¬Expected Magic

Requirements and Analysis Document

Version: 1.0

Date 01--04--17

Authors: Rasmus Tomasson, Arvid Runvik, Robert Agrell, Sofia Larborn

1 Introduction

Music has always been an important element in our society and in recent years games like guitar hero and rockband have flourished. Our vision is to create a cooperative entertaining experience where players can have lots of fun on a casual level.

Unlike guitar hero and rock band whose playability depends on whether or not the user wants to spend lots of money on gimmicky plastic guitars and other accessories, Unexpected Magic will incorporate the fun and entertaining cooperative aspect of a music party game while at the same time keeping the price at a reasonable level, being no charge at all.

Much like the educational music teaching application Synthesia, our game will mimic the way notes fall down towards the bottom of the screen, while adding the element of a game by letting the user interact directly with the falling notes when they reach a certain point and thus gaining score based on accuracy. The screen will be divided into segments of octaves and when playing cooperative mode every user will be responsible for his/her own section of the screen while all the players simultaneously plays the same song.

In short the user sees a falling note, her or she pushes a key on the keyboard and the application responds by playing the corresponding note. If the played note was correct the user will receive points. As an attempt to create variation we will implement different instruments, in a virtual sense, having different keyboard layouts and sound outputs.

The application will be solely developed as an offline desktop application for Windows, Linux and Mac. With our described game mechanics hand in hand with charming pixel graphic we think this application will be an entertaining musical experience for anyone interested in music, games or just having fun in general.

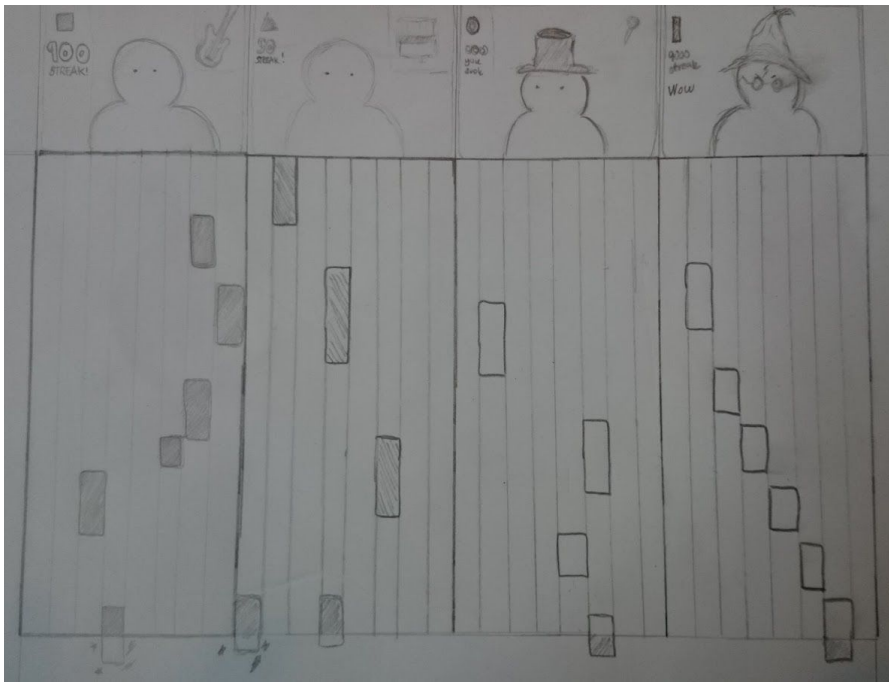
1.1 Definitions, acronyms and abbreviations

Word list in order to avoid confusion

- **Instrument** = a specific keyboard layout and sound output
- **Unexpected Magic** = the name of our game
- **Voice** = derived from the musical term, every player plays their voice of the song.
- **Note =** a falling object containing information about a tone.

2 Requirements

2.1 User interface



2.2 Functional requirements

2.2.1 Create a profile

- pick avatar
- name

2.2.2 Look at leaderboard

- see highscores

2.2.3 Options

- adjust possible options

2.2.4 Play a song/round

- pick song
- pick profile
- number of players
- pick voice/instrument
- play note
- receive score

2.2.5 Play playlist/different game mode

- pick mode/customize playlist

2.3 Non-functional requirements

2.3.1 Usability

Unexpected magic will attempt to be conventional in its game design and graphical user interface design. The gameplay mimicking previously created music games will be intuitive for experienced players. New user will be able to learn from either a visual or lyrical tutorial.

All and all intuitiveness and likeness towards actual instruments especially piano will the serve the purpose of intuitive usability

2.3.3 Performance

Our goal for Unexpected Magic is that the application should be able to play a song without any sound delay or visual delay thus keeping the experience immersive and focused on the musical aspect of gameplay.

