VG101 Project Design Report

Project Group 30

1. Group Number: 30

2. Members:

Name	Name in Hanzi	Student ID
Binhao Qin	秦彬皓	519370910092
Jingbo Qu	瞿敬柏	519370910091
Wenqi Cao	曹文琦	519370910111

3. Project Name: C++-based Rhythm Game

4. Language Used: C++

5. Project Summary

This project aims to make a rhythm game by C++ and OpenGL, with a User Interface provided.

6. Motivation

Music games can help players relax and increase their sense of music. Developing a simple music game is a challenge as well as a meaningful task for us. We can learn useful concepts and applications of C++ that may not be learned in class while developing. (e.g. class, virtual function, JSON file, OpenGL, etc.)

7. Design

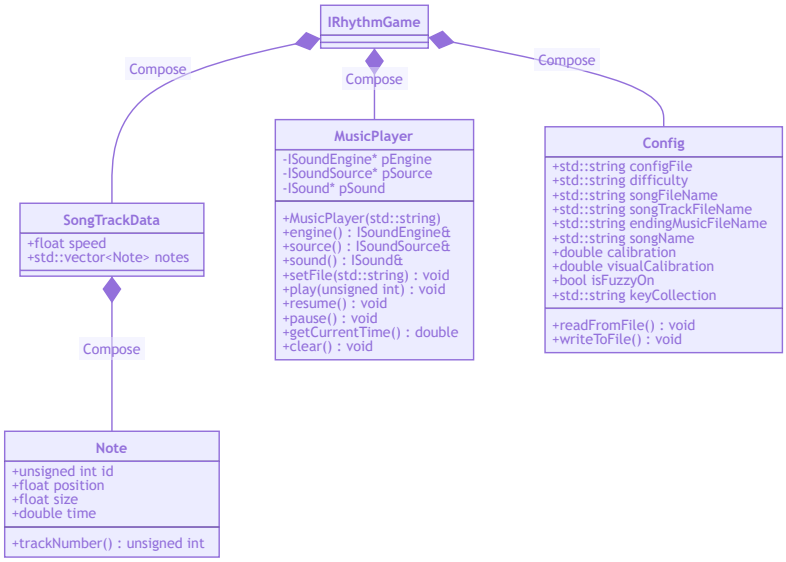
To efficiently develop a C++ project, we break the project into three parts following the MVC logic: Model, View and Control. Each of our member is in charge of a part. We also follow the procedure of Object-Oriented Programing through API. Each of us asks other members for demand of functions inside our own classes, and determine a purely virtual class (interface in C++) and let our own implementation inherit these virtual classes. This ensures the stability of API and change of implementation of one part doesn't interfere with another.

7.1 Features

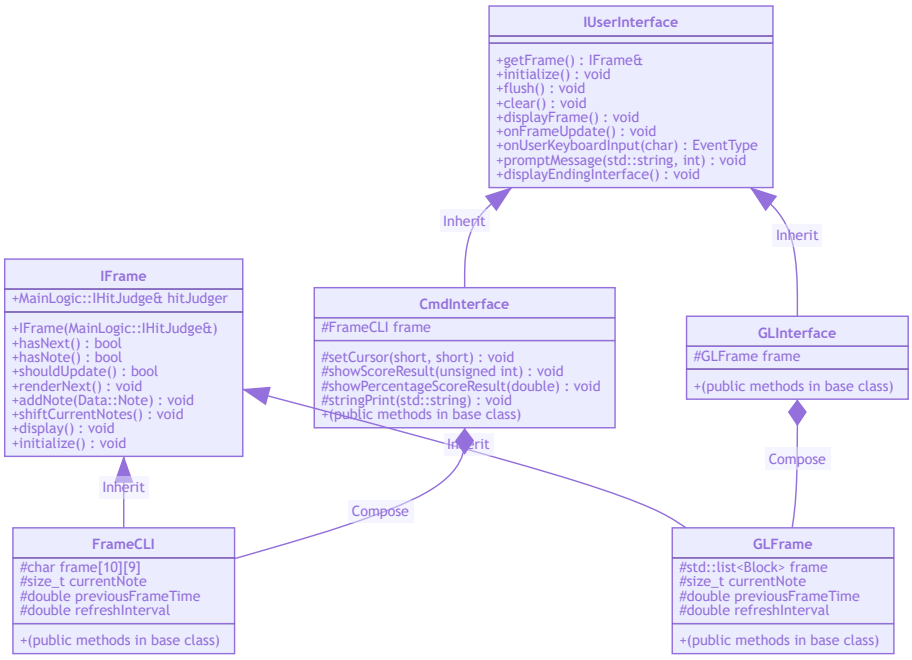
- You can change the difficulty level and select the song by modifying the `config.json` file.
- You can do Audio and Visual Calibration for better gaming experience
- You can change the key mapping by changing the value of `"keyMap"` in your `config.json`.
- You can Pause, Resume, Replay and Exit at anytime you want.
- The final result of the game will be showed when the game ends

