

# Arcaea Assembly

## 1. 简介

用masm32和win32编程实现一款简易的下落式音游。

## 2. 开发环境

- IDE: VS Code
- Assembler: MASM32
- OS: Win10

## 3. 运行方法

- vscode工作区配置

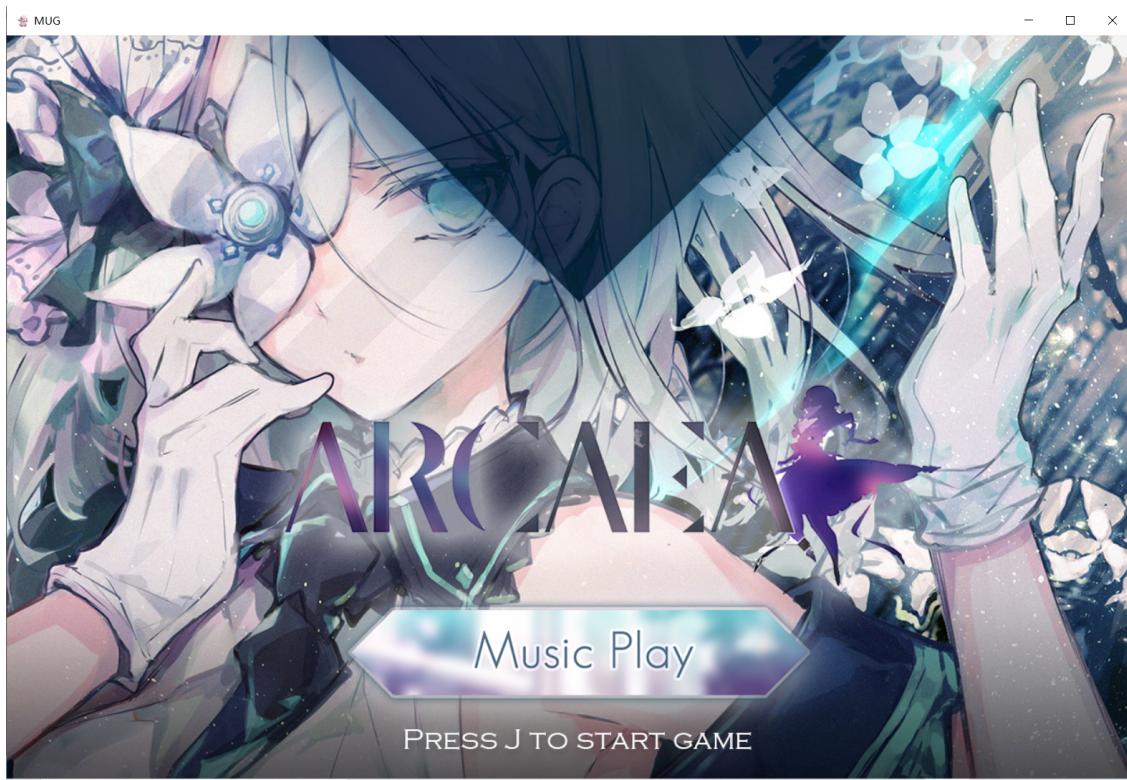
```
1  {
2      "cmake.configureArgs": ["-DMASM_HOME=C:/masm32"],
3      "C_Cpp.default.configurationProvider": "ms-vscode.cmake.tools",
4      "files.associations": {
5          "charconv": "cpp"
6      },
7      "debug.allowBreakpointsEverywhere": true
8  }
```

