary>

void GenerateAdditionQuestion()

{
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs
para = Random.Range(1, 5); // 1-20■■■■
parb = Random.Range(1, 5);
correctAnswer = para + parb;
questionText = $"{para} + {parb}";
}

/// <summary>
/// ■■■■■■
/// </summary>
void GenerateSubtractionQuestion()
{
para = Random.Range(1, 10); // 1-30■■■■
parb = Random.Range(1, para);  // ■■■■■■■
correctAnswer = para - parb;
questionText = $"{para} - {parb}";
}

/// <summary>
/// ■■■■■■
/// </summary>
void GenerateMultiplicationQuestion()
{
int a = Random.Range(1, 13); // 1-12■■■■
int b = Random.Range(1, 13);
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs
correctAnswer = a * b;
questionText = $"{a} × {b}";
}

/// <summary>
/// ■■■■■■■■
/// </summary>
/// <param name="number">■■■■■</param>
void OnNumberButtonPressed(string number)
{
if (isProcessing) return; // ■■■■■■■■■■■■■■

// ■■■■■■■■■3■■■
if (currentAnswer.Length < 3)
{
currentAnswer += number;
UpdateTitleDisplay();
}
}

/// <summary>
/// ■■■■■■■■
/// </summary>
void OnBackspaceButtonPressed()
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs
{
if (isProcessing) return; // ■■■■■■■■■■■■■■■

if (currentAnswer.Length > 0)
{
currentAnswer = currentAnswer.Substring(0, currentAnswer.Length - 1);
UpdateTitleDisplay();



}
}

/// <summary>
/// ■■■■■■■■
/// </summary>
void OnConfirmButtonPressed()
{
if (autopress != null)
{
StopCoroutine(autopress);
autopress = null;
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs
if (isProcessing || string.IsNullOrEmpty(currentAnswer)) return;


isProcessing = true;
int answer = int.Parse(currentAnswer);


// ■■■■■■■■
if (answer == correctAnswer)
{
ShowFeedback(true);
}
else
{
ShowFeedback(false);
}
}


/// <summary>
/// ■■■■■■
/// </summary>
/// <param name="isCorrect">■■■■■■</param>
void ShowFeedback(bool isCorrect)
{
// ■■■■■■■
if (isCorrect)
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
{
if (right != null)
{
right.gameObject.SetActive(true);
// ■■■■■■
right.transform.localScale = Vector3.zero;
right.transform.DOScale(Vector3.one, 0.3f).SetEase(Ease.OutBack);
}
}
else
{
if (wrong != null)
{
wrong.gameObject.SetActive(true);
// ■■■■■■
wrong.transform.localScale = Vector3.zero;
wrong.transform.DOScale(Vector3.one, 0.3f).SetEase(Ease.OutBack);
}
}

