

3XA3 Module Interface Specification

Rhythm Master

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

March 18, 2021

Contents

Button Handler Module	9
Template Module	9
Uses	9
Syntax	9
Exported Constants	9
Exported Types	9
Exported Access Programs	9
Semantics	9
State Variables	9
Environment Variables	9
State Invariant	9
Assumptions	9
Access Routine Semantics	10
Local Functions/Constants	10
Settings Data 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	11
Assumptions	11
Access Routine Semantics	12
Local Functions/Constants	12
Save File Handler Module	13
Template Module	13
Uses	13
Syntax	13
Exported Constants	13
Exported Types	13
Exported Access Programs	13
Semantics	13
State Variables	13
Environment Variables	13
State Invariant	13
Assumptions	13
Access Routine Semantics	14

Local Functions/Constants	14
Settings Menu Module	15
Template Module	15
Uses	15
Syntax	15
Exported Constants	15
Exported Types	15
Exported Access Programs	15
Semantics	15
State Variables	15
Environment Variables	15
State Invariant	16
Assumptions	16
Access Routine Semantics	16
Local Functions/Constants	17
Leaderboard 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	18
Access Routine Semantics	19
Local Functions/Constants	19
Game Manager	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	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
Leaderboard Calculator	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	24
Access Routine Semantics	25
Local Functions/Constants	25
Instructions	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	26
Access Routine Semantics	27

Local Functions/Constants	27
Note Spawner	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	28
State Invariant	28
Assumptions	28
Access Routine Semantics	29
Local Functions/Constants	29
Pause Menu	30
Template Module	30
Uses	30
Syntax	30
Exported Constants	30
Exported Types	30
Exported Access Programs	30
Semantics	30
State Variables	30
Environment Variables	30
State Invariant	30
Assumptions	30
Access Routine Semantics	31
Local Functions/Constants	31
Main Menu	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	33
Assumptions	33
Access Routine Semantics	33

Local Functions/Constants	33
Collision Detector	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
Note Scroller	36
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	36
Assumptions	36
Access Routine Semantics	37
Local Functions/Constants	37
Score Calculator	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	39
State Invariant	39
Assumptions	39
Access Routine Semantics	39

Local Functions/Constants	40
-------------------------------------	----

List of Tables

1 Revision History	8
------------------------------	---

List of Figures

Table 1: **Revision History**

Date	Version	Notes
March 1, 2021	1.0	Initial Document
March 13, 2021	1.1	Wrote MIS for 10 modules
March 17, 2021	1.2	Finished MIS for all modules

Button Handler Module

Template Module

ButtonHandler 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
Start			
Update			

Semantics

State Variables

keyToPress: KeyCode

distance: \mathbb{R}

Environment Variables

button: Gameplay button that is displayed on screen that can be controlled via the keyboard key that was binded it. It is also a Unity game object.

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

Update():

- transition:

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

$\text{UnityEngine.Input.GetKeyDown}(\text{keyToPress}) \Rightarrow \text{button.z} