두둠칫~!

Team #6 음치박치

141317 공철규

141322 심기성
Dept. Electronics and Computer Engineering
Chonnam National University
Gwangju

I. 개발동기

팀원들이 모여 프로젝트 주제에 대해 토의하던 중에 바쁜 학교생활로 인해 지쳐 있는 학생들이 눈에 띄었다. 그 모습을 보고 학생들 뿐만 아니라 바쁜 일상에 지친 사람들 모두가 즐길 수 있는 게임을 만들어보자는 의견이 나왔고, 누구나 쉽고 신나게 할 수 있는 리듬게임을 프로젝트의 주제로 정하게 되었다.

자신이 좋아하는 음악을 들으면서 즐길 수 있는 리듬게임은 지친 사람들이 활력을 되찾고, 스트레스를 해소할 수 있게 해줄 것이다. 또한 어려운 게임규칙이 있거나 많은 시간을 소비하는 게임이 아니기 때문에 바쁜 생활 속 자투리 시간에 할 수 있는 적합한 게임이 될 것이다.

우리는 재미있고 실용적인 게임을 만드는 것을 목적으로 하여 기존의 게임에 새로운 기능들을 추가한 리듬게임 '두둠칫~!! '의 개발을 계획하게 되었다.

II. 특징

A. *Related applications*

기존의 온라인 건반형 리듬게임인 'O2Jam(오투잼) '은 음악에 맞춰 내려오는 노트가 판정선에 일치할 때 알맞은 키를 누르는 방식의 게임이다. 노트는 숏노트와 롱노트가 있으며 숏노트는 단순히 해당되는 키를 누르면 되나, 롱노트는 길이에 맞춰 그 키를 눌렀다가 끝지점에 맞춰 떼면 된다. 키를 누르는 순간 판정선에 노트가 얼마나 가까웠는가에 따라 COOL, GOOD, BAD, MISS의4단계로 판정이 나뉜다. COOL과 GOOD의 판정을 연속해서 받으면 콤보가 오르며, BAD와MISS의 판정을 받으면 HP가 감소한다. HP의 값이0이 되면 자동으로 게임이 종료되는 서바이벌 게이지 방식이다.

B. *Characteristics of the target application*

'두둠칫~!!'은 'O2Jam(오투잼)'과 동일한 방식의 리듬게임이다. 오투잼처럼 플레이 할 수 있는 노멀모드, 게임 점수에 영향을 줘서 사용자에게 다채로운 느낌을 줄 수 있는 아이템 모드가 있어 사용자에게 흥미를 유발시킨다.

III. Requirement Lists

이 게임은 즐겁고 가벼운 마음으로 플레이하면서 신나는 노래와 함께 평소에 쌓였던 스트레스를 풀고자 하여 개발을 계획하게 된 게임이다. 하지만 기존의 리듬게임과 별 차이 없다면 금방 실증을 느낄 수 있다고 생각하여 다음과 같은 옵션을 넣어주어 흥미를 가지도록 하였다.

- 시작화면에 2가지 버튼을 두어 게임을 시작하고 종료하는 것을 선택할 수 있다.
- 메뉴화면에는 좌 우 버튼을 통해서 노래를 선택할 수 있다.
- 메뉴화면에서는 현재 선택된 노래의 이미지를 볼 수 있다.
- 메뉴 화면에서 현제 선택된 노래의 하이라이트 부분이 나온다
- 게임을 Item / Normal 두 가지 모드로 나누어 사용자의 취향에 맞게 플레이 할 수 있다.
- Item Mode는 특정 위치에서 2가지 다른 색을 가진 노트가 불시에 떨어져 각 노트에 따라 랜덤으로 특수효과가 발생하도록 한다.
- HP 형식의 기존 게임과는 다르게 노트를 12번 놓치게 되면 게임이 오버된다.

IV. FunCtion SpecifiCation

분 류	클래스	매소드	설 명
메 뉴	Initial	public Initial()	- 초기 화면을 담당하는 기본 생성자
		public void mouseEntered(MouseEvent e)	- 마우스가 버튼에 올라갔을 때의 이벤트를 나타내는 메소드(이미지, 커서 변경)
		public void mouseExited(MouseEvent e)	- 마우스가 버튼에서 나왔을 때의 이벤트를 나타내는 메소드 (이미지, 커서 변경)
		public void mousePressed(MouseEvent e)	- 버튼 클릭 시 이벤트가 발생하도록 하는 메소드 (판넬 변경. 프로그램 종료)
		public void paintComponent(Graphics g)	- contentPane에 이미지를 그려주는 메소드
	Main	public static void main(String[] args)	- 메인 메소드
		new Initial()	- 메인 메소드 실행 시 초기화면이 담긴 프레임을 호출해주는 생성자
	Menu	public Menu(JFrame menu)	- 메뉴화면을 담당하는 기본 생성자
		public void mouseEntered(MouseEvent e)	- 마우스가 버튼에 올라갔을 때의 이벤트를 나타내는 메소드(이미지, 커서 변경)
		public void mouseExited(MouseEvent e)	- 마우스가 버튼에서 나왔을 때의 이벤트를 나타내는 메소드 (이미지, 커서 변경)
		public void mousePressed(MouseEvent e)	- 버튼 클릭 시 이벤트가 발생하도록 하는 메소드 (곡 변경. 게임 시작)
		public void paintComponent(Graphics g)	- contentPane에 이미지를 그려주는 메소드
		public void selectedTrack(int nowSelected)	- 현재 선택된 음악에 대해 Track Class에 저장된 음악의 타이틀 이미지, 앨범 이미지, Selected Music을 불러오고 음악을 시작해주는 메소드
		public void selectedLeft()	- 왼쪽화살표 버튼이 선택된 음악에서 trackList에 저장된 이전 음악으로 이동할 수 있는 기능을 갖도록 해주는 메소드
		public void selectedRight()	- 오른쪽화살표 버튼이 선택된 음악에서 trackList에 저장된 다음 음악으로 이동할 수 있는 기능을 갖도록 해주는 메소드
		public void gamestart(int nowSelected,Boolean normal)	- 현재 선택된 음악과 모드에 대해 게임을 실행 할 수 있도록 해주는 메소드
	Track	gettter And setter	- 각 변수에 대해 값들을 가져오고 대입해주는 메소드
		public Track(String titleImage, String startImage, String gameImage, String startMusic, String gameMusic)	- 각 변수로 설정된 값들을 트랙리스트에 추가시켜 주는 생성자
	Music	public Music(String name, boolean isLoop)	- 음악 파일을 가져와주는 생성자
		public void close()	- 음악을 종료시켜 주는 메소드
		public int getTime()	- 음악의 시간을 가져와주는 메소드
		public void run()	- 음악을 재생시켜주는 메소드
게 임	Game	public Game(Music Music, int notetime[][],boolean item)	- 음악, 노트가 생성되는 시간, 아이템모드인지 확인하는 불린, 3가지의 값을 받아 게임을 생성하는 생성자
		public void makeNote(int key)	- 키값을 받아 그 킷값에 해당되는 노트를 생성하는 메소드
		public void checkNote(int key)	- 키가 눌러졌을 경우 노트가 정확한 위치에 있는지 확인 후 점수를 올리고 노트를 삭제하는 메소드
		public void buffAction()	- 아이템 모드에서 버프가 작동시 랜덤으로