其中 `cmake.configureArgs` 换成自己的masm32目录。

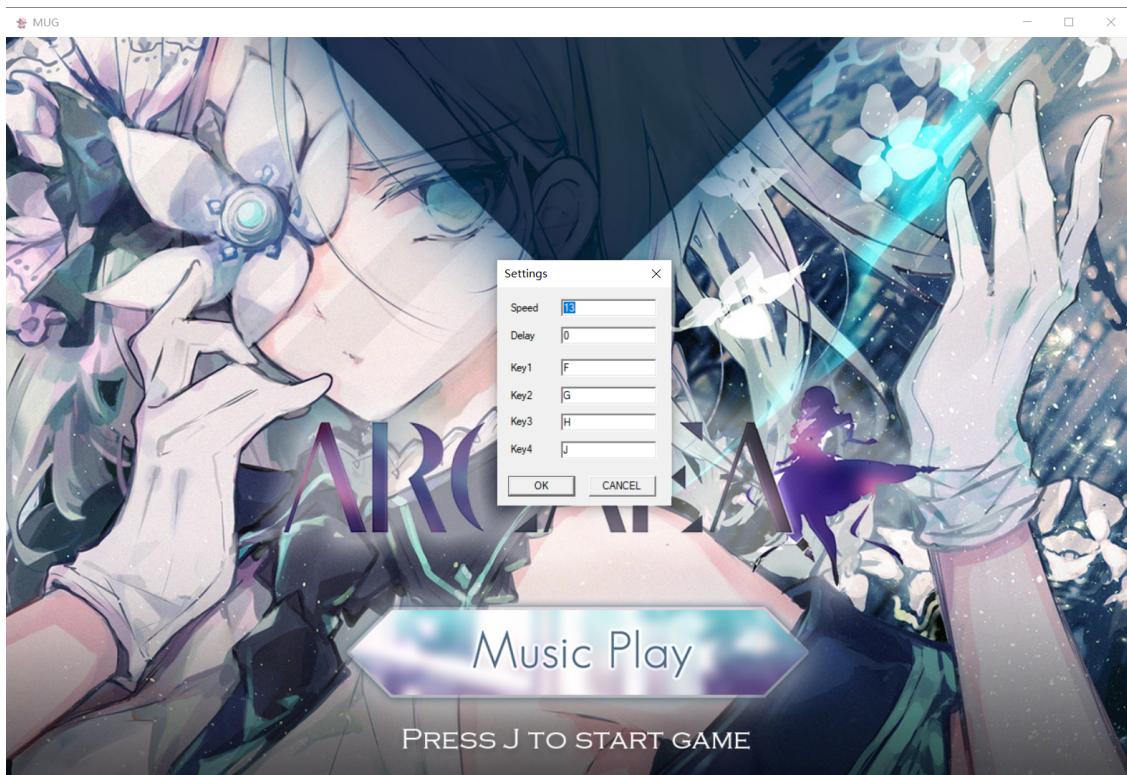
- 运行 `.\level-build` 目录中的 `cyaegha.cpp` 和 `sheriruth.cpp` 文件, 在 `.\bin\Debug` 目录中生成 `cyaegha.exe` 和 `sheriruth.exe` (分别选择build target为 `cyaegha` 和 `sheriruth`)。
- 在 `.\bin\Debug` 运行 `cyaegha.exe` 和 `sheriruth.exe`, 在 `.\bin\Debug\levels` 目录下生成两个二进制谱面文件 `cyaegha.level` 和 `sheriruth.level`。
- 选择build target为 `asm_mug`, 编译链接项目文件和资源文件, 在 `.\bin\Debug` 目录下生成 `asm_mug.exe` 可执行文件。

## 4. 游戏说明

- 开始页面:



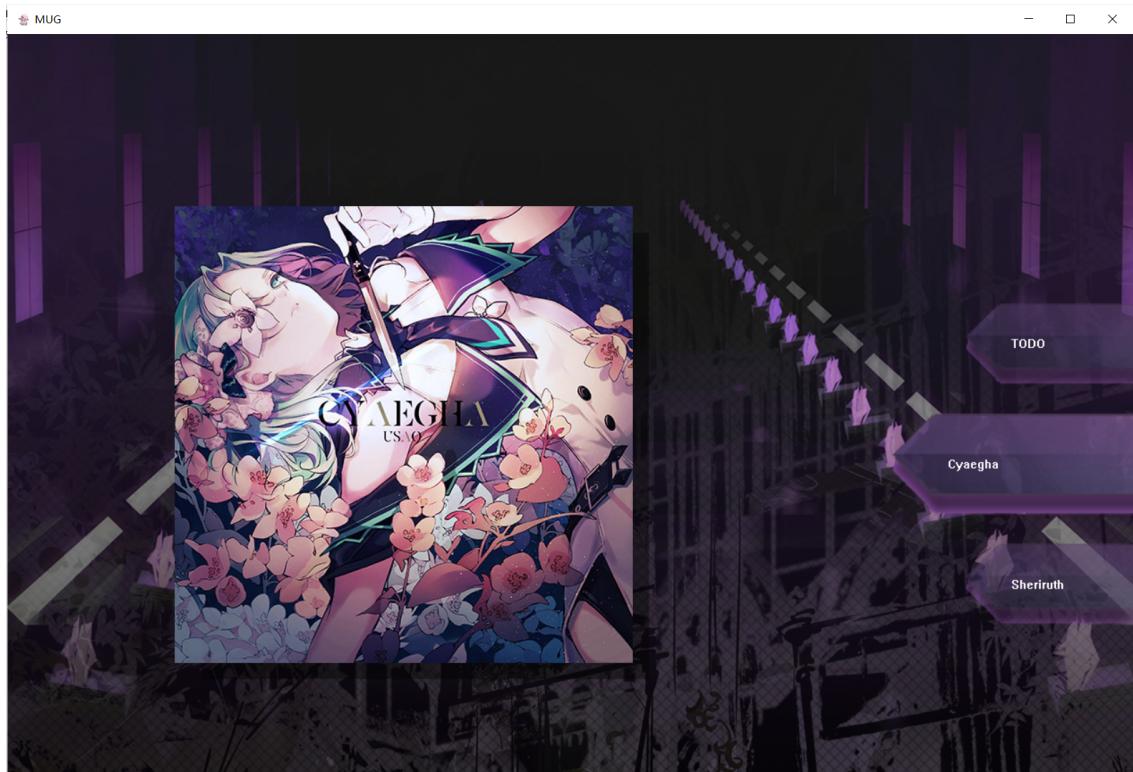
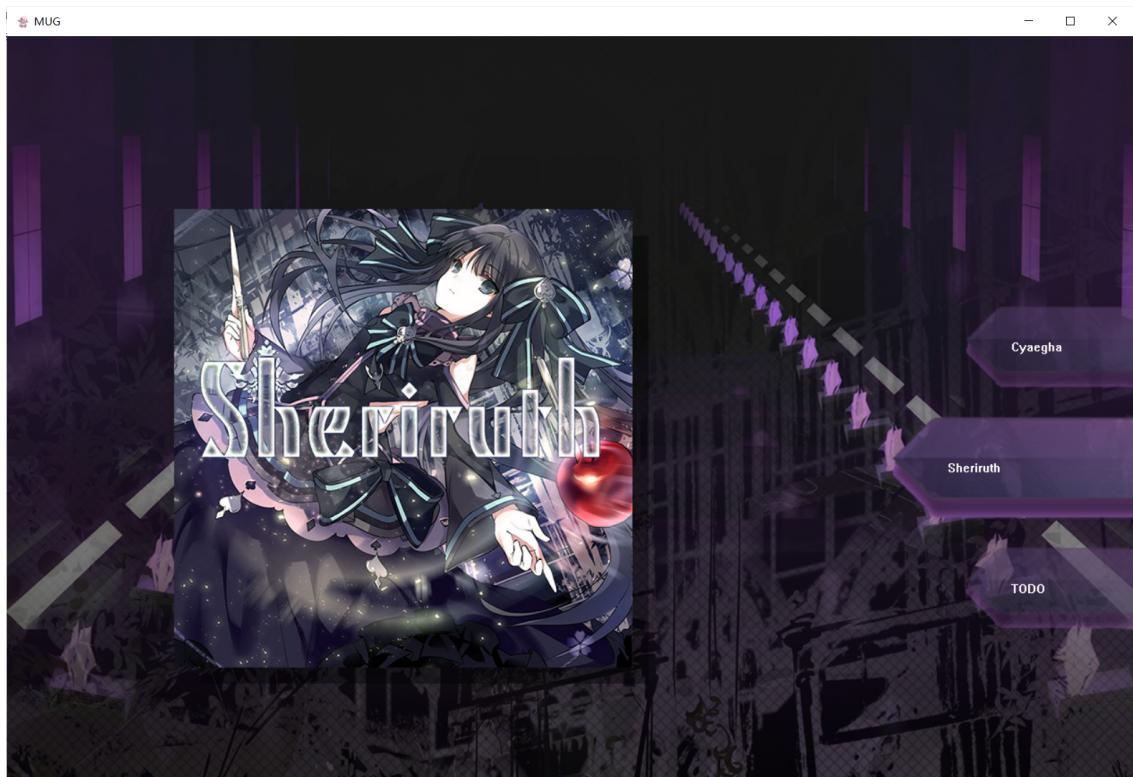
如图所示，按J键开始游戏，进入选歌页面；按H键打开设置窗口。



