

3XA3 Module Interface Specification

Rhythm Master

Team #16, Rhythm Masters
Almen Ng, nga18
David Yao, yaod9
Veerash Palanichamy, palanicv

March 18, 2021

Contents

Button Controller Module	11
Template Module	11
Uses	11
Syntax	11
Exported Constants	11
Exported Types	11
Exported Access Programs	11
Semantics	11
State Variables	11
Environment Variables	11
State Invariant	12
Assumptions	12
Access Routine Semantics	12
Local Functions/Constants	13
 Change Scene Module	 14
Template Module	14
Uses	14
Syntax	14
Exported Constants	14
Exported Types	14
Exported Access Programs	14
Semantics	14
State Variables	14
Environment Variables	14
State Invariant	14
Assumptions	14
Access Routine Semantics	14
Local Functions/Constants	15
 Collision Detector Module	 16
Template Module	16
Uses	16
Syntax	16
Exported Constants	16
Exported Types	16
Exported Access Programs	16
Semantics	16
State Variables	16
Environment Variables	16
State Invariant	16
Assumptions	16
Access Routine Semantics	17

Local Functions/Constants	17
Complete Screen Module	18
Template Module	18
Uses	18
Syntax	18
Exported Constants	18
Exported Types	18
Exported Access Programs	18
Semantics	18
State Variables	18
Environment Variables	18
State Invariant	18
Assumptions	19
Access Routine Semantics	19
Local Functions/Constants	19
Detect Key	20
Template Module	20
Uses	20
Syntax	20
Exported Constants	20
Exported Types	20
Exported Access Programs	20
Semantics	20
State Variables	20
Environment Variables	20
State Invariant	20
Assumptions	20
Access Routine Semantics	21
Local Functions/Constants	21
Effects Module	22
Template Module	22
Uses	22
Syntax	22
Exported Constants	22
Exported Types	22
Exported Access Programs	22
Semantics	22
State Variables	22
Environment Variables	22
State Invariant	22
Assumptions	22
Access Routine Semantics	23

Local Functions/Constants	23
Effects Manager	24
Template Module	24
Uses	24
Syntax	24
Exported Constants	24
Exported Types	24
Exported Access Programs	24
Semantics	24
State Variables	24
Environment Variables	24
State Invariant	24
Assumptions	25
Access Routine Semantics	25
Local Functions/Constants	25
FileIO Module	26
Template Module	26
Uses	26
Syntax	26
Exported Constants	26
Exported Types	26
Exported Access Programs	26
Semantics	26
State Variables	26
Environment Variables	26
State Invariant	26
Assumptions	27
Access Routine Semantics	27
Local Functions/Constants	27
Game Manager Module	28
Template Module	28
Uses	28
Syntax	28
Exported Constants	28
Exported Types	28
Exported Access Programs	28
Semantics	28
State Variables	28
Environment Variables	29
State Invariant	29
Assumptions	29
Access Routine Semantics	29

Local Functions/Constants	30
Leaderboard Entry	32
Template Module	32
Uses	32
Syntax	32
Exported Constants	32
Exported Types	32
Exported Access Programs	32
Semantics	32
State Variables	32
Environment Variables	32
State Invariant	32
Assumptions	32
Access Routine Semantics	32
Local Functions/Constants	33
Load Settings Data Module	34
Template Module	34
Uses	34
Syntax	34
Exported Constants	34
Exported Types	34
Exported Access Programs	34
Semantics	34
State Variables	34
Environment Variables	34
State Invariant	34
Assumptions	34
Access Routine Semantics	35
Local Functions/Constants	35
Main Menu	36
Template Module	36
Uses	36
Syntax	36
Exported Constants	36
Exported Types	36
Exported Access Programs	36
Semantics	36
State Variables	36
Environment Variables	36
State Invariant	37
Assumptions	37
Access Routine Semantics	37

Local Functions/Constants	37
Note Object	38
Template Module	38
Uses	38
Syntax	38
Exported Constants	38
Exported Types	38
Exported Access Programs	38
Semantics	38
State Variables	38
Environment Variables	38
State Invariant	38
Assumptions	38
Access Routine Semantics	38
Local Functions/Constants	39
Note Scroller	40
Module	40
Uses	40
Syntax	40
Exported Constants	40
Exported Types	40
Exported Access Programs	40
Semantics	40
State Variables	40
Environment Variables	40
State Invariant	40
Assumptions	40
Access Routine Semantics	40
Local Functions/Constants	41
Note Spawner	42
Template Module	42
Uses	42
Syntax	42
Exported Constants	42
Exported Types	42
Exported Access Programs	42
Semantics	42
State Variables	42
Environment Variables	42
State Invariant	42
Assumptions	43
Access Routine Semantics	43

Local Functions/Constants	43
Pause Menu	44
Template Module	44
Uses	44
Syntax	44
Exported Constants	44
Exported Types	44
Exported Access Programs	44
Semantics	44
State Variables	44
Environment Variables	44
State Invariant	44
Assumptions	44
Access Routine Semantics	45
Local Functions/Constants	45
PopulateLeaderboard Module	46
Template Module	46
Uses	46
Syntax	46
Exported Constants	46
Exported Types	46
Exported Access Programs	46
Semantics	46
State Variables	46
Environment Variables	46
State Invariant	46
Assumptions	46
Access Routine Semantics	47
Local Functions/Constants	47
Quit Game Module	48
Template Module	48
Uses	48
Syntax	48
Exported Constants	48
Exported Types	48
Exported Access Programs	48
Semantics	48
State Variables	48
Environment Variables	48
State Invariant	48
Assumptions	48
Access Routine Semantics	48

Local Functions/Constants	49
Score Calculator Module	50
Template Module	50
Uses	50
Syntax	50
Exported Constants	50
Exported Types	50
Exported Access Programs	50
Semantics	50
State Variables	50
Environment Variables	51
State Invariant	51
Assumptions	51
Access Routine Semantics	51
Local Functions/Constants	52
Settings Menu Module	53
Template Module	53
Uses	53
Syntax	53
Exported Constants	53
Exported Types	53
Exported Access Programs	53
Semantics	53
State Variables	53
Environment Variables	53
State Invariant	54
Assumptions	54
Access Routine Semantics	54
Local Functions/Constants	55
Leaderboard Calculator	56
Template Module	56
Uses	56
Syntax	56
Exported Constants	56
Exported Types	56
Exported Access Programs	56
Semantics	56
State Variables	56
Environment Variables	56
State Invariant	56
Assumptions	56
Access Routine Semantics	57

Local Functions/Constants	57
Instructions	58
Template Module	58
Uses	58
Syntax	58
Exported Constants	58
Exported Types	58
Exported Access Programs	58
Semantics	58
State Variables	58
Environment Variables	58
State Invariant	58
Assumptions	58
Access Routine Semantics	59
Local Functions/Constants	59

List of Tables

1 Revision History	10
--------------------------------	----

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 1, 2021	0.1	Initial Document
March 13, 2021	0.2	Wrote MIS for 10 modules
March 17, 2021	0.5	Finished MIS for all modules
April 12, 2021	1.0	Revised MIS for final product

Button **Controller** Module

Template Module

Button**Controller** inherits UnityEngine.MonoBehaviour

Uses

UnityEngine, System.Collections

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
CanPressLong			
CantPressLong			
CanPress			
CantPress			

Semantics

State Variables

keyToPress: KeyCode

canBePressed: bool

canBePressedLong: bool

longBeingPressed: bool

colour: string

code: int

finishedLong: bool

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

Start():

- transition:
keyToPress := (KeyCode)PlayerPrefs.GetInt(colour,code)
- exception: None

Update():

- transition:
//move the button up when the button is pressed, and back when it is released

UnityEngine.Input.GetKeyDown(keyToPress) \Rightarrow *button.z := button.z + distance*
UnityEngine.Input.GetKeyUp(keyToPress) \Rightarrow *button.z := button.z - distance*
- exception: None

CanPressLong():

- transition:
canBePressedLong := true
- exception: None

CantPressLong():

- transition:
longBeingPressed \Rightarrow finishedLong := true
canBePressedLong := false
- exception: None

CanPress():

- transition:
canBePressed := true
- exception: None

CantPress():

- transition:
canBePressed := false
- exception: None

Local Functions/Constants

N/A

Change Scene Module

Template Module

ChangeScene inherits UnityEngine.MonoBehaviour

Uses

System.Collections, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
BtnChangeScene	string		

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

BtnChangeScene(scene_name):

- transition: None
- exception: None

Local Functions/Constants

N/A

Collision Detector Module

Template Module

Collision Detector inherits UnityEngine.MonoBehaviour

Uses

System.Collections, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
NoteHit			
NoteMissed			
LongNoteHit			
LongNoteClicked			

Semantics

State Variables

hitLast: bool

Environment Variables

collider: Collider

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

NoteHit()