7.2 Data Dictionary or Datagram

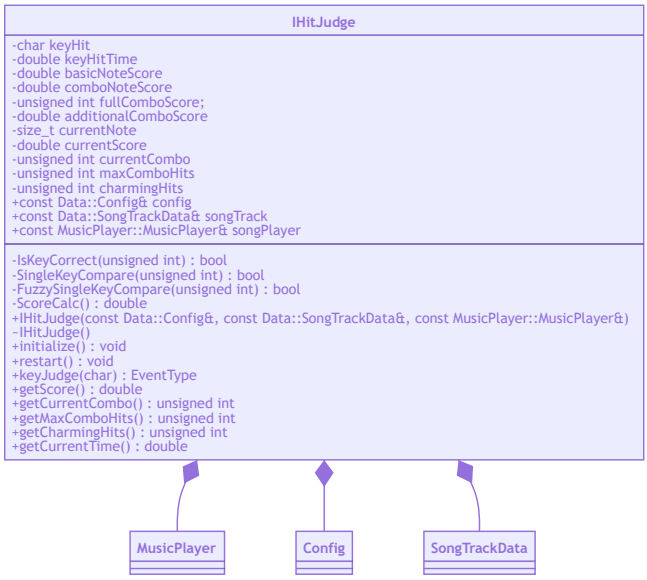
Model, Music and Data:



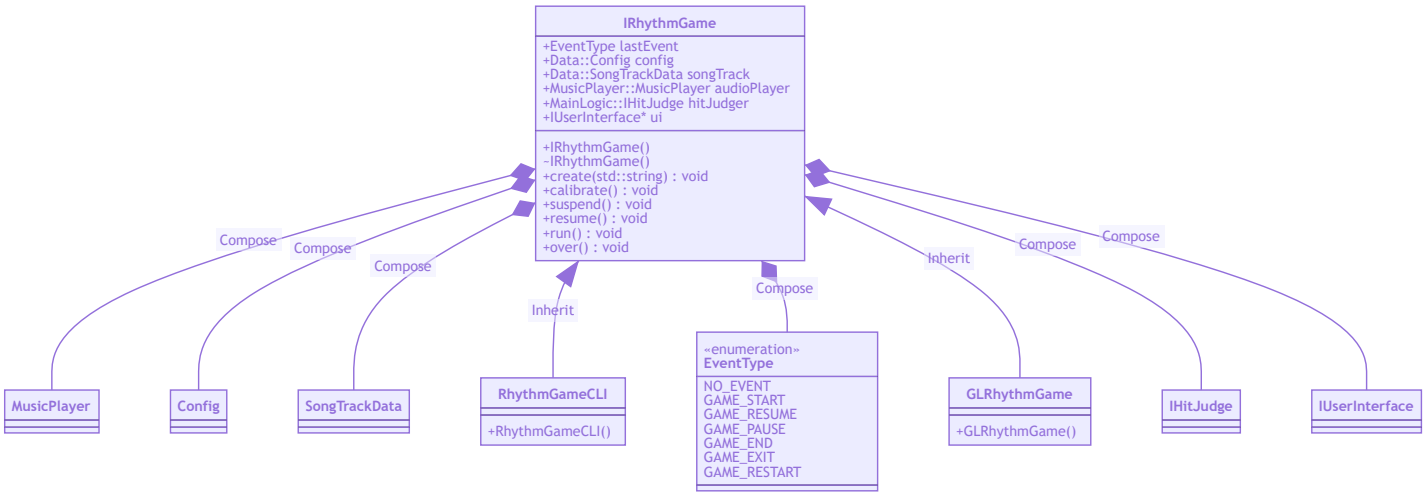
View:



Game Logic



Control Logic

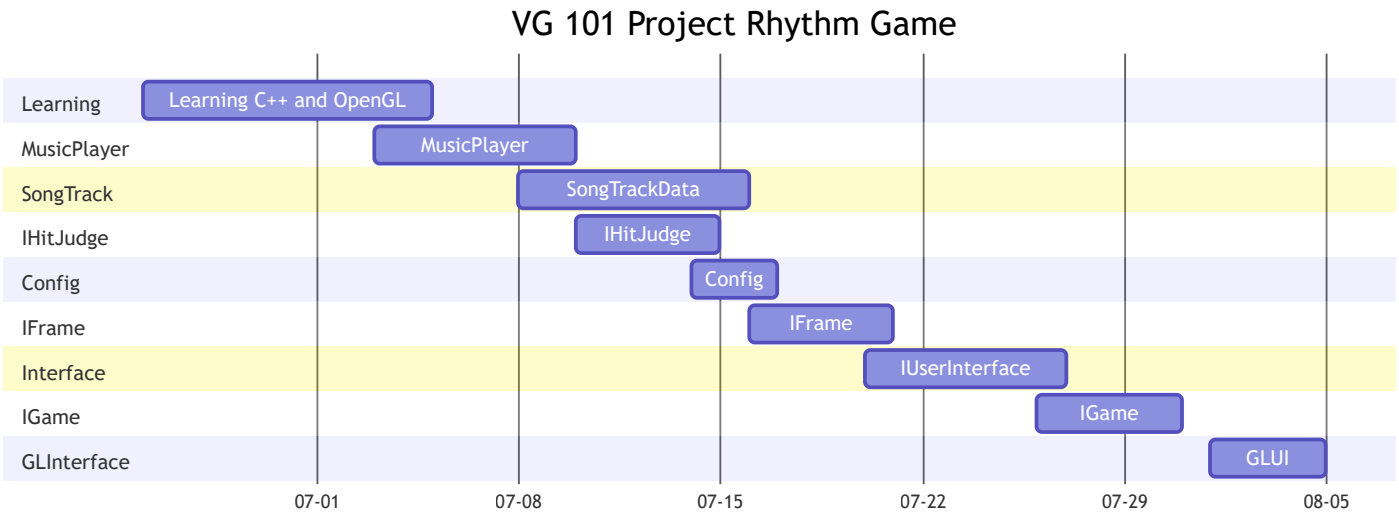


8. Outcome

	Proposed	Achieved	Comment
Bottom-line	1.1 Games can be played through Command Line	YES	
	1.2 As the music plays, "0"s will appear and move downward from the top of the screen in 4 or 6 columns	YES	'0' has been replaced by "#" for better visualization
	1.3 Players can tap space as "0" hit the line at the bottom of the screen	YES	
	1.4 Handle one piece of music score that the game can play	YES	
	1.5 Hitting the space key in time will gain points	YES	
	1.6 Total scores can be correctly calculated at the end of the game	YES	
Expected	2.1 Handle more than one piece of music that the game can play	YES	

	Proposed	Achieved	Comment
	2.2 As the music plays, notes will appear and move downward from the top of the screen in 4 or 6 columns	YES	
	2.3 Players can tap (press) corresponding keys bound to several columns as notes hit the line at the bottom of the screen.	YES	
	2.4 Hitting the notes more accurately increases the final score after completing the whole song.	YES	
	2.5 Games can be stopped and replayed according to the user's will	YES	
	2.6 Update and calculate scores while playing	YES	
Potential	3.1 Cover, Contents and Play/Stop/Exit are included in GUI (Graphical User Interface)	YES	Additional: Replay Implemented
	3.2 As the music plays, notes are falling from the top of the screen in 4 or 6 columns	NO	
	3.3 Correctness effect, combo effect and falling effect of the notes are included in GUI	NO	
	3.4 Use algorithm to transform midi to music that the game can play	NO	But we find a project that could transfer midi into the music
	3.5 Players tap or hold (press or long-press) corresponding keys bound to several columns according to the music as notes hit the line at the bottom of the screen.	NO	
	3.6 Calibration before playing the game	YES	
	3.7 Games can be played in GUI (Graphical User Interface)	NO	

9. Time Table and Actual Progress



10. Extra Learning Result

- JSON parsing with C++
- OpenGL and freeglut
- irrKlang audio libraries
- Windows Console APIs for C++

- virtual functions and inheritance

11. Task Assignment Summary

Wenqi Cao:

- `IHitJudge`
- `CmdInterface`
- `IRhythmGame::calibrate()`
- Game Rule and User Experience Design
- Demo Video
- README

Binhao Qin:

- `Config`
- `Note`, `SongTrackData`
- `MusicPlayer`
- `IFrame`, `IUserInterface`
- `FrameCLI`, `CmdInterface`
- `GLFrame`, `GLInterface`
- System Integration and Interface Design
- Report

Jingbo Qu:

- `FrameCLI`, `CmdInterface`
- `IRhythmGame`
- `GLFrame`, `GLInterface`
- UI and Color Design
- Report