			버프효과를 주는 메소드
		public void nurffAction()	- 아이템 모드에서 너프가 작동시 랜덤으로 너프를 먹이는 메소드
		public void musicQ()	- 음악을 실행하는 메소드
	MyKey Listener	**public void** keyPressed	- 키가 눌러질 경우 키 이펙트를 호출하고 눌러졌다는걸 알려주는 메소드
		public void keyReleased(KeyEvent e)	- 키가 릴리즈 될 경우 키 이펙트를 없에도록 호출하는 매소드
	Back	**public void** paintComponent(Graphics g)	- 백그라운드 이미지를 그린후 킷값이 눌러지면 키 이펙트를 그리는 메소드
	Game Over	**public void** paintComponent(Graphics g)	- 게임이 끝났을 경우 게임오버 혹은 게임 클리어 창을 띄어주는 메소드
	MyPanel	**public void** paintComponent(Graphics g)	- 노트에 해당되는 판넬을 생성하는 메소드
	Anzi young	Anziyoung(**boolean** item)	- 아이템모드 불린값을 받아 아이템모드인지 확인하는 메소드
		run()	- 게임을 실행하는 메소드
	RED	RED(**boolean** item)	- 아이템모드 불린값을 받아 아이템모드인지 확인하는 메소드
		run()	- 게임을 실행하는 메소드
	Twice	Twice(**boolean** item)	- 아이템모드 불린값을 받아 아이템모드인지 확인하는 메소드
		run()	- 게임을 실행하는 메소드

V. Usecases

- 플레이어가 프로그램을 시작할 경우 초기 화면이 생성되고 시작하기 버튼을 통해 메뉴화면으로 이동 가능하고 종료하기 버튼을 통해 프로그램 종료가 가능하다.

- 메뉴화면에서 좌우버튼을 통해 음악 선택이 가능하고 게임의 모드 버튼(NORMAL / ITEM)을 통해 바로 게임하면으로 이동 가능하다.

- Normal Mode의 경우

  · 노래의 박자에 맞춰 노트가 떨어지게 되는데 노트가 바닥에 떨어지는 순간에 그에 해당하는 키를 누르지 못하면 miss가 발생하게 된다.

  · miss가 한 게임에서 12번 발생 할 경우 플레이 도중에 게임오버가 되며 게임 오버 창이 나타나고 홈 화면으로 돌아온다.

  · miss가 발생하지 않고 계속해서 떨어지는 노트에 키를 누른다면 Combo 카운터가 올라가 더 많은 점수를 획득 할 수 있게 된다.

- Item Mode의 경우

  · 게임 방식은 Normal Mode와 같다.

  · 특정 위치에서 랜덤하게 빨간색과 노란색의 노트가 떨어지는데 그 기능은 다음과 같다.

  1) 노란색 노트 : 노란색 노트를 맞추게 되면 아래의 3가지 버프 중 랜덤으로 한가지를 받게 된다.

  (1). 5초간 점수 2배

  (2). 5초간 콤보 수가 떨어지지 않는다.

  (3). 점수 +100점

  (4). miss가 2개 사라진다.

  2) 빨간색 노트 : 빨간색 노트를 맞추지 못하면 아래의 3가지 너프 중 랜덤으로 한가지를 받게 된다.

  (1) 5초간 점수 0.5배

  (2) 5초간 콤보가 0에서 오르지 않는다.