- Transition: Checks distance between centre of note and collider. Spawns effect and calls NormalHit(), GoodHit(), or PerfectHit() from GameManager.
- Exception: None

NoteMissed()

- Transition: Spawns effect and calls NoteMissed() from GameManager.
- Exception: None

LongNoteClicked()

- Transition: Calls LongHit() from GameManager.
- Exception: None

LongNoteClicked()

- Transition: hitLast := true. Calls LongClicked() from GameManager.
- Exception: None

Local Functions/Constants

OnTriggerEnter(noteCollider)

- Transition: collider := noteCollider. Checks if *noteCollider* is an "Activator". If it is, *canBePressed* will be set to true.
- Exception: None

OnTriggerExit(noteCollider):

- Transition: collider := null. Checks if *noteCollider* is an "Activator". If it is, *canBePressed* will be set to false.
- Exception: None

Complete Screen Module

Template Module

CompleteScreen inherits UnityEngine.MonoBehaviour

Uses

System.Collections, Systems.Collections.Generic; UnityEngine, TMPro

Syntax

Exported Constants

N/A

Exported Types

CompleteScreen = this

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayCompleteScreen	\mathbb{R} , \mathbb{R} , \mathbb{R} , \mathbb{R}		
SaveFinalScore			

Semantics

State Variables

finalScoreSave: \mathbb{R}

Environment Variables

completeScreen: GameObject of complete screen screen
normalHitText: TextMeshProUGUI field for number of normal hits
goodHitText: TextMeshProUGUI field for number of good hits
perfectHitText: TextMeshProUGUI field for number of perfect hits
missedHitText: TextMeshProUGUI field for number of missed hits
accuracyText: TextMeshProUGUI field for accuracy
finalScoreText: TextMeshProUGUI field for final score
userNameInput: TextMeshProUGUI field for username input
submitButton: GameObject field for submit button
instance: CompleteScreen

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

DisplayCompleteScreen(normalCount, goodCount, perfectCount, missedCount, accuracy, finalScore)

- transition: finalScoreSave := finalScore
completeScreen.activeInHierarchy \Rightarrow All texts := textCount.ToString()
- exception: None

SaveFinalScore()

- transition: Add entry for current game to the leaderboard.
- exception: None

Local Functions/Constants

N/A

Detect Key

Template Module

DetectKey inherits UnityEngine.MonoBehaviour

Uses

System, System.Collections, System.Collections.Generic, UnityEngine, UnityEngine.UI

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions

Semantics

State Variables

keyColour: string

keyPressed: string

Environment Variables

key: GameObject that defines the key

ui: GameObject that controls the UI aspect of rebinding the keys

settingsButtonText: Text field that shows the user what key the button is binded to

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

N/A

Local Functions/Constants

Update()

- transition: $\text{Input.anyKeyDown} \Rightarrow \text{key.keyToPress}$ is set to keycode.
- exception: None

Effects Module

Template Module

Effects inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			

Semantics

State Variables

~~$lifetime: \mathbb{R}$~~
N/A

Environment Variables

~~$missEffect: GameObject$~~
 ~~$okEffect: GameObject$~~
 ~~$goodEffect: GameObject$~~
 ~~$perfEffect: GameObject$~~
N/A

State Invariant

~~$lifetime \succ 0$~~
N/A

Assumptions

N/A

Access Routine Semantics

~~Start():~~

- ~~• transition: Display either *missEffect*, *okEffect*, *goodEffect*, *perfEffect*.~~
- ~~• exception: None~~

~~Update():~~

- ~~• transition: Effect is deleted after *lifetime* has passed.~~
- ~~• exception: None~~

Local Functions/Constants

~~N/A~~

~~*lifetime*: \mathbb{N}~~

~~*lifetime* $\equiv 1$~~

~~Update():~~

- ~~• transition: Effect is deleted after *lifetime* has passed.~~
- ~~• exception: None~~

Effects Manager

Template Module

Effects Manager inherits `UnityEngine.MonoBehaviour`

Uses

`System.Collections`, `System.Collections.Generic`, `UnityEngine`

Syntax

Exported Constants

N/A

Exported Types

`EffectsManager` = this

Exported Access Programs

Routine name	In	Out	Exceptions
<code>SpawnNormalEffect</code>	<code>R, R, R</code>		
<code>SpawnGoodEffect</code>	<code>R, R, R</code>		
<code>SpawnPerfectEffect</code>	<code>R, R, R</code>		
<code>SpawnMissedEffect</code>	<code>R, R, R</code>		

