**MAGIC PIANO--C Language Report**

Group 13

In this report, we will introduce our project from four aspects.

**1. Problem statement**

Totally, MAGIC PIANO(魔幻钢琴) is a music game, with an inspiration from two series of music software, *Piano Master*(钢琴大师)—to listen, play and play music by scores, and *FL*(水果音乐制作软件)—to create music as representatives. We have combined them together, and add the function of set Speed for creating music as innovation.

To be specific, MAGIC PIANO has three areas, including Created music area(原创曲目区), where you can create music and then listen to it, Classic music area(经典曲目区), where you can listen to music, and also can play music by scores, and finally, Playing music area(自由弹奏区), where you can play freely by keyboard.

**2. Analysis**

Problem1: save the created music

Analysis1: use knowledge of files to save the created music

Problem2: manage the music

Analysis2: use a link list to save names of music and to manage them, such as travelling the link list to search for a piece of music

Problem3: how to play the music after creating music files

Analysis3: make an algorithm to get voice height of sound and send it to the Beep function to make a sound

Problem4: scorePlay function

Analysis4: make an algorithm to let users get scores when playing with the beats of music

Problem5: interface

Analysis5: use easyX graphics library and the functions in that to make the interface

**3. Game design**

The chart shows from the users’ perspective, what the game should be, and how we should design the areas. What’s more, we are supposed to make the interface by easyX graphics library.

Save the information into files and play it

created

listen

classic

playing

Show introduction of the game

introduction

start

score

entry

Match the keys with notes

Search for the music and scorePlay()

Search for the music and play it

with information shown

**4. Implementation**

**4.1 total condition of project**

The project has about 1,700 rows of codes.

Listen area includes 7 pieces of music, and Score part includes 3 pieces of music.

Interface is made by easyX graphics library.

**4.2 Division of work**

*Xiong Yueqi* made the main body of functions and interface.

*Zhuang Xu* made the music list and the management of them by link list.

*Du Yunlong* collected the music and made the beat arrays for scorePlay part.

**4.3 Specific details of the project**

Following are documents in our project.

main.cpp: the main function and includes several functions(begin(), login(), select(), introduction())to paint some interface at the beginning of game.

create.h: create music, save it into file and play it, includes write(), to write the information, Run(), to play the music.

classical.h: classic music area, includes Listen(), to listen to and show the information of music, Score(), to play the music with scores and calls the function scorePlay(), and several functions to paint the interfaces.

scorePlay(): an algorithm to play music by scores.

play.h: match the notes(音符)to the key on the keyboard, and includes the function to show some tips when playing freely.

music.h: musicList() to save the information of pieces, create\_music\_list() to create the link list, and destory\_list() to destroy the list at the end of the game.

Function declaration: some declaration of functions, which are called several times in documents.

And let me show you three main algorithms we developed as representatives.

They are making Music list, Get\_Voice\_Height(), a function to get voice height(声音频率), which is used to calculate voice height of each note and send it to Beep function to make a sound, and scorePlay(), a function to let you play music by getting scores in the score part in classic music area(*Piano master* can be used as a reference).

Then let me tell you the main structure of Get\_Voice\_Height() and scorePlay().

**Start**

Get three arguments

Get

*voiceListIndex*

Edit the *voiceListIndex*

Get the *voiceHeight*

Edit the *voiceHeight*

**Return**

In function Get\_Voice\_Height(), the function gets three arguments from the function create()-- Voice level(音阶，如0--7), Voice Flag(升降调，#升半调/b降半调/-不升不降), and dimension(八度，这里我们的设置有七个八度1--7). Then we use Voice level to get *voiceListIndex*, an index for each note to calculate later. Then we use Voice Flag to edit this index, and get the voiceHeight(声音频率雏形)from the voiceHeightList(频率对照表，是把index和实际频率一一对应的表，是预设好的). Finally we edit the voiceHeight by the third argument, dimension. After this, the function returns the voiceHeight. (这个声音频率后来传递给Beep函数令其发声).

scorePlay() is the most important function in our game, which supports the part of playing by scores. Following is the flow chart.

track()

If input Esc?

**No**

**Start**

static\_game()

update()

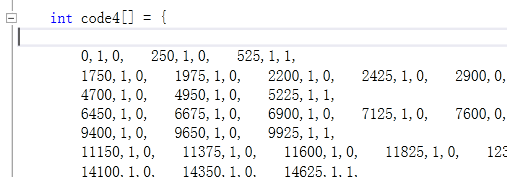
**Finish**

**Yes**

In this function scorePlay(), at first it’s the function static game()(初始化游戏界面函数), and update the users’ input then. After that we judge whether the input value is Esc key: if yes, finish the function, if no, that is the track() function. Then the function prosesses the next input. The track() worth statement.

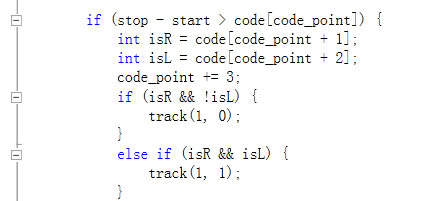
First, in Classical.h.

When we set the diamond blocks for the music, we use an array to set the blocks to match the beat of the music.



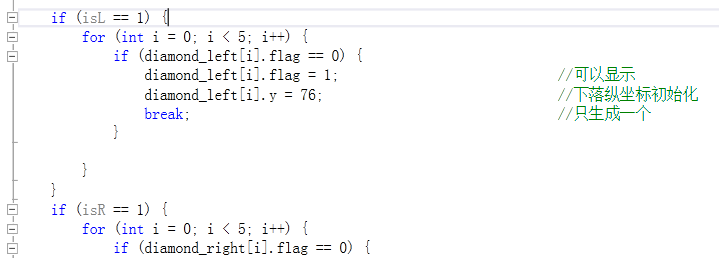
Each three numbers mean a diamond block(水晶条), means the time, such as 250 milliseconds, the right track, and the left track. 1 means yes and 0 means no.

Then in following picture, the function reads the numbers in the array, and sends the information to *Track()* function.

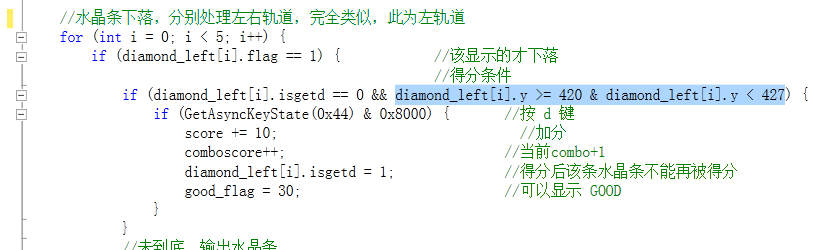


The two arguments mean whether show a block on the right track, and so does the left.

Then in scorePlay.h.



Here decide whether to show the diamond block.



And then here lets the block fall down.

If we click at the right beat, this means we click when the block falls at the bottom, the ordinate is 420—427, the score will plus 10, the combo will plus 1, and the GOOD picture will appear.

**4.4 Summary of the project**

Our project’s objected grade is 94—96, and from our perspective, we have finished the project well and we are satisfied with this. We have also learnt more knowledge about collaboration during this progress.