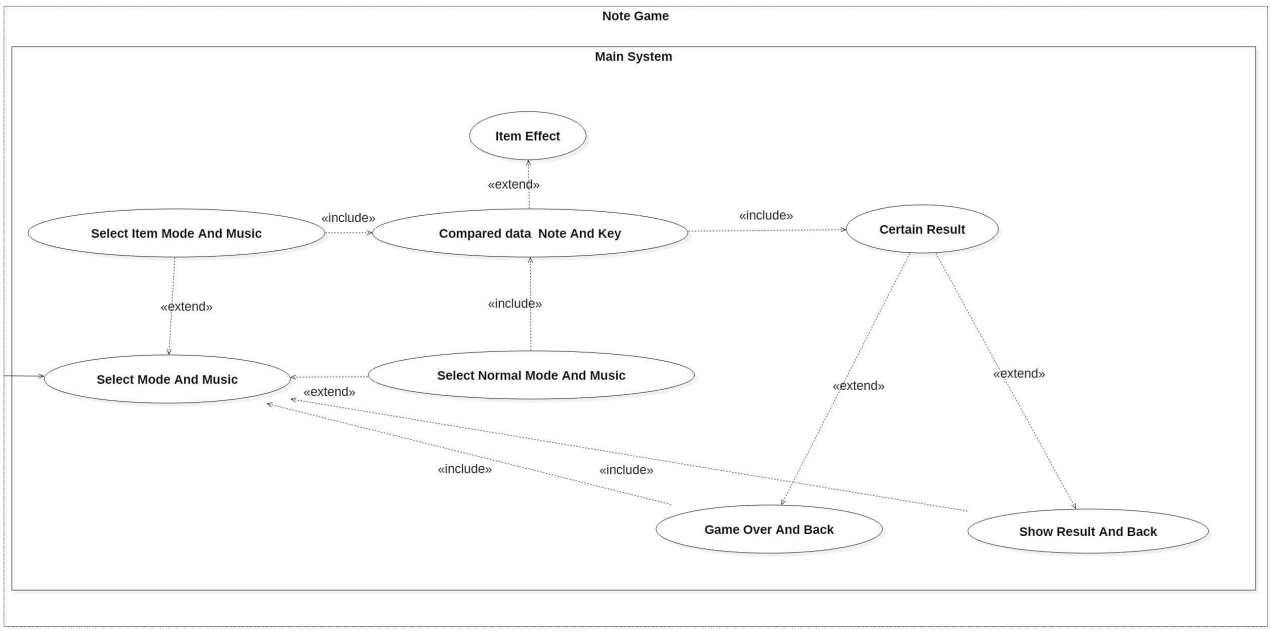
  (3) 점수 -100점

  (4) miss 2개 추가

- 게임 클리어 / 실패 시

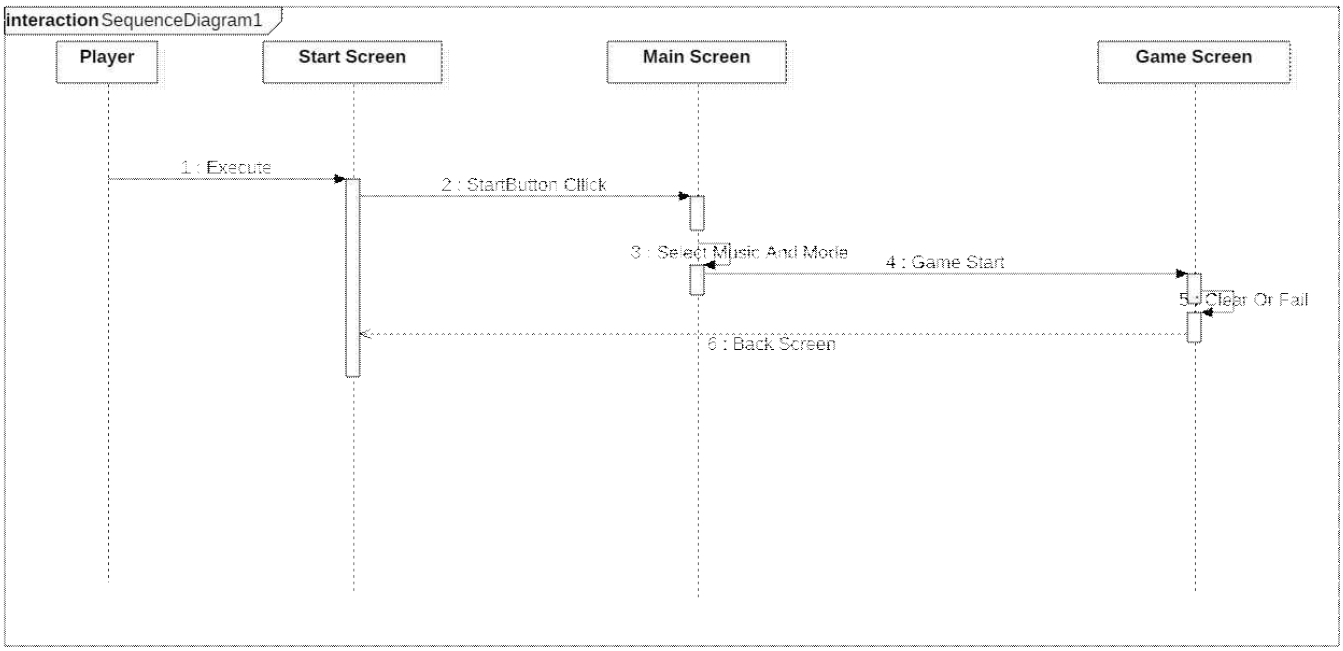
  · 게임이 클리어되거나 실패할 경우 다시하기 버튼이 생성되며 클릭 시 초기화면으로 돌아가게 된다.