Semantics

State Variables

N/A

Environment Variables

instance: `EffectsManager`

normalEffect: `GameObject` of normal effect

goodEffect: `GameObject` of good effect

perfEffect: `GameObject` of perfect effect

missEffect: `GameObject` of miss effect

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

SpawnNormalEffect(x, y, z)

- transition: Instantiate *normalEffect* at the position(x coordinate, y coordinate, z coordinate) defined by the parameters x, y , & z respectively

SpawnGoodEffect(x, y, z)

- transition: Instantiate *goodEffect* at the position(x coordinate, y coordinate, z coordinate) defined by the parameters x, y , & z respectively

SpawnPerfectEffect(x, y, z)

- transition: Instantiate *perfectEffect* at the position(x coordinate, y coordinate, z coordinate) defined by the parameters x, y , & z respectively

SpawnMissedEffect(x, y, z)

- transition: Instantiate *missedEffect* at the position(x coordinate, y coordinate, z coordinate) defined by the parameters x, y , & z respectively

Local Functions/Constants

Start():

- transition: instance := this
- exception: None

FileIO Module

Template Module

SaveFileHandler

Uses

System.Collections, System.Collections.Generic, UnityEngine, UnityEngine.PlayerPrefs

Syntax

Exported Constants

N/A

Exported Types

N/A FileIO = this

Exported Access Programs

Routine name	In	Out	Exceptions
writeUserData	N, String		
getAllData		seq of < N, String >	
ReadFile	String, Char	seq of < String >	
GetLeadboardList		seq of < LeaderboardEntry >	
AddEntryToLeaderboard	LeaderboardEntry		

Semantics

State Variables

~~filename: String~~

N/A

Environment Variables

~~file: Local file to which user data will be written and read from.~~

~~instance: FileIO~~

State Invariant

N/A

Assumptions

N/A

It is assumed that the file in the path exists, is in the correct format, and the layout within the file is consistent with the parsing of the file.

Access Routine Semantics

`writeUserData(score, name):`

- **transition:** $file := file || \langle \text{name score} \rangle$
- **exception:** None

`getAllData():`

- **output:** $out := \langle i : \mathbb{N} | 0 \leq i < |file| : \langle file[i][0], file[i][1] \rangle \rangle$
//file[x][y] means line x word number y in that file
- **exception:** None

`ReadFile(pathName, separator):`

- **output:** array of string, signifying the separated values in the file separated by *separator*, located in the file path, *pathName*.
- **exception:** none

`GetLeaderboardList():`

- **output:** Returns a list of `LeaderboardEntry` objects that represent the leaderboard.
- **exception:** none

`AddEntryToLeaderboard(entry):`

- **transition:** Adds *entry*, a `LeaderboardEntry` object to the list of `LeaderboardEntry` objects representing the leaderboard
- **exception:** none

Local Functions/Constants

N/A

Game Manager Module

Template Module

GameManager inherits UnityEngine.MonoBehaviour

Uses

System.Collections, Systems.Collections.Generic, UnityEngine, UnityEngine.UI, TMPro

Syntax

Exported Constants

N/A

Exported Types

GameManager = this

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
StartMusic			
NormalHit			
GoodHit			
PerfectHit			
LongHit			
NoteMissed			
LongClicked			
SetMultiplier			
SetScore			

Semantics

State Variables

instance: GameManager
~~*startPlaying*~~: ~~boolean~~
~~*music*~~: song: AudioSource
guitar: AudioSource
currentScore: \mathbb{Z}
currentMultiplier: \mathbb{Z}
songStarted: bool = false
totalNotes: \mathbb{R}

$normalHits: \mathbb{R}$
 $goodHits: \mathbb{R}$
 $perfectHits: \mathbb{R}$
 $missedHits: \mathbb{R}$
 $accuracy: \mathbb{R}$ $noteList: \text{seq of Strings}$ $defaultVolume: \mathbb{R}$

Environment Variables

~~$theNS$: NoteScroller controlling the movement of notes along the game screen.~~
 $scoreText$: TextMeshProUGUI field for current score
 $multiText$: TextMeshProUGUI field for current multiplier

State Invariant

$totalNotes, normalHits, goodHits, perfectHits, missedHits, accuracy, defaultVolume \geq 0$

Assumptions

N/A

Access Routine Semantics

$Start()$:

- ~~$transition: instance := this$~~
- ~~$exception: None$~~

$Update()$:

- ~~$transition$: Initiates both music playing and note scrolling when $startPlaying = true$.~~
- ~~$exception: None$~~

$StartMusic()$:

- $transition$: Play *song* and *guitar* and mute *guitar* when called
- $exception$: None

$NormalHit()$:

- $transition$: Set $guitar.mute = false$ if $guitar.mute = true$, increase $normalHits$ by calling $NormalHit()$ from the ScoreCalculator module, and call $NoteHit()$
- $exception$: None

$GoodHit()$:

- $transition$: Set $guitar.mute = false$ if $guitar.mute = true$, increase $goodHits$ by calling $GoodHit()$ from the ScoreCalculator module, and call $NoteHit()$

- exception: None

PerfectHit():

- transition: Set *guitar.mute = false* if *guitar.mute = true*, increase *perfectHits* by calling PerfectHit() from the ScoreCalculator module, and call NoteHit()
- exception: None

LongHit():

- transition: Increase *normalHits* by calling LongHit() from the ScoreCalculator module, and call NoteHit()
- exception: None

NoteMissed():

- transition: Set *guitar.mute = true* if *guitar.mute = false*, increase *missedHits* by calling NoteMissed() from the ScoreCalculator module, and call NoteHit()
- exception: None

LongClicked():

- transition: Set *guitar.mute = false* if *guitar.mute = true*
- exception: None

SetMultiplier(mult):

- transition: Set *currentMultiplier = mult*
- exception: None

Local Functions/Constants

Start():

- transition: *instance := this*, *currentScore := 0*, *currentMultiplier := 1*, call ReadFile from FileIO and store the output in *noteList*, set the volume to *defaultVolume*
- exception: None

Update():

- transition: Call GameComplete() if the the PauseMenu is not displayed, the song is not playing, and the song has started
- exception: None

NoteHit():

- transition: Set *multiText* to *currentMultiplier* and *scoreText* to *currentScore*
- exception: None

GameComplete():

- transition: Call `Accuracycalculation()` from `ScoreCalculator` and store the returned value in *accuracy*, call `DisplayCompleteScreen()` from the `CompleteScreen` module with parameters *normalHits*, *goodHits*, *perfectHits*, *missedHits*, and *accuracy* to display the Complete Screen.
- exception: None

Leaderboard Entry

Template Module

Leaderboard Entry

Uses

System.Collections, System.Collections.Generic

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions

Semantics

State Variables

name: String

score: \mathbb{Z}

date: String

Environment Variables

N/A

State Invariant

score ≥ 0

Assumptions

N/A

Access Routine Semantics

N/A

Local Functions/Constants

N/A

Load Settings Data Module

Template Module

SettingsData

Uses

UnityEngine.PlayerPrefs

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
setVolumeLevel	\mathbb{R}		
setKeyBinds	seq of \mathbb{N}		
getKeyBinds		seq of \mathbb{N}	
getVolumeLevel		\mathbb{R}	

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

~~setVolumeLevel(v):~~

- ~~transition: UnityEngine.PlayerPrefs.SetFloat(“volume”, v)~~
- ~~exception: None~~

~~setKeyBinds(s):~~

- ~~transition: $\forall (i : \mathbb{N} | 0 \leq i < |s| : \text{UnityEngine.PlayerPrefs.SetInt}(\text{nameMap}[i], s[i]))$~~
- ~~exception: None~~

~~getKeyBinds():~~

- ~~output: $\text{out} := \langle i : \mathbb{N} | 0 \leq i < |s| : \text{UnityEngine.PlayerPrefs.GetInt}(\text{nameMap}[i], s[i]) \rangle$~~
- ~~exception: None~~

~~getVolumeLevel():~~

- ~~output: $\text{out} := \text{UnityEngine.PlayerPrefs.GetFloat}(\text{“volume”}, v)$~~
- ~~exception: None~~

N/A

Local Functions/Constants

~~nameMap: String [“GreenB”, “RedB”, “YellowB”, “BlueB”, “PinkB”]~~

~~greenCode: \mathbb{Z}~~

~~greenCode \equiv 97~~

~~redCode: \mathbb{Z}~~

~~redCode \equiv 115~~

~~yellowCode: \mathbb{Z}~~

~~yellowCode \equiv 100~~

~~blueCode: \mathbb{Z}~~

~~blueCode \equiv 102~~

~~pinkCode: \mathbb{Z}~~

~~pinkCode \equiv 118~~

~~Start():~~

- ~~transition: Set *greenButtonSettings*, *redbuttonSettings*, *yellowButtonSettings*, *blueButtonSettings*, *pinkButtonSettings* to *greenCode*, *redCode*, *yellowCode*, *blueCode*, and *pinkCode* respectively~~
- ~~exception: None~~

Main Menu

Template Module

MainMenu inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine, UnityEngine.SceneManagement

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
PlayGame			
QuitGame			
NavigateSettings			

Semantics

State Variables

N/A

Environment Variables

gameUI: Unity Scene for the game play

settingsUI: Unity Scene that enables editing of the settings

startGameButton: Button that will trigger the action of navigating to the *gameUI*, specifically calling the **PlayGame()** function.

quitGameButton: Button that will trigger the action of quitting the game, specifically calling the **QuitGame()** function.

navigateSettingsButton: Button that will trigger the action of navigating to the **Settings Menu**, specifically calling the **NavigateSettings()** function.