// ■■■■
if (isCorrect)
{
score += 10; // ■■10■
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

```
if (currentQuestionType == QuestionType.Addition)
{
correctAdditionQuestions++;
}
else if (currentQuestionType == QuestionType.Subtraction)
{
correctSubtractionQuestions++;
}
Debug.Log($"■■■■■■■■■: {score}");
explan.text = $"■■";
}
else
{
explan.text = $"■■,■■■■■: {correctAnswer}";
Debug.Log($"■■■■■■■■■■: {correctAnswer}");
}

progress.text = $"■■{score}/{100}({currentQuestion * 10}%)";

// 3■■■■■■■
DOVirtual.DelayedCall(1.0f, () =>
{
explan.text = $"■■■■";
ansure.text = "_";
GenerateNextQuestion();
});
}

void SaveStatistics()
{
// ■■■■■■■■
int savedTotalAddition = PlayerPrefs.GetInt("TotalAdditionQuestions", 0);
int savedCorrectAddition = PlayerPrefs.GetInt("CorrectAdditionQuestions", 0);
int savedTotalSubtraction = PlayerPrefs.GetInt("TotalSubtractionQuestions", 0);
int savedCorrectSubtraction = PlayerPrefs.GetInt("CorrectSubtractionQuestions", 0);

// ■■■■■■
savedTotalAddition += totalAdditionQuestions;
savedCorrectAddition += correctAdditionQuestions;
savedTotalSubtraction += totalSubtractionQuestions;
savedCorrectSubtraction += correctSubtractionQuestions;

// ■■■■■■■■
PlayerPrefs.SetInt("TotalAdditionQuestions", savedTotalAddition);
PlayerPrefs.SetInt("CorrectAdditionQuestions", savedCorrectAddition);
PlayerPrefs.SetInt("TotalSubtractionQuestions", savedTotalSubtraction);
PlayerPrefs.SetInt("CorrectSubtractionQuestions", savedCorrectSubtraction);
PlayerPrefs.Save();
```

```
}

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

Debug.Log($"■■■■■■■ - ■■: {correctAdditionQuestions}/{totalAdditionQuestions}, ■■: {correctSubtracti
}
/// <summary>
/// ■■■■■■■■■■■
/// </summary>
void ResetCurrentStatistics()
{
totalAdditionQuestions = 0;
correctAdditionQuestions = 0;
totalSubtractionQuestions = 0;
correctSubtractionQuestions = 0;
}

/// <summary>
/// ■■■■■■
/// </summary>
void HideFeedbackIcons()
{
if (right != null) right.gameObject.SetActive(false);
if (wrong != null) wrong.gameObject.SetActive(false);
}

/// <summary>
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

/// ■■■■■■

/// </summary>

void UpdateTitleDisplay()

{

//title.text = $"■{currentQuestion}■: {questionText} = {currentAnswer}";

if (currentAnswer.Length > 0)

{

ansure.text = $"{currentAnswer}";

}

else

{

ansure.text = "_";

}



if (currentAnswer == correctAnswer.ToString())

{

autopress = pressok();

StartCoroutine(autopress);

}

}

IEnumerator pressok()

{

yield return new WaitForSeconds(0.5f);
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_practice.cs

OnConfirmButtonPressed();

}

IEnumerator autopress = null;


/// <summary>

/// ■■■■

/// </summary>

void EndPractice()

{

HideFeedbackIcons();

progress.text = $"■■■■■■■■: {score}/100";

Debug.Log($"■■■■■■■■: {score}/100");

Main.DispEvent("event_over", new object[] { score, 30 });

// ■■■■■■■■■■■■■■■■■■■■■

}

}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_result.cs

■using System.Collections;

using System.Collections.Generic;

using TMPro;

using UnityEngine;

using UnityEngine.UI;

public class frm_result : frmbase

{

public Button back, again;

public TextMeshProUGUI source, time;

// Start is called before the first frame update

private void Start()

{

Main.RegistEvent("event_over", (x)=>

{

object[]par = x as object[];

source.text = "■■■" +(int) par[0];

time.text = $"■■■{(int)par[1]}■";

show();

return null;

});

back.onClick.AddListener(() =>

{

Main.DispEvent("gamebegin");

hide();
});
again.onClick.AddListener(() =>
{
Main.DispEvent("event_begin");
});

Main.RegistEvent("event_begin", (x) =>
{
hide();
return null;
});
}

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

}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_setup.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class frm_setup : frmbase
{
public Button reset;
// Start is called before the first frame update
void Start()
{
Main.RegistEvent("event_setup", (x) =>
{
show();
return null;
});
Main.RegistEvent("gamebegin", (x) =>
{
hide();
return null;
});
Main.RegistEvent("event_mix", (x) =>
{
hide();
return null;
});
Main.RegistEvent("event_chengjiu", (x) =>
{
hide();
return null;
});
reset.onClick.AddListener(() =>
{
PlayerPrefs.DeleteAll();
PlayerPrefs.Save();
});
}

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

}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_tools.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
public class frm_tools : frmbase
{
public Button main, practice, result, setup;
private Button _curr;

Button curr
{
get
{
return _curr;
}
set
{
if (_curr != null)
{
_curr.GetComponent<Image>().color = Color.white;
_curr.transform.Find("Image").GetComponent<Image>().color = Color.gray;
_curr.GetComponentInChildren<TextMeshProUGUI>().color = Color.gray;
}
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_tools.cs

```
_curr = value;
if(_curr != null)
{
_curr.GetComponent<Image>().color =new Color(0.1568f,0.6196f,0.9607f);
_curr.transform.Find("Image").GetComponent<Image>().color = Color.white;
_curr.GetComponentInChildren<TextMeshProUGUI>().color = Color.white;
}
}
}
private void Awake()
{
Main.RegistEvent("gamebegin", (object parm) =>
{
curr = main;
return null;
});
main.onClick.AddListener(() =>
{
curr = main;
Main.DispEvent("gamebegin");
});
practice.onClick.AddListener(() =>
{
curr = practice;
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_tools.cs

```
Main.DispEvent("event_mix");
});
result.onClick.AddListener(() =>
{
curr = result;
Main.DispEvent("event_chengjiu");
});
setup.onClick.AddListener(() => {
curr = setup;
Main.DispEvent("event_setup");
});
Main.RegistEvent("event_chengjiu", (x) =>
{
curr = result;
return null;
});
}
// Start is called before the first frame update
void Start()
{

}

// Update is called once per frame
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_tools.cs

```
void Update()
{


}
}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_yanshi.cs

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using TMPro;

public class frm_yanshi : frmbase

{

public playctrl ctrl;

public TextMeshProUGUI source;

public Button back;

// Start is called before the first frame update

private void Awake()

{

Main.RegistEvent("gamebegin", (object parm) =>

{

hide();

return null;

});

Main.RegistEvent("event_backfromplay", (object parm) =>

{

hide();

return null;

});

}
```

```
■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_yanshi.cs

void Start()

{

Main.RegistEvent("event_play", (x) =>

{




int[] pars = x as int[];

if (pars[2] == pars[0] + pars[1]) {

source.text =  pars[0] + " + " + pars[1] + " = " + pars[2];

ctrl.PlayOnePlusOne(pars[0],pars[1]);

}

else

{

source.text = pars[0] + " - " + pars[1] + " = " + pars[2];

ctrl.PlayOneSubOne(pars[0],pars[1]);

}

show();

return null;

});


back.onClick.AddListener(() =>

{


Main.DispEvent("event_backfromplay");
```

■■: c:\Users\Administrator\mathpractice\client\Assets\Scripts\frm\frm_yanshi.cs

```
});
}

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

}
}
```