VI. Usecase Diagram

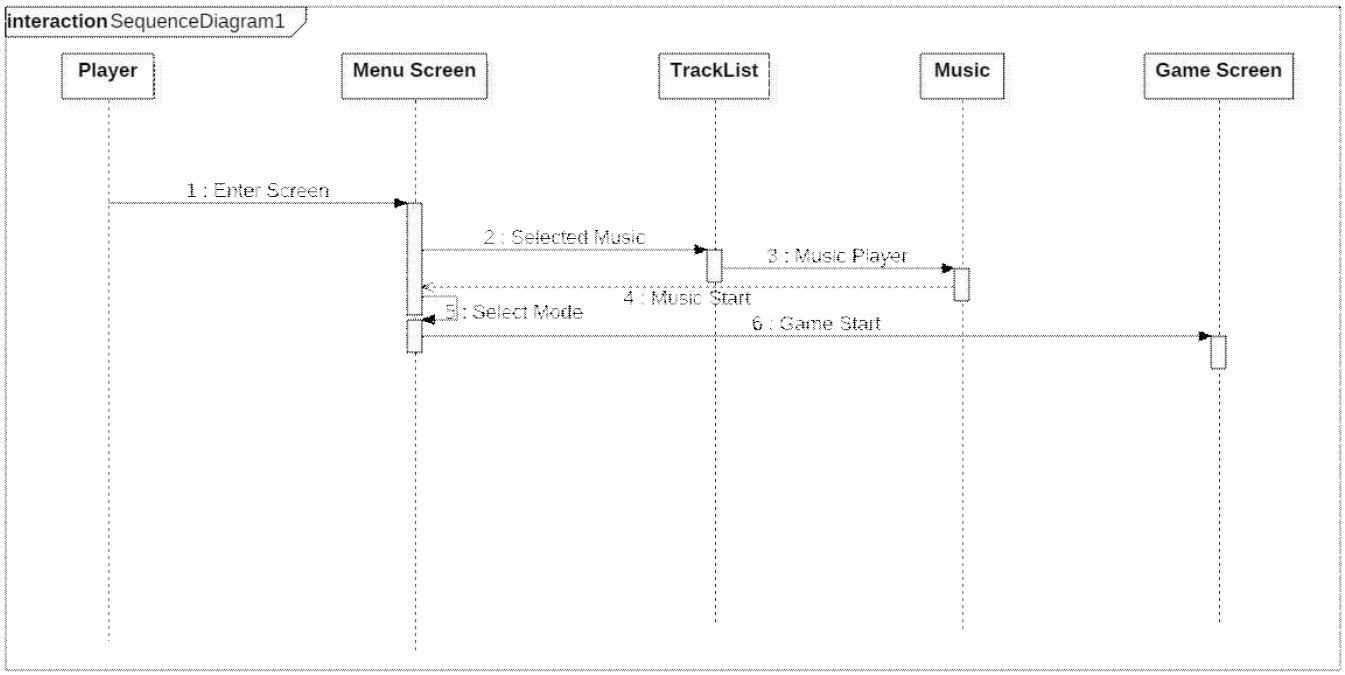


VII. Sequence Diagram

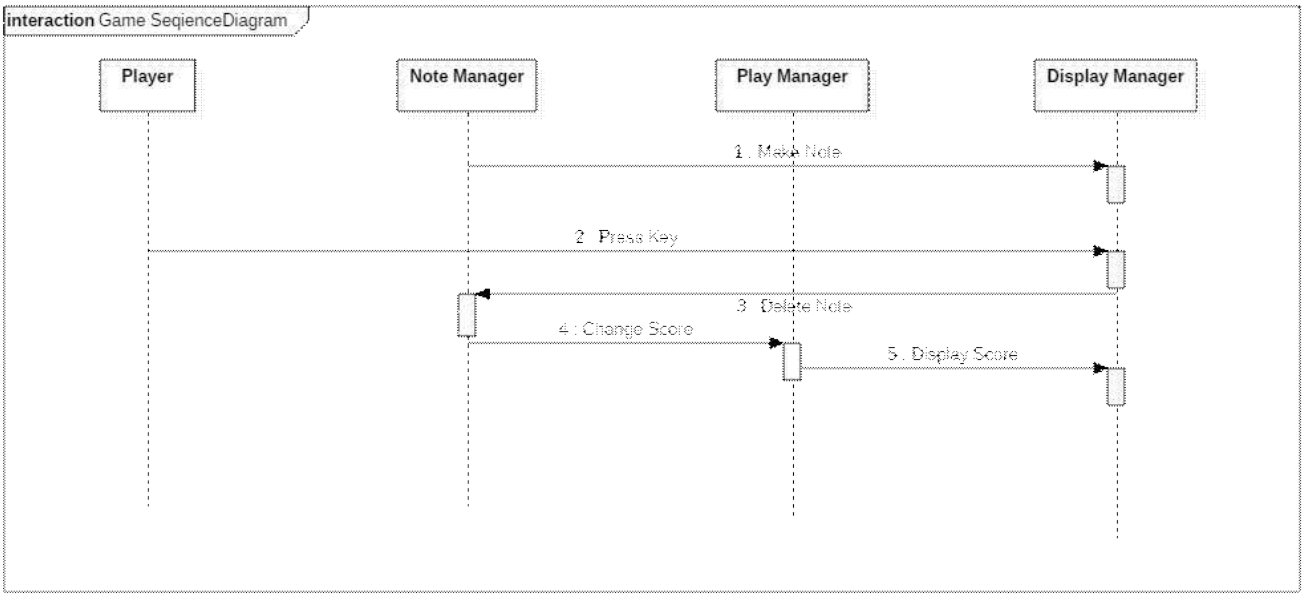
전체 Sequence Diagram



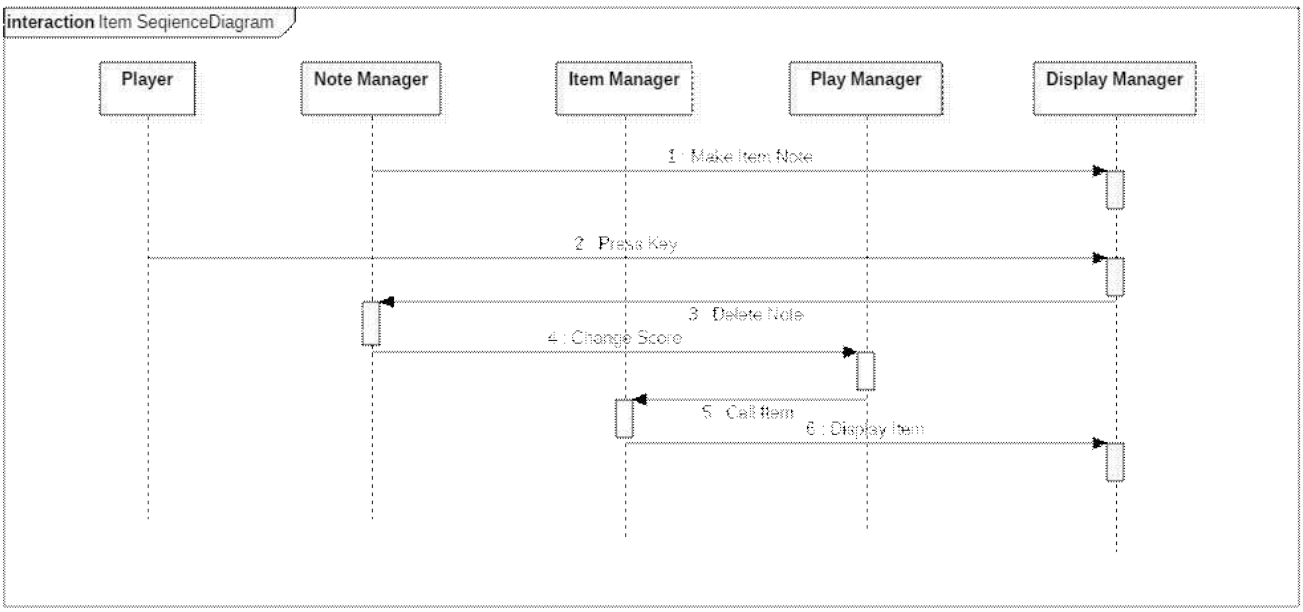
메뉴 화면 Sequence Diagram



게임 Sequence Diagram

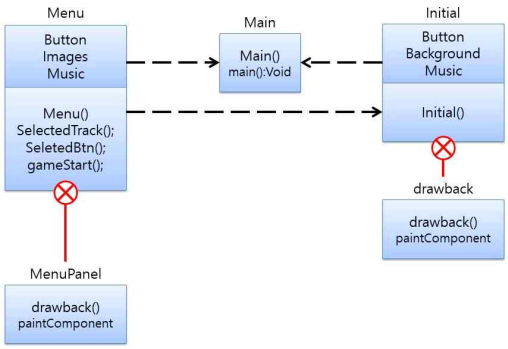


아이템 Sequence Diagram



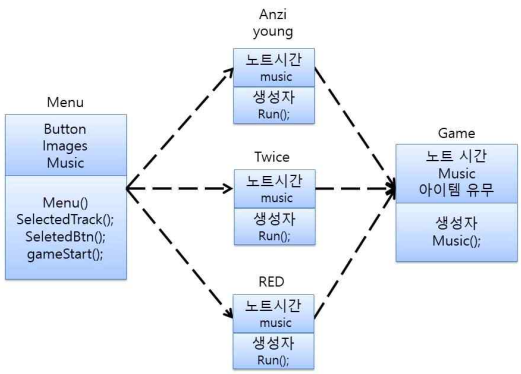
VIII. Class Diagram

1. 메인클래스 및 메뉴 클래스



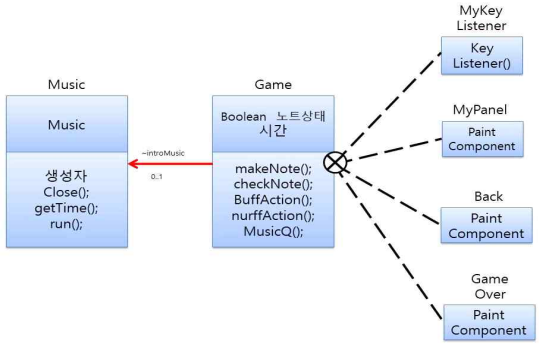
매인클래스는 메인함수를 가지고 있는데 이것을 이용하여 Initial클래스를 객체로 소환한다. Initial클래스는 시작화면을 담당하고 있으며 drawback이라는 내부 클래스를 가지고 배경을 그린다. menu 클래스는 곡 선택 화면을 만드는 클래스로서 Initial 클래스와 매인함수에 의해 만들어지는데 MenuPanel이라는 내부 클래스를 통해서 배경화면을 만들어 낸다.

2. 게임클래스와 게임 객체



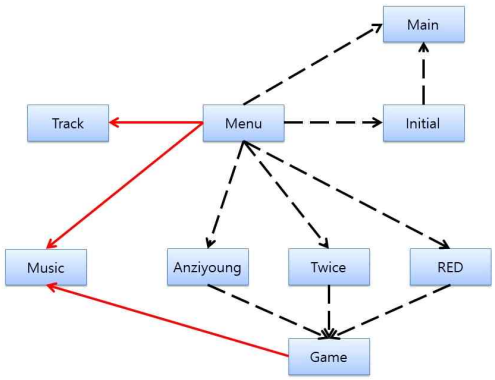
Menu클래스에서 게임을 실행시키는 방법에 관한 다이어 그램이다. 우리 프로그램은 3개의 게임을 가지고 있는데 그 게임은 각각 Anziyoung, Twice, RED클래스에 해당한다. 각 클래스는 노트시간과 음악을 가지고 있으며 게임 클래스에 넣어주어 각 게임의 특성을 갖는 게임 클래스를 호출해 낸다.

3. 게임클래스와 하위 클래스