State Invariant

Assumptions

~~The environment variables are initialized manually through the Unity interface.~~

Access Routine Semantics

PlayGame():

- Transition: Navigates to *gameUI* once *startGameButton* is pressed.
- Exception: None

~~QuitGame():~~

- ~~• Transition: Quits the application once *quitGameButton* is pressed.~~
- ~~• Exception: None~~

~~NavigateSettings():~~

- ~~• Transition: Navigates to *settingsUI* once *navigateSettingsButton* is pressed.~~
- ~~• Exception: None~~

Local Functions/Constants

N/A

Note Object

Template Module

NoteObject inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions

Semantics

State Variables

N/A

Environment Variables

keyToPress: KeyCode *key*: GameObject

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

N/A

Local Functions/Constants

N/A

Note Scroller

Module

NoteScroller inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
<i>Start</i>			
<i>Update</i>			

Semantics

State Variables

hasStarted: \mathbb{B}

beatTempo: \mathbb{R}

Environment Variables

N/A

State Invariant

N/A

Assumptions

beatTempo is set externally through the Unity interface.

Access Routine Semantics

N/A

Local Functions/Constants

Start():

- Transition: Set note collider size. Remove note after it leaves the screen.
- Exception: None

Update():

- Transition: Checks *hasStarted*. If True, move the GameObject based on *beatTempo*.
- Exception: None

Note Spawner

Template Module

NoteSpawner inherits UnityEngine.MonoBehaviour

Uses

GameManager, System, System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
<i>Start</i> <i>SpawnNote</i>			

Semantics

State Variables

BPM := \mathbb{R}
floatIndex: \mathbb{R}
nextIndex: \mathbb{Z}
index: \mathbb{Z}
noteList: string[]

Environment Variables

notes: GameObject[]
longNotes: GameObject[]
keys: GameObject[]

State Invariant

N/A

Assumptions

noteList is populated by GameManager and FileIO.

Access Routine Semantics

SpawnNote():

- transition:
Instantiates a single *note* and adds it to the game.
- exception: None

Local Functions/Constants

numNotes := 5

spawnDelay := 2

minute spawnStartTime := 60

Start():

- transition:
BPM := 175, *nextIndex* := 0. Repeatedly calls SpawnNote, starting after *spawnStartTime* and repeating every *minute/BPM*.
- exception: None

SpawnNote():

- transition:
Parse the next entry in *noteList*, representing the notes to spawn on the current beat.
Spawn the correct notes.
nextIndex := *nextIndex* + 1
- exception: None

SpawnLongNote(*len*, *index*):

- transition:
Spawn a long note of length *len*, from the given *index* in *noteList*.
- exception: None

Pause Menu

Template Module

PauseMenu inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine
~~UnityEngine.Input, UnityEngine.GameObject~~

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Settings Update			
Resume			
Pause			

Semantics

State Variables

Paused: \mathbb{B}

~~*SettingsShown*: \mathbb{B}~~

Environment Variables

pauseMenuUI: GameObject that shows the pause menu

~~*settingsMenuUI*: GameObject that shows the settings menu~~

State Invariant

None

Assumptions

The environment variables are initialized manually through the Unity interface.