2.3.4 Supportability

Unexpected magic will as previously mentioned run on Windows, Linux and Mac. Possibly support for multiple keyboards or similar controls will be implemented.

2.3.5 Implementation

The Java environment will be used to implement this application. Libgdx will be used as an additional library aiding with functionality on the aspect of game logic. Gimp will be used in order to create pixel graphic. Some kind of musical library might be used depending on whether or not libgdx has the requirements in relation to the musical aspect of our application

2.3.6 Packaging and installation

As of now we are unsure of exactly how the application will be packaged and delivered to the users.

2.3.7 Legal

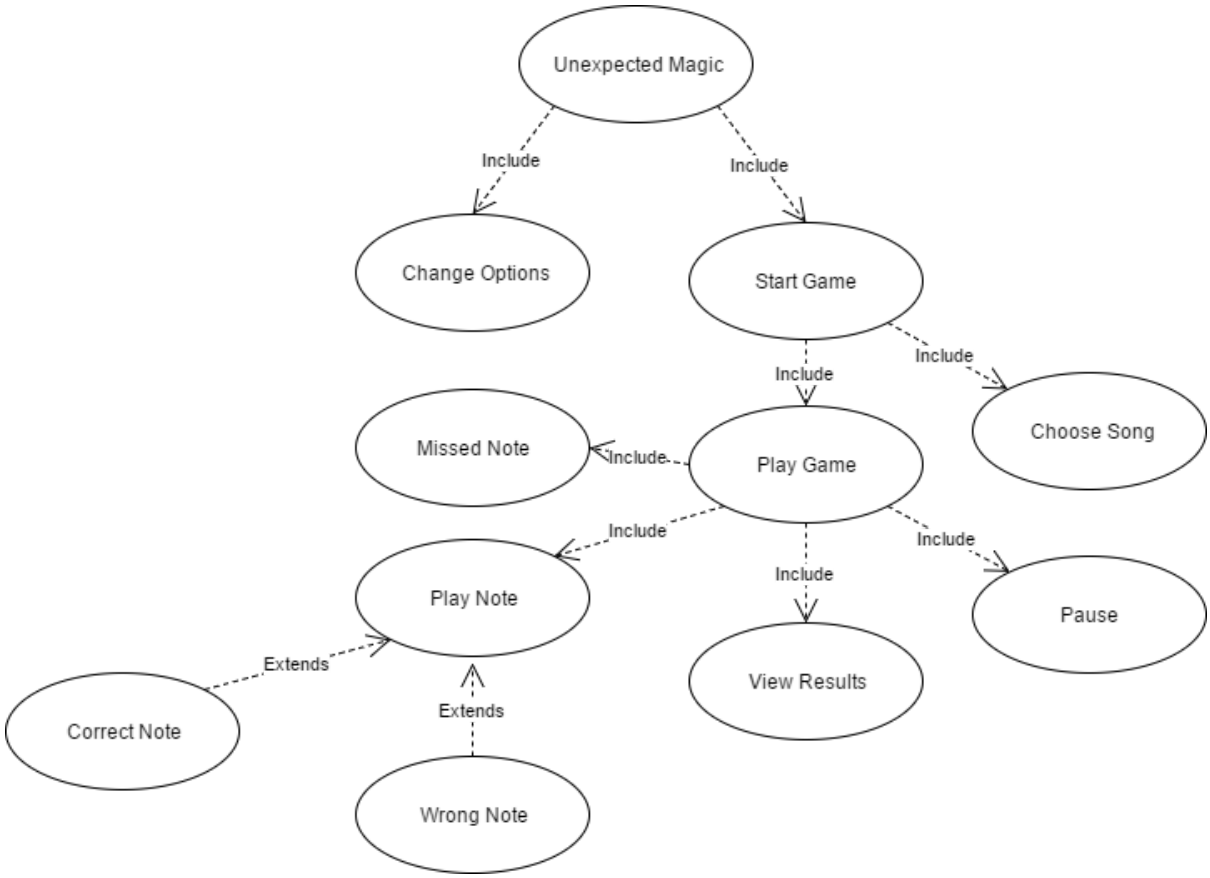
Songs played in the game will not be included without correct legal circumstances.

2.3.8 Testability

There should be automated test verifying all use cases.

3 Use cases

3.1 Use case diagram



3.2 Use case listing

3.2.1 High priority use cases

- start game ?
- play note
- start song ?