게임 클래스는 4가지 하위 클래스를 갖는다 MyPanel은 게임에 필요한 노트를 생성해주며 Back는 배경화면을 제작 GameOver는 게임이 끝났을 때 게임 오버 화면, 혹은 게임 클리어 화면을 띄워주는 클래스이다. 게임 클래스 내의 시간과 노트상태를 서로 긴밀하게 이용하며 시간의 흐름에 맞춰 혹은 노트상태에 맞춰 상호 연동이 있다. 또한 introMusic이라는 객체를 갖는데 Music클래스를 이용하여 생성을 한다.

4. 전체 클래스



전체 클래스간의 관계도는 다음과 같다 중앙을 기준으로 위에 4개의 클래스와 아래 Music을 제외한 4개의 클래스로 구분할 수 있으며 위의 클래스는 메뉴를 아래 클래스들은 게임을 담당하고 있다.

IX. User Interface Designs

초기화면

< 초기화면 >

1. 시작하기 버튼
   - 메뉴 화면으로 이동하는 버튼
2. 종료하기 버튼
   - 프로그램을 종료해주는 버튼

메뉴 화면

< 메뉴 화면 >

1. 타이틀 이미지 및 앨범 이미지
   - 현재 선택 된 곡의 타이틀 이미지 및 앨범 이미지를 보여주는 부분

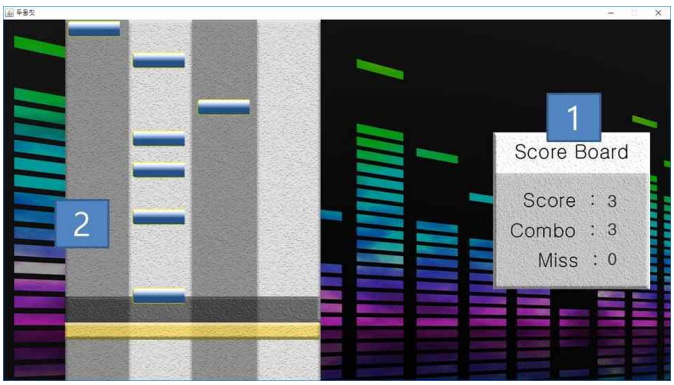
2, 3. 곡 변경 버튼

   - 선택 곡을 좌 우로 넘겨주는 버튼
   - 클릭 시 아래 그림과 같이 타이틀 이미지 및 앨범 이미지가 변경된다.

4. 노멀 모드 버튼
   - 누르면 해당곡의 노멀버전을 플레이 할 수 있는 버튼

5. 아이템 모드 버튼
   - 누르면 해당곡의 아이템 버전을 플레이 할 수 있는 버튼

노멀모드 게임 화면



<노멀모드 게임 화면>

1. 스코어 보드

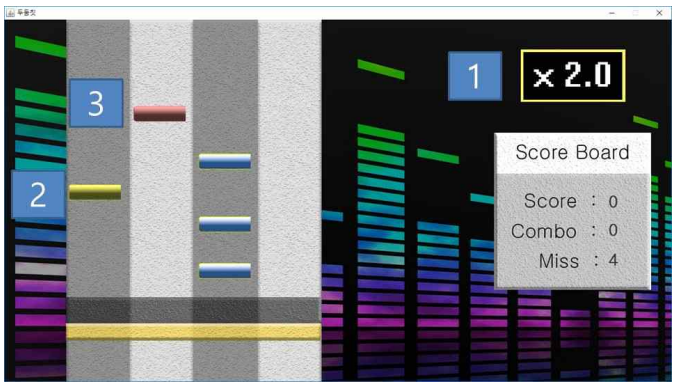
    - 스코어를 띄워주는 부분

2. 게임 진행화면

    - 리듬에 맞게 파란색 노트가 떨어진다.

