**C语言项目报告**

**项目名称：XBeats音乐游戏**

**项目成员：刘彦麟、杨智博、刘淼**

**填写日期：2021年7月9日**

1. **摘要（Abstract）**

本游戏是一款采用Unity开发的电脑端下落式音游，在实现音游基本功能的基础上，为玩家带来了类似于移动端的游戏体验。游戏中有六条轨道，分别产生音符，玩家需要跟随音乐节奏，在恰当的时间以要求的方式按下对应的按键来击中音符，以尽量取得高分。

1. **问题描述（Problem Statement）**

从2005年风靡中国的端游“劲舞团”，到如今音乐游戏人尽皆知，大众或多或少都亲身体验过，音乐游戏已经成为众多游戏类型中用户广泛的一类游戏。音乐游戏是一种将节奏，画面，音乐进行有机结合的休闲游戏。玩家需要伴随音乐节奏在恰当的时间敲击音符，以尽量取得高分。在游玩的过程中，玩家能从其特有的节奏感，打击感和强烈的画面冲击感中获得乐趣，创造一种仿佛玩家是在演奏该音乐的感觉，也因为分数和评级的加入，使得玩家也可以在其中获得更高的成就感。

随着Phigros、Deemo、Cytus、Muse Dash、冰与火之舞等优秀音游的普及，越来越多的玩家感受到了音游的独特魅力。然而，由于智能手机及平板电脑的屏幕要远远灵活与电脑键盘，目前的音游主要集中在手机端。而且目前主流音乐游戏大都是2D下落式玩法，该项目意在开发一款PC端的下落式音游，填补PC端音游的空白，尽可能让玩家在电脑端也能感受到音游的打击感和前所未有的画面冲击感。