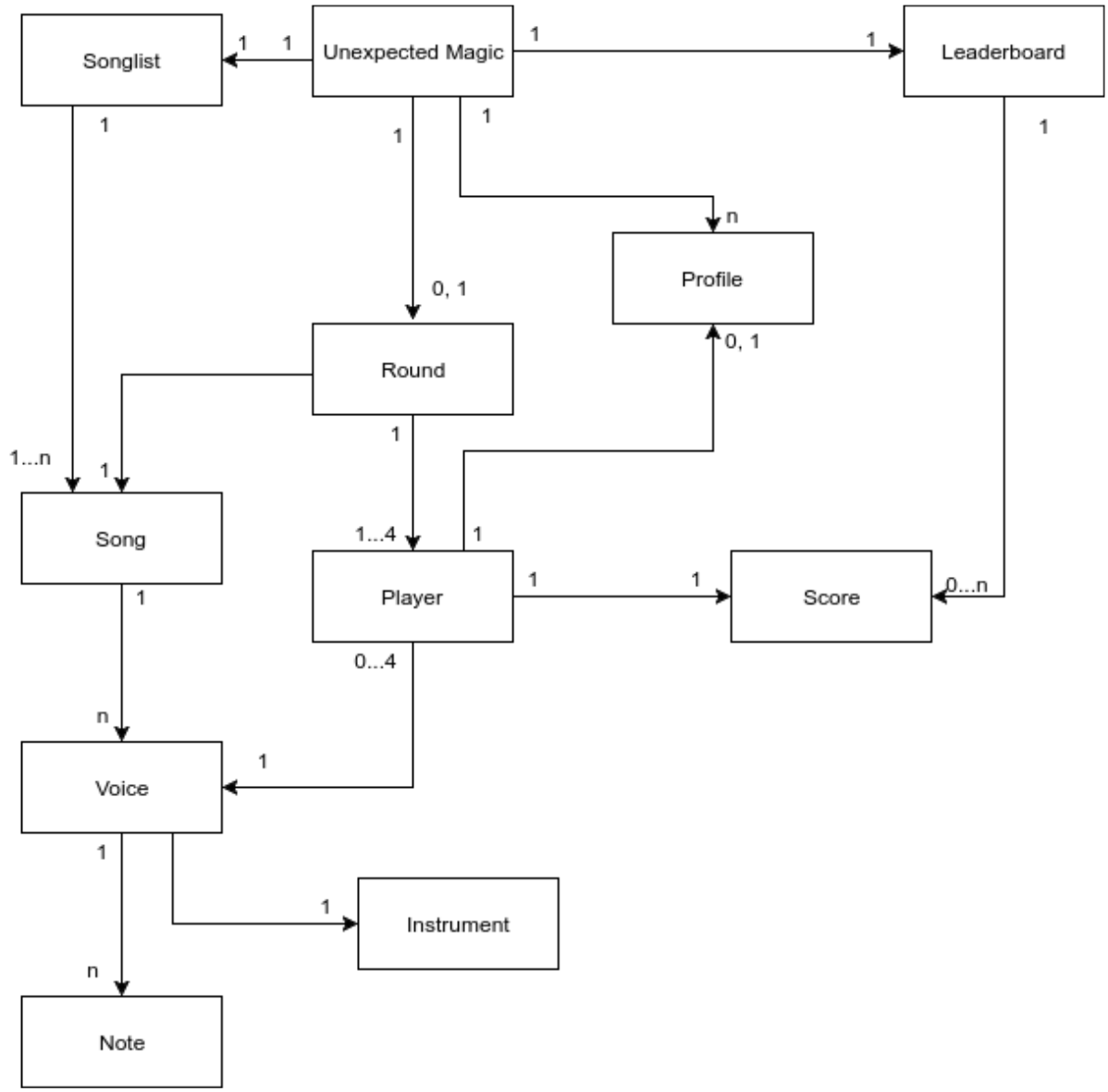
3.2.2 Medium priority use cases

- choose song
- choose players
- choose instrument
- view end score

3.2.3 Low priority use cases

- pause game
- create profile
- adjust options
- view leaderboard

4 Domain model



4.1 Class responsibilities

Unexpected Magic: Main Class, this class keeps track of everything.

Songlist: Keeps track of all the songs in the song library.

Leaderboard: Keeps track of statistics from past games.

Profile: Handles personalization options such as names and avatars.

Player: In-game representation of the player, keeps track of what notes are played.

Song: Representation of the song, keeps track of the voices that make it up.

Voice: Keeps track of the notes and instrument of a particular voice.

Instrument: Defines the instrument.

Note: A note, keeps track of pitch and note length.

Score: Keeps track of score, note streaks.

5 References

MONOPOLY-EXAMPLE RAD
http://www.cse.chalmers.se/edu/year/2017/course/tda367/lectures/MonopolyRAD.pdf