    - 각 라인은 왼쪽부터 a,s,d,f에 대응하며 키를 누를시 붉은색 이펙트가 뜬다.

아이템 모드 게임 화면



<아이템 모드 게임 화면>

1. 아이템 출력

    - 현제 아이템 정보가 뜬다

    - 노란색은 버프 빨간색은 너프를 뜻한다.

2. 노란색 노트

    - 맞추면 버프를 주는 노트

3. 붉은색 노트

    - miss가 나면 너프를 주는 노트

X. Screenshot of API Manual

1. 자바독 인덱스

Package do_doom_chit

Class Summary

Class	Description
Anziyoung	썸탈거야 게임을 만드는 클래스
Game	게임의 기본이 되는 클래스 이 클래스를 객체로 생성하게 되면 게임을 할 수 있게 된다.
Initial	초기 화면을 담당하는 클래스입니다.
Main	메인 클래스 메인 메소드가 있는곳
Menu	곡 선택 및 모드 선택 화면을 담당하는 클래스입니다
Music	음악을 실행시켜주는 클래스입니다.
RED	빨간맛 게임을 만드는 클래스
Track	음악의 트랙을 관리하는 클래스입니다.
Twice	트와이스 게임을 만드는 클래스

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

2. 게임 클래스 설명

do_doom_chit

Class Game

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame
            do_doom_chit.Game

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants

---

public class **Game**
extends javax.swing.JFrame

게임의 기본이 되는 클래스 이 클래스를 객체로 생성하게 되면 게임을 할 수 있게 된다.

Since:
2017-10-14

Version:
1.4

Author:
심기성

See Also:
Serialized Form

3. 게임 클래스 메소드 설명

Constructor Summary

Constructors

Constructor and Description
Game(Music Music, int[][] notetime, boolean item) 게임이 실행되는 곳

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	buffAction() 버프효과를 주는 메소드
void	checkNote(int key) 노트를 체크하는 메소드*
void	makeNote(int key) 노트를 만드는 메소드
void	music() 음악을 실행해주는 메소드
void	nurfAction() 너프효과를 주는 메소드

Methods inherited from class javax.swing.JFrame

getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getGraphics, getJMenuBar, getLayeredPane, getRootPane, getTransferHandler, isDefaultLookAndFeelDecorated, remove, repaint, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setIconImage, setJMenuBar,

4. 메뉴 클래스 설명

do_doom_chit

Class Menu

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
          do_doom_chit.Menu

All Implemented Interfaces:

java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.accessibility.Accessible

---

public class **Menu**
extends javax.swing.JPanel

곡 선택 및 모드 선택 화면을 담당하는 클래스입니다

Author:

공철규

See Also:

Serialized Form

Nested Class Summary

Nested classes/interfaces inherited from class javax.swing.JComponent

javax.swing.JComponent.AccessibleJComponent

5. 메뉴 클래스 메소드 설명

Constructor Summary

Constructors

Constructor and Description
Menu(javax.swing.JFrame menu) Menu클래스에 대한 기본 생성자 입니다.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	gamestart(int nowSelected, java.lang.Boolean normal) 현재 선택된 음악과 모드에 대해 게임을 실행 할 수있도록 해주는 메소드입니다.
void	selectedLeft() 왼쪽화살표 버튼이 선택된 음악에서 trackList에 저장된 이전 음악으로 이동할 수 있는 기능을 갖도록 해주는 메소드입니다.
void	selectedRight() 왼쪽화살표 버튼이 선택된 음악에서 trackList에 저장된 다음 음악으로 이동할 수 있는 기능을 갖도록 해주는 메소드입니다.
void	selectedTrack(int nowSelected) 현재 선택된 음악에 대해 Track Class에 저장된 음악의 타이틀 이미지, 앨범 이미지, Selected Music을 불러오고 음악을 시작해주는 메소드입니다.

Methods inherited from class javax.swing.JPanel

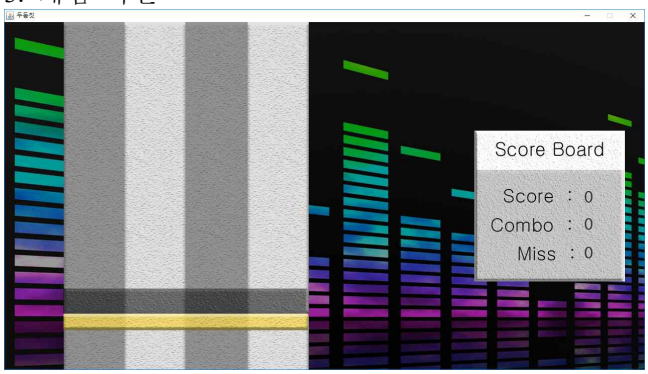
getAccessibleContext, getUI, getUIClassID, setUI, updateUI