目前本游戏中有6首曲目可供游玩。随着游戏的后续开发，还会添加更多的曲目，从而进一步增强游戏的可玩性。

1. **组内分工（Group Division）**

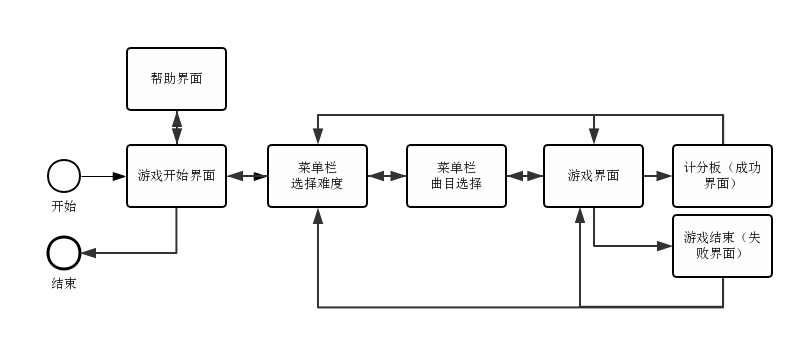
唐化-刘彦麟：游戏核心程序，音乐以及音效制作，谱面制作，计分板模块。 40%

唐化-杨智博：UI以及素材收集和绘制，特效，帮助模块，暂停开始及进度条模块。 35%

唐化-刘淼：菜单栏设计，计分规则设计，谱面校对，部署测试。 25%

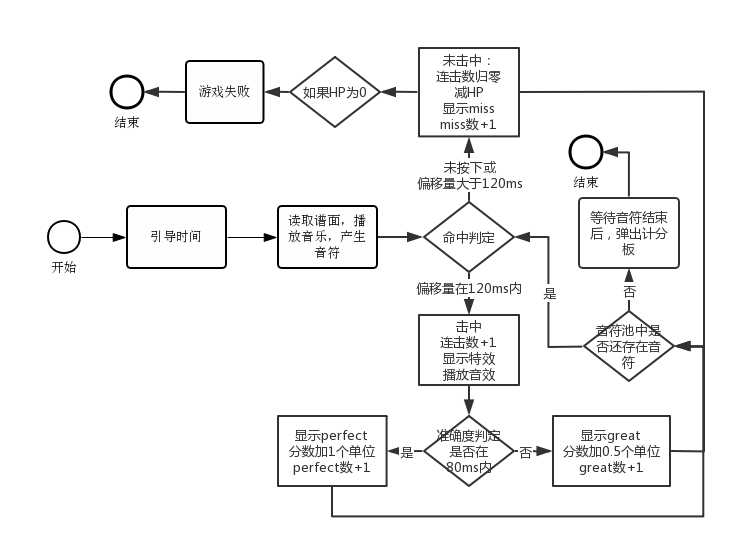
1. **分析（Analysis）**

游戏整体的进行逻辑如下：



1. **设计（Design）**

各功能模块主要集中于游戏界面。现简述游戏过程中各功能的设计：



计分规则：

1单位分数=1000000/音符总数（取整数）。

每个音符对应1单位分数。若为miss，不得分；若为great，得0.5单

位分数；若为perfect，得1单位分数（为避免全部perfect时得分不为 1000000，此时将直接赋分1000000）。

评级：总分处于规定的区间，将对应相应的评级，并在计分板中显示。

0~599999——D

600000~699999——C

700000~799999——B

800000~899999——A

900000~949999——S

950000~999999——SS

1000000——SSS

1. **实施（Implementation）**

现列举部分核心模块函数的具体实现代码：

//从池中取对象的方法

public NoteObject GetFreshNoteObject()

{

NoteObject retObj;

if(noteObjectPool.Count>0)

{

retObj = noteObjectPool.Pop();

}

else

{

retObj = Instantiate(noteObject);

}

retObj.transform.position = Vector3.one \* 2;

retObj.gameObject.SetActive(true);

retObj.enabled = true;

return retObj;

}

//将音符对象放回对象池

public void ReturnNoteObjectToPool(NoteObject obj)

{

if(obj!=null)

{

obj.enabled = false;

obj.gameObject.SetActive(false);

noteObjectPool.Push(obj);

}

}

//音符在谱面上产生的位置偏移量

int GetSpawnSampleOffset()

{

//出生位置与目标点的位置

float spawnDistToTarget = targetTopTrans.position.z - transform.position.z;

//到达目标点的时间

float spawnPosToTargetTime = spawnDistToTarget / gameController.noteSpeed;

return (int)spawnPosToTargetTime \* gameController.SampleRate;

}

//检查是否生成下一个新音符

void CheckSpawnNext()

{

int samplesToTarget = GetSpawnSampleOffset();

int currentTime = gameController.DelayedSampleTime;

while (pendingEventIdx < laneEvents.Count && laneEvents[pendingEventIdx].StartSample < currentTime + samplesToTarget)

{

KoreographyEvent evt = laneEvents[pendingEventIdx];

int noteNum = evt.GetIntValue();

NoteObject newObj = gameController.GetFreshNoteObject();

isLongNoteStart = false;

isLongNoteEnd = false;

if(noteNum>6)

{

isLongNoteStart = true;

noteNum = noteNum - 6;

if(noteNum>6)

{

isLongNoteEnd = true;

isLongNoteStart = false;

noteNum = noteNum - 6;

}

}

newObj.Initialize(evt, noteNum, this, gameController, isLongNoteStart, isLongNoteEnd);

trackedNotes.Enqueue(newObj);

pendingEventIdx++;

}

}

//检测是否有击中音符对象

//如果是，它将执行命中并删除

public void CheckNoteHit()

{

if (!gameController.gameStart)

{

CreateDownEffect();

return;

}

if(trackedNotes.Count>0)

{

NoteObject noteObject = trackedNotes.Peek();

if(noteObject.hitOffset>-6000)

{

trackedNotes.Dequeue();

int hitLevel = noteObject.IsNoteHittable();

if (hitLevel == 2)

{

gameController.perfectNum++;

}

else if (hitLevel == 1)

{

gameController.goodNum++;

}

if (hitLevel>0)

{

//增加连击数

gameController.comboNum++;

if (gameController.comboNum > gameController.maxCombo)

{

gameController.maxCombo = gameController.comboNum;

}

gameController.ChangeHitLevelSprite(hitLevel);

//更新分数

gameController.UpdateScoreText(0.5 \* hitLevel \* (1000000 / gameController.notesInTotal));

if (noteObject.isLongNote)