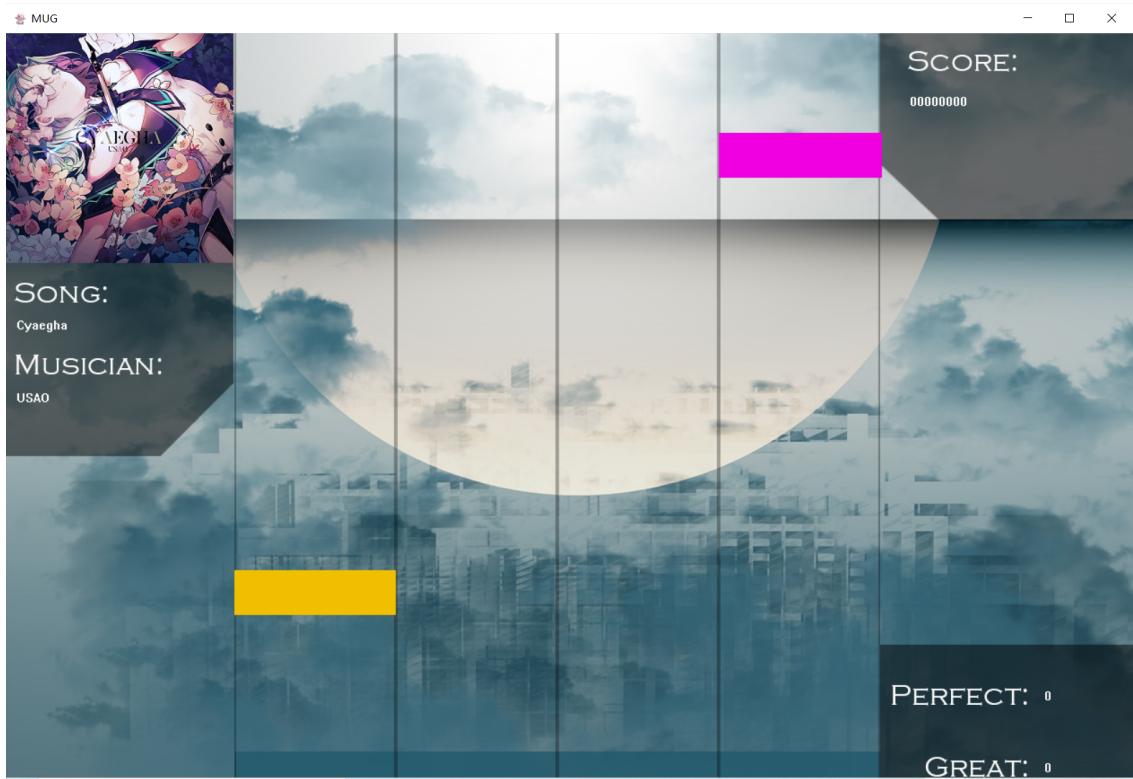
设置窗口可设置速度、延迟和按键映射，如图为默认设置。

- **选歌页面：**

选歌页面功能画面左侧为当前选中曲目封面图片，滚动鼠标滚轮可切换选中的歌曲。曲目名称在画面右侧显示，其中被最大文本框圈中的歌曲为被选中歌曲。在此页面按H键可开始游玩选中曲目，按Esc回到开始页面。