XI. Screenshot of application demo

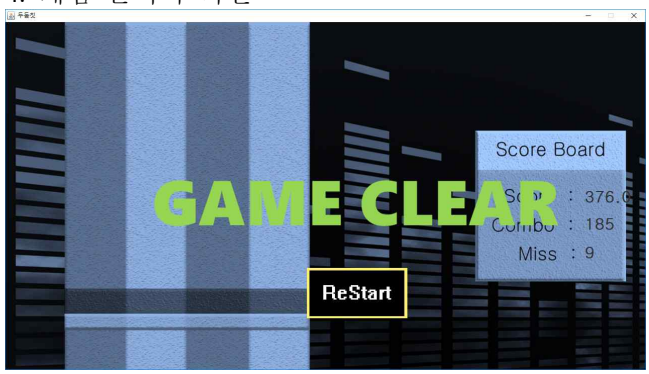
1. 시작화면

2. 곡 선택 화면

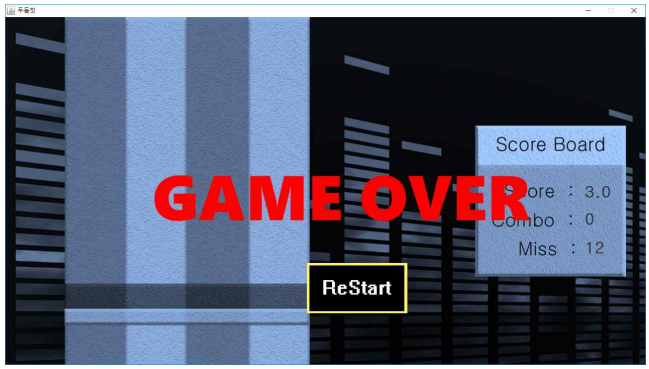
3. 게임 화면



4. 게임 클리어 화면



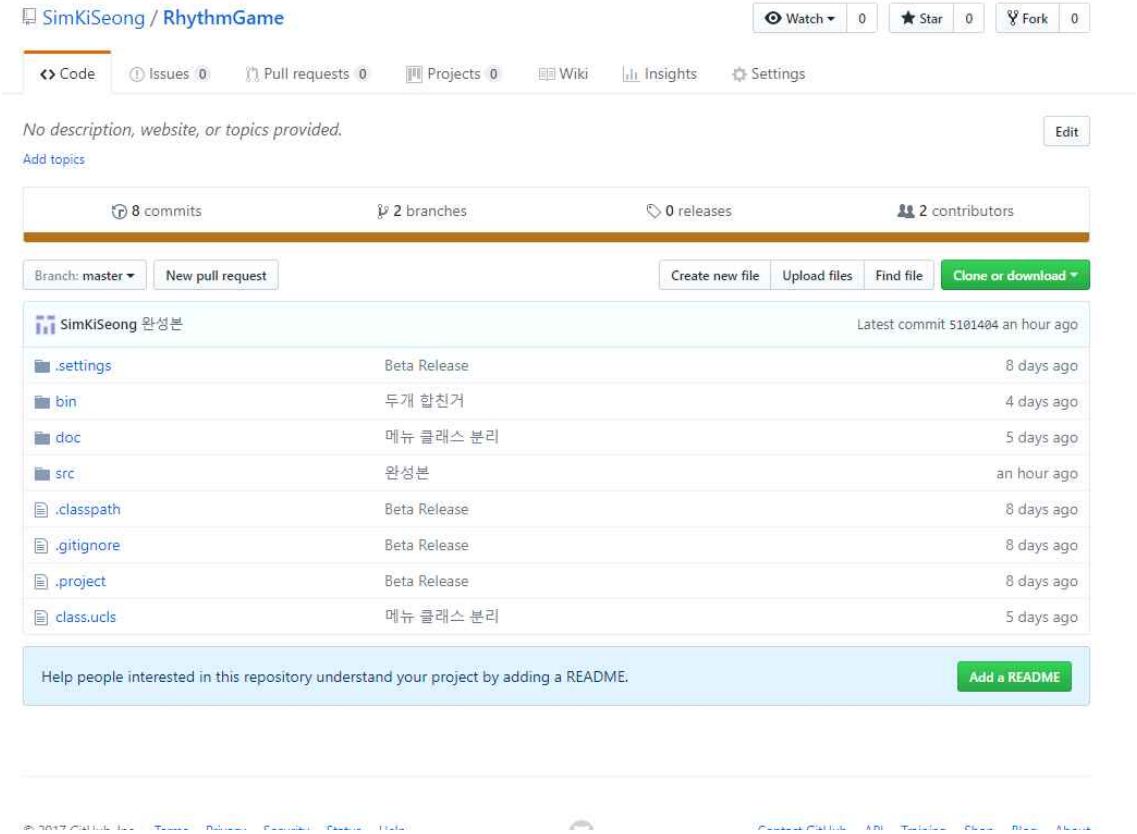
5.게임 오버 화면



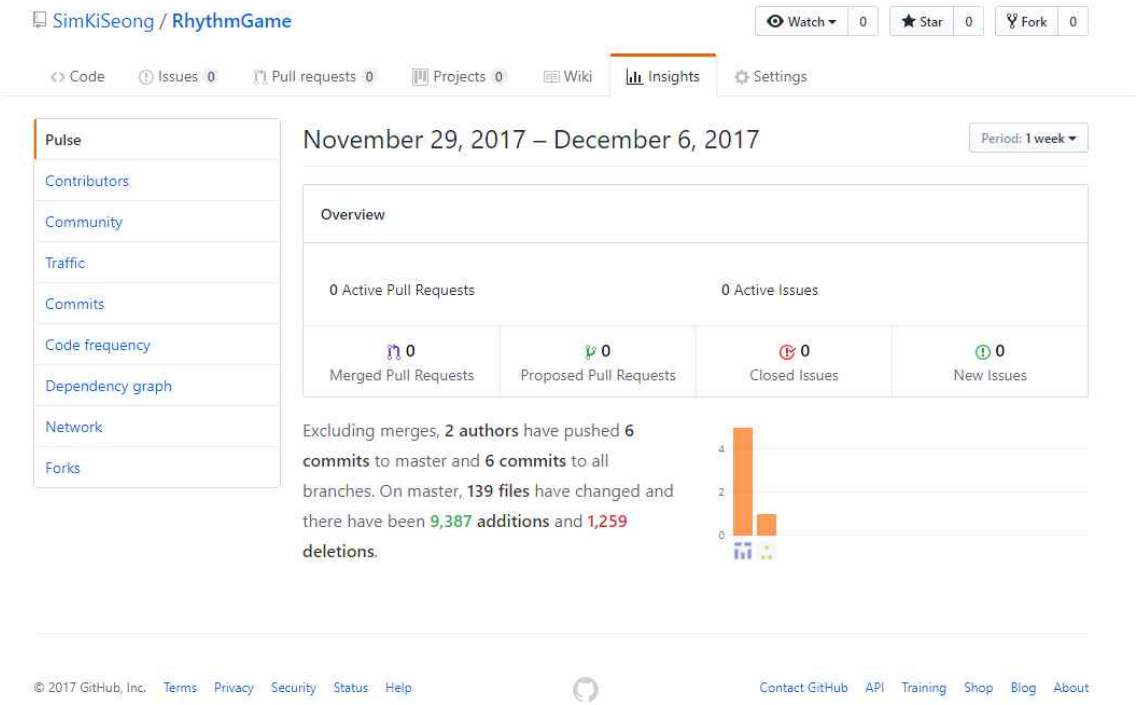
XII. Screenshot of Githun pages

URL : https://github.com/SimKiSeong/RhythmGame

1. code page



2. Insights Pulse



XIII. Work distribution & Progress of Each member

이 름	클래스	세부 사항	라인수
공철규	Menu Initail Track Music	프로그램 시작 시 나타나는 초기화면과 그 이후 나타나는 메인화면을 구현하였습니다. 초기화면은 배경화면과 배경음악, 시작하기버튼과 종료하기 버튼으로 구성되어 있고 일단 초기화면이 나타나면 배경음악이 무한 반복으로 함께 재생됩니다, 화면의 시작하기 버튼 클릭 시 패널이 메뉴화면으로 전환되고 종료하기 버튼 클릭 시 프로그램이 종료됩니다. 메뉴화면으로 전환 될 경우 트랙리스트에 추가된 음악 중 첫 번째 인덱스의 음악의 하이라이트가 재생됩니다. 메뉴화면은 음악의 타이틀 이미지, 앨범 이미지, 음악을 넘기는 좌, 우 버튼, 게임의 모드를 선택할 수 있는 노말모드 버튼과 아이템 모드 버튼 2개로 구성되어 있습니다. 트랙리스트의 음악 추가는 임의적으로는 불가능합니다. 앨범이미지 하단의 오른쪽 화살표 버튼 클릭 시 트랙리스트의 인덱스가 1 증가하여 다음 곡으로 넘어가게 되고 반대로 왼쪽 화살표 버튼 클릭 시 인덱스가 1 감소하여 이전 노래로 돌아가게 됩니다. 각 버튼이 클릭되어 음악이 바뀔 때 마다 음악에 맞는 앨범이미지와 타이틀 이미지가 다시 그려지며 음악의 하이라이트가 재생됩니다. 음악은 1번만 재생되며 하이라이트가 끝날 경우 음악이 멈추게 됩니다. 화면의 오른쪽에 위치한 노말버튼과 아이템 버튼 클릭 시 각 모드에 맞는 게임이 실행됩니다.	617
심기성	Game Anziyoung RED Twice	게임 파트에 관한 전반적인 사항을 담당하였습니다. 기본적으로 모든 게임의 모듈이 될 게임클래스를 작성하였습니다. 이 게임모듈을 기반으로 3개의 게임을 만들었으며 아이템모드와 노멀모드까지 구현하였습니다. 게임을 맡게 되었는데 제일 고민되는 일은 메모리관련 문제였습니다. 게임이라는 프로그램 특성상 이미지를 많이 로드해야 합니다. 이미지는 용량이 매우 크기 때문에 많이 로딩을 하면 할수록 당연히 컴퓨터는 랙이 걸리게 되고 랙이 걸린다면 게임은 게임으로서 기능을 상실하게 됩니다. 그 문제를 해결하기 위해서 필요한 이미지를 불러와서 붙이는 방식이 아닌 최대한 이미지를 적게 만들기 위해 작업했습니다. 