Access Routine Semantics

Resume():

- Transition: Set *pauseMenuUI* to false, unfreeze time, and set *gamePaused* to False.
- Exception: None

Pause():

- Transition: Set *pauseMenuUI* to active, freeze time, and set *gamePaused* to True.
- Exception: None

Settings():

- Transition: *SettingsShown* := true
Set *settingsMenuUI* to active.
- Exception: None

Local Functions/Constants

Update():

- Transition: Checks if "esc" button has been pressed and *gamePaused* is True. If both are True, call **Resume()**, otherwise, if only "esc" button has been pressed, call **Pause()**.
- Exception: None

PopulateLeaderboard Module

Template Module

PopulateLeaderboard inherits UnityEngine.MonoBehaviour

Uses

FileIO, System.Collections, System.Collections.Generic, UnityEngine, UnityEngine.UI

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			

Semantics

State Variables

~~saveFile: SaveFileHandler~~ ~~playerList: seq of (String, N)~~

Environment Variables

~~table: table that displays the player rank, name and score. This table's data can be indexed such as table[row][column] where 1st columns is the rank, the 2nd the name and lastly the score. The table will also have a heading of rank, player and score.~~

~~textTemplate: Text. Used to display text.~~

State Invariant

N/A

Assumptions

~~saveFile should have been assigned referenced to a SaveFileHandler component~~

Access Routine Semantics

Local Functions/Constants

Start():

- transition:
Read leaderboard list from FileIO, sort it from highest to lowest score, and output each entry to the screen.
- exception: None

Quit Game Module

Template Module

QuitGame inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
ExitGame			

Semantics

State Variables

N/A

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

ExitGame():

- Transition: Quit the application.
- Exception: None

Local Functions/Constants

N/A

Score Calculator Module

Template Module

ScoreCalculator inherits UnityEngine.MonoBehaviour

Uses

System.Collections, System.Collections.Generic, UnityEngine

Syntax

Exported Constants

scorePerNote := 100
scorePerGoodNote := 125
scorePerPerfectNote := 150
scorePerLongNote := 200

Exported Types

ScoreCalculator = this

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
NoteHit			
NormalHit	Z, Z	R	
GoodHit	Z, Z	R	
PerfectHit	Z, Z	R	
NoteMissed		R	
LongHit	Z, Z	R	
AccuracyCalculation	R, R, R, R	R	

Semantics

State Variables

currentMultiplier: \mathbb{N}
multiplierTracker: \mathbb{N}
multiplierThresholds: seq of $\langle \mathbb{N} \rangle$
~~*totalNotes*: \mathbb{N}~~
~~*normalHits*: \mathbb{N}~~
~~*goodHits*: \mathbb{N}~~
~~*perfectHits*: \mathbb{N}~~
~~*missedHits*: \mathbb{N}~~

~~*currentScore*: \mathbb{N}~~
~~*totalNotes*: \mathbb{N}~~

Environment Variables

instance: ScoreCalculator

State Invariant

N/A

Assumptions

multiplierThresholds are set manually within the Unity interface.

Access Routine Semantics

NormalHit(*currentScore*, *currentMultiplier*):

- Transition: *GameManager.instance.SetScore*
(*currentScore* + *scorePerNote* * *currentMultiplier*)
Calls *NoteHit(currentMultiplier)*.
- Output: *out* := 1
- Exception: None

GoodHit(*currentScore*, *currentMultiplier*):

- Transition: *GameManager.instance.SetScore*
(*currentScore* + *scorePerGoodNote* * *currentMultiplier*)
Calls *NoteHit(currentMultiplier)*.
- Output: *out* := 1
- Exception: None

PerfectHit(*currentScore*, *currentMultiplier*):

- Transition: *GameManager.instance.SetScore*
(*currentScore* + *scorePerPerfectNote* * *currentMultiplier*)
Calls *NoteHit(currentMultiplier)*.
- Output: *out* := 1
- Exception: None

NoteMissed(*currentScore*, *currentMultiplier*):

- Transition: Resets the *currentMultiplier* and *multiplierTracker* to its initial values, 1 and 0 respectively.
- Output: `out := 1`
- Exception: None

`LongHit(currentScore, currentMultiplier):`

- Transition: `GameManager.instance.SetScore`
(`currentScore + scorePerLongNote * currentMultiplier`)
Calls `NoteHit(currentMultiplier)`.
- Output: `out := 1`

`AccuracyCalculation(normalHits, goodHits, perfectHits, totalNotes):`

- Transition: `float totalHits := normalHits + goodHits + perfectHits`
- Output: `out := totalHits / totalNotes * 100`
- Exception: None

Local Functions/Constants

`Start():`

- Transition: `instance := this`
- Exception: None

`NoteHit():`

- Transition: Increments *multiplierTracker* every time a note is hit consecutively and increments the *currentMultiplier* once a threshold based on *multiplierThresholds* is met by comparing *multiplierThreshold[currentMultiplier]* and *multiplierTracker*.
- Exception: None

scorePerNote: \mathbb{N}

scorePerNote $\equiv 100$

scorePerGoodNote: \mathbb{N}

scorePerGoodNote $\equiv 125$

scorePerPerfectNote: \mathbb{N}

scorePerPerfectNote $\equiv 150$

scorePerLongNote: \mathbb{N}

scorePerLongNote $\equiv 200$

Settings Menu Module

Template Module

SettingsMenu inherits UnityEngine.MonoBehaviour

Uses

SettingsData, System.Collections, **System.Collections.Generic**, UnityEngine, **UnityEngine.UI**