- 游戏页面：



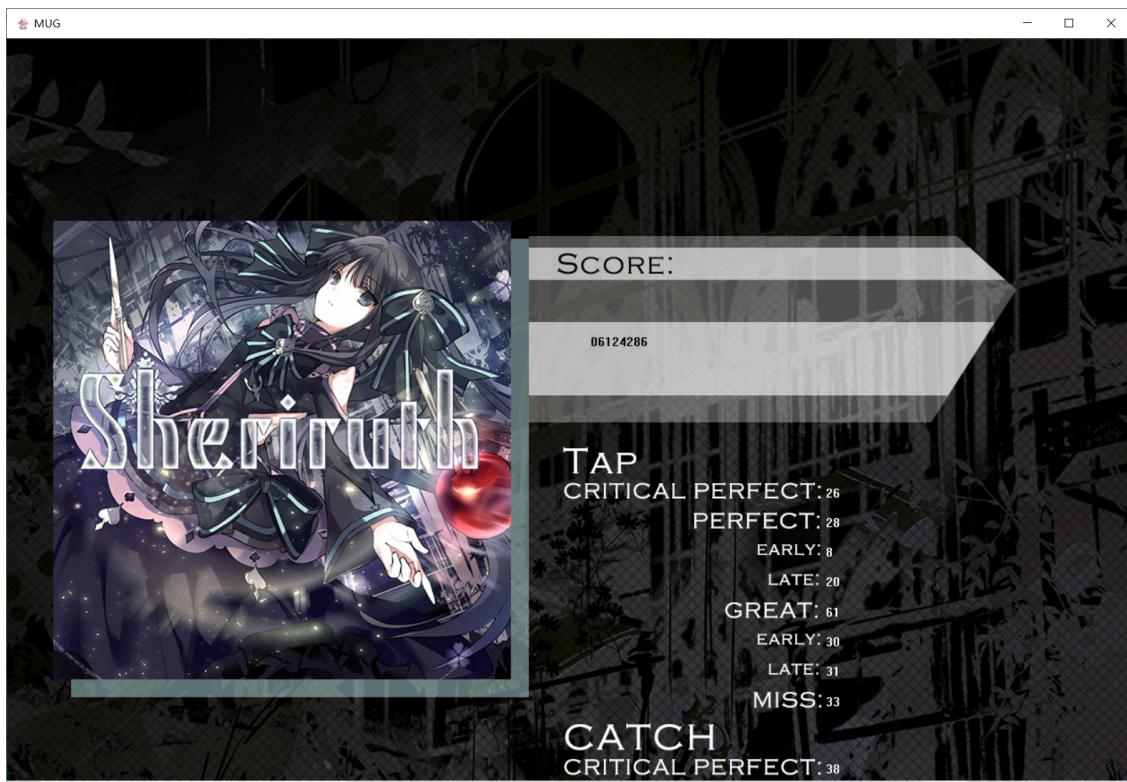
左上角为曲目封面，下方有曲目名称和作者；右上角为当前得分；右下角为当前记录的perfect、great和miss数。

下方深色矩形的上边缘为判定线。

游戏中下落方块分为两种：tap方块（紫色）和catch方块（黄色）。tap方块要求玩家在方块落到判定线附近时敲击对应轨道按键，游戏会根据玩家敲击按键的时机判断本次动作评价，具体可分为：critical perfect、perfect、great、miss，判定依据为敲击按键时方块重心距判定线的绝对距离，距离越短评价越高，评价越高得分越高。catch方块要求玩家在块落到判定线附近时对应轨道按键为按下状态，这类方块的评价只有critical perfect和miss两种，判定依据为方块在判定线附近时是否检测到按键被按下，评价越高得分越高。

不同轨道对应的按键可在设置页面中修改，在游玩页面按Esc可结束当前游戏，跳转到结算页面。

- **结算页面：**



结算页面显示本次游玩的成绩。画面左侧为游玩曲目封面，右侧显示游玩分数信息以及获得各种评价的方块数量。需要说明的是：PERFECT标签后两行有较小字体的EARLY标签和LATE标签，分别指示了因过早敲击按键而得到PERFECT评价的方块数量以及因过晚敲击按键而得到PERFECT评价的方块数量，二者的和应等于PERFECT标签后的数据。GREAT标签同理。

在结算页面按Esc可回到选歌页面。

## 5. 实现思路

- 首先用C++写出绘制谱面的工具 `level_builder.hpp` 和游戏的大体框架。
- 然后根据B站歌曲的节奏谱用谱面绘制工具写入歌曲信息 (e.g. name, author, musicPath, imagePath etc.) 和 BPM(Beats Per Minute)

以Cyaegha为例，该音乐乐谱的拍号为4/4，按照以下格式扩充谱面：

```

1 auto lb = LevelBuilder(info, 200.0f);
2 sec_1:
3     rep(4) lb.Add(4, {2, '-'});
4 sec_2:
5     lb.Add(4, {2, 'o'}, {1, '-'});
6     rep(3) lb.Add(4, {2, '-'});
7 ...

```

- 该乐曲的信息是info, bpm为200
- 第一小节重复4次，在第二轨道有一个Catch类型的四分音符，同时会根据bpm算出音符的时值
- 第二小节第一个四分音符在第二轨道有一个Tap类型的四分音符，然后在第一轨道有一个Catch类型的四分音符，之后在第二轨道有一个时值为三个四分音符的Catch类型的音符。

写完一首歌的谱面之后，将以二进制文件的形式写入 `./bin/Debug/levels` 目录，避免了解码等不必要的麻烦。

- 使用masm32和win32 api 实现用C++编写的游戏框架并填充细节。
- 游玩界面汇编从相应目录根据选歌页面选择的歌曲读取相应的谱面文件，给定一个固定的帧率绘制下落块

- 判定思路：按键按下时，记录此时的瞬间时间，并且进行tap的判定，找到第一个非miss判定，过早的话则跳过；按键松开时和每帧都要进行catch的判定，每帧也都要进行tap miss的判定。

## 6. 难点和创新点

---

### 6.1 难点

- 我们实现的音游是一个综合性的项目，涉及到文件读写、文本编辑、音乐播放、鼠标键盘事件响应、图形绘制等各个方面。
- 音游需要谱面、音乐、按键三者都达到一个相互匹配的状态
- 游戏按键判定方面，需要实现非常灵敏的键盘交互（GameUpdateJudgement等函数）
- 绘制图像和文字方面，需要做到位置的精准与数据的实时更新（GameDraw等函数）

### 6.2. 创新点

- 我们在写代码前对游戏架构进行了充分的讨论和设计，使用单例模式管理游戏相关的资源。最终的代码实现没有任何硬编码，具有可扩展性。
- 我们设计好了一套自动化的谱面生成工具，实现了从谱面到二进制文件的端到端的编写，用户可以方便地根据自己的喜好自定义新谱面。
- 游戏记录了每个判定（e.g. perfect 和 great）是因为过早还是过晚按键得到的，方便玩家根据反馈改进，提升自己的水平。
- 我们允许用户根据自己的喜好调整下落速度（1 ~ 16）、判定延迟和按键映射。判定延迟功能增加了适配用户系统的能力，用户可以根据系统延迟调整。
- 增加了按键的打击音效，给用户提供了沉浸式的体验。
- 我们给按键增加了打击特效，用汇编实现了简易的动画，给用户提供了沉浸式的体验。