그렇게 해서 스코어보드나 노트가 내려오는 레일의 이미지 모두 배경화면에 붙어있는 하나의 이미지로 구성되어 있으며 실제로 게임을 하면 키 이펙트와 노트만이 움직이고 있습니다. 그렇게 하여 많은 메모리를 확보 할 수 있었습니다. 메모리를 확보한 이후로 신경을 쓴 것은 게임의 알고리즘이었습니다. 제일 먼저 생각하게 된 것은 역시 메모리와 관련된 부분이였습니다. 컴퓨터마다 객체를 생성하는 시간, 로딩하는시간 즉 같은 코드여도 코드를 돌리는 시간이 모두 다르기 때문에 나타나는 차이를 없게	778

		만들고 싶었습니다. 그래서 모든 컴퓨터에서 동일하게 흘러가는 것은 시간이라고 생각하게 되었고 모든 코드를 시간에 관계하여 짜게 되었습니다. 우선 게임의 모든 변수를 선언하고 이미지도 모두 불러온 다음에 현제 시간을 저장해 둡니다. 그 후 계속 시간을 체크하여 프로그램 시작 경과시간을 구합니다. 이 경과시간을 기준으로 모든게 굴러가게 됩니다. 사용자가 게임을 실행하자마자 게임을 하는 것이 아니라 여유를 가지고 게임을 하도록 하는 게임시작 3초의 딜레이, 노트가 생성되는 타이밍 제어, 시간에 따른 노트 객체의 좌표를 다루게 됩니다. 이로 인해 어떤 컴퓨터에서도 같은 시간에 같은 노트를 같은 위치에서 만날 수 있게 됩니다. 그 다음은 알고리즘 단계 입니다. 노트 객체를 만들어서 일정 부분 내에서 충돌검사를 해야 한다고 생각 했지만 막상 그것을 구현 한다고 생각하니까 각 객체마다 리스너를 달 수도 없고 리스너에다가 객체를 확인하는 것 또한 좋은 방법이 아니라고 생각했습니다. 그러다가 생각하게 된 것이 바로 상태 변수들입니다. 마치 cpu에 인터럽트를 위한 플래그 같은 기능을 하는 변수들입니다. 예를 하나 보겠습니다. 키 리스너가 키를 누르면 키 이벤트 상태를 알려주는 변수를 활성화 시킵니다. 그러면 노트 체크를 하는 변수가 노트의 상태를 확인하는 방식입니다. 이렇게 많은 부분에서 상태 변수들이 쓰였는데요 이렇게 해서 각 기능별로 공유하고 있는 변수를 통해 불필요하게 많은 코드를 돌리는게 아니라 필요할때만 필요한 코드를 돌리게 되었습니다. 이렇게 해서 기본 게임을 다 만들었습니다. 기본 게임을 다 만들고 저희의 차별화된 부분인 아이템 모드를 구현하는데 2가지 선택지를 두고 고민하였습니다. 우선 첫 번째 방법은 기본 게임 모듈을 가지고 새로운 아이템 게임클래스를 만드는 것입니다. 두 번째 방법은 기존 게임에 아이템 기능을 넣어놓고 생성자를 통해 아이템 여부를 받아 아이템 모드일때만 아이템 기능을 실행시키는 방법입니다. 저는 2번째 방법을 선택하였습니다. 게임 클래스가 완성이 되었고 실제 플레이될 게임을 만드는 단계가 되었습니다. 각 게임들은 2가지의 고유값을 통해서 구분이 되는데 바로 노래와 노트패턴입니다. 이 둘을 다르게 하여 게임 고유의 특징을 띄게 되고 게임별로 다른 게임이 되게 됩니다. 각 게임 클래스는 자신의 게임에 쓰일 노래와 자신 게임의 노트패턴을 시간으로 가지고 있으며 이것을 게임에 넘겨줘 다른 게임을 호출하게 됩니다.	