{

hasLongNote = true;

gameController.hitMusic.clip = gameController.hitSound;

gameController.hitMusic.Play();

CreateHitLongEffect();

}

else if(noteObject.isLongNoteEnd)

{

hasLongNote = false;

}

else

{

gameController.hitMusic.clip = gameController.hitSound;

gameController.hitMusic.Play();

CreateHitEffect();

}

}

else

{

gameController.ChangeHitLevelSprite(hitLevel);

//未击中

//减HP

gameController.UpdateHP();

//断连

gameController.HideComboNumText();

gameController.comboNum = 0;

}

noteObject.OnHit();

}

else

{

CreateDownEffect();

}

}

else

{

CreateDownEffect();

}

}

//更新音符位置的方法

public void UpdatePosition()

{

Vector3 pos = laneController.TargetPosition;

pos.z -= (gameController.DelayedSampleTime - trackedEvents.StartSample) / (float)gameController.SampleRate \* gameController.noteSpeed;

transform.position = pos;

}

1. **测试（Test）**

以下截取了游戏过程中的一些画面：



图1 Start界面

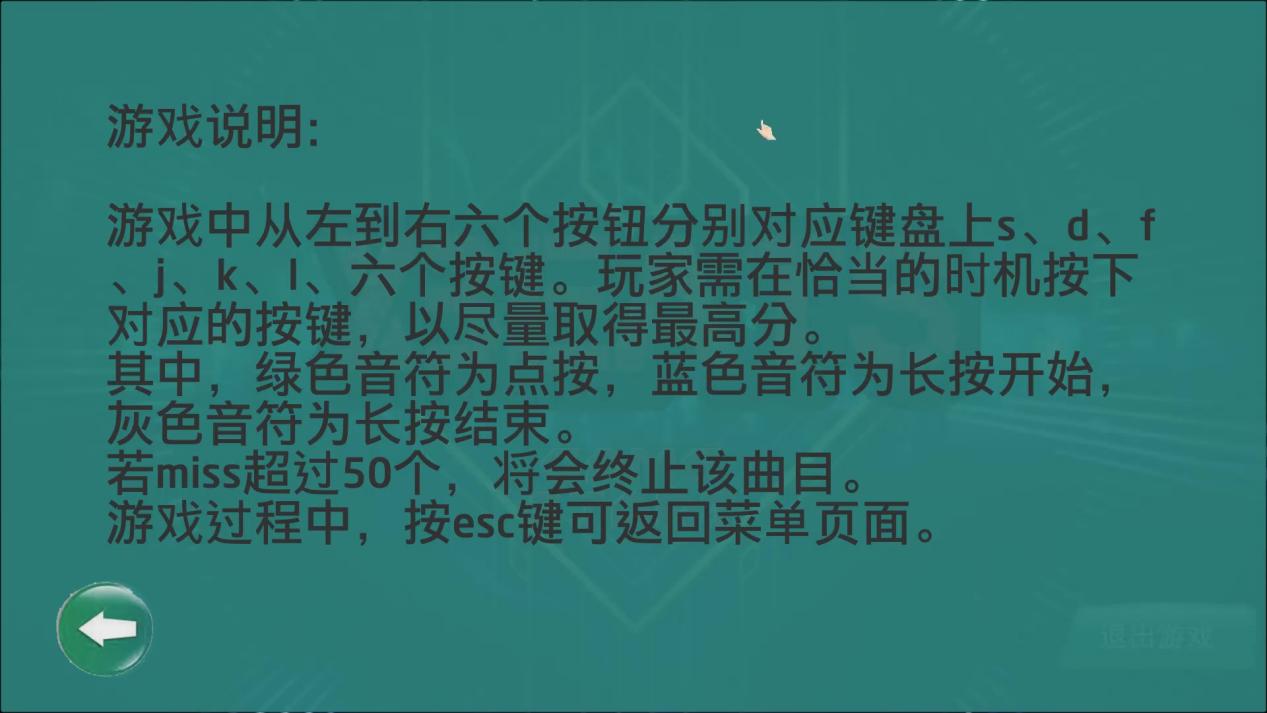


图2 帮助界面



图3 Menu界面——难度选择

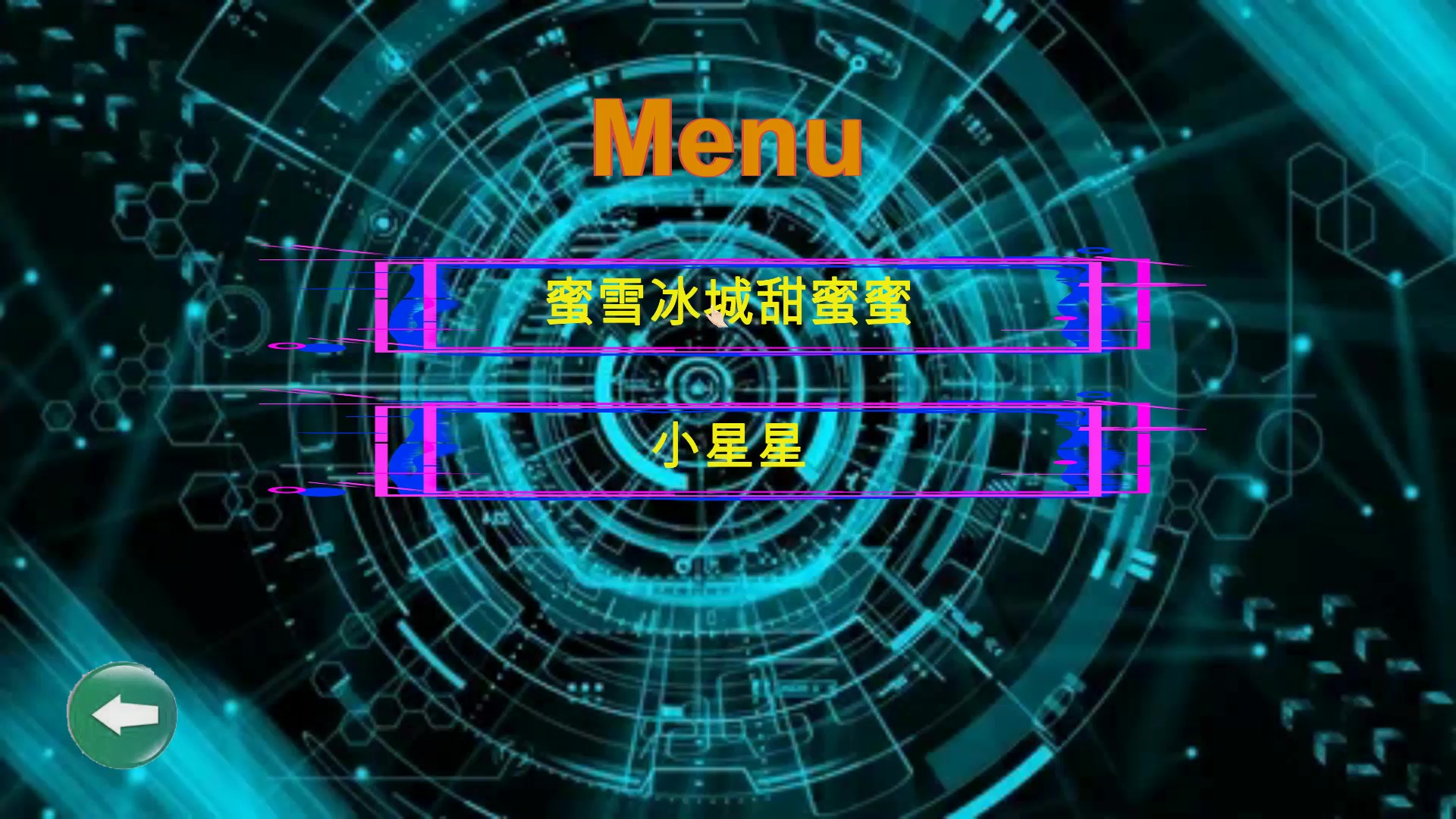


图4 Menu界面——曲目选择



图5曲目——Remember Our Summer



图6 曲目——二十三号病床



图7 曲目——Vindication



图8 计分板