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
SetVolume Start			
GreenPressed Update			
RedPressed saveSettings			
YellowPressed loadSettings			
BluePressed			
PinkPressed			

Semantics

State Variables

~~*temp_volume: N*~~
~~*data: SettingsData*~~

Environment Variables

slider: Slider used to adjust the volume. The slider can take any value from 0 to 100. This slider will call the **sliderUpdate()** function when the user changes the slider

~~*uiGreen textfield_green*~~: Text field where the user chooses which keyboard key controls the green button during gameplay

~~*uiRed textfield_red*~~: Text field where the user chooses which keyboard key controls the red button during gameplay

~~*uiYellow textfield-yellow*~~: Text field where the user chooses which keyboard key controls the yellow button during gameplay

~~*uiBlue textfield-blue*~~: Text field where the user chooses which keyboard key controls the blue button during gameplay

~~*uiPink textfield-pink*~~: Text field where the user chooses which keyboard key controls the pink button during gameplay

~~*apply-button*~~: Button that will trigger the action of changing the actual settings of the game and saving the settings, specifically call the **saveSettings()** function.

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

SetVolume():

- transition:
Sets AudioListener volume to slider value and saves to PlayerPrefs.
- exception: None

GreenPressed():

- transition:
Displays uiGreen.
- exception: None

RedPressed():

- transition:
Displays uiRed.
- exception: None

YellowPressed():

- transition:
Displays uiYellow.
- exception: None

BluePressed():

- transition:
Displays uiBlue.
- exception: None

PinkPressed():

- transition:
Displays uiPink.
- exception: None

~~saveSettings():~~

- ~~transition:~~
~~*data.setVolumeLevel(temp_volume)*~~
~~*data.setKeyBinds((value(textfield_green),*~~
~~*value(textfield_red),*~~
~~*value(textfield_yellow),*~~
~~*value(textfield_blue),*~~
~~*value(textfield_pink)))*~~
- ~~exception: None~~

~~sliderUpdate():~~

- ~~transition: *temp_volume := slider_value(slider)*~~
- ~~exception: None~~

Local Functions/Constants

Start():

- transition:
slider := data.getVolumeLevel()
textfield_green := data.getKeyBinds()[0]
textfield_red := data.getKeyBinds()[1]
textfield_yellow := data.getKeyBinds()[2]
textfield_blue := data.getKeyBinds()[3]
textfield_pink := data.getKeyBinds()[4]
- exception: None

value: *textfield* → \mathbb{N}

value \equiv returns the value that the textfields contains

slider_value: *slider* → \mathbb{N}

slider_value \equiv returns the value of a slider

Leaderboard Calculator

Template Module

LeaderboardCalculator inherits UnityEngine.MonoBehaviour

Uses

System.Collections.Generic

Syntax

Exported Constants

N/A

Exported Types

playerList: seq of $\langle String, \mathbb{N} \rangle$

Exported Access Programs

Routine name	In	Out	Exceptions
Sort	seq of $\langle String, \mathbb{N} \rangle$	seq of $\langle String, \mathbb{N} \rangle$	None

Semantics

State Variables

playerList: seq of $\langle String, \mathbb{N} \rangle$

Environment Variables

N/A

State Invariant

N/A

Assumptions

The scores in *playerList* are all from the same game track.

~~Access Routine Semantics~~

~~Sort(vector<pair<string,int>> &playerList):~~

- ~~• transition:~~
~~Values in playerList are sorted in descending order of their int values.~~
- ~~• exception: None~~

~~Local Functions/Constants~~

~~None~~

Instructions

Template Module

Instructions inherits UnityEngine.MonoBehaviour

Uses

UnityEngine.Input, UnityEngine.GameObject

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Toggle			

Semantics

State Variables

N/A

Environment Variables

instructionUI: GameObject showing the instructions
toggleButton: Button to toggle the instruction screen

State Invariant

N/A

Assumptions

N/A

~~Access Routine Semantics~~

~~Toggle():~~

- ~~transition:~~
Displays the instruction screen if it is not currently being displayed, otherwise stops displaying it.
- ~~exception:~~ None

~~Local Functions/Constants~~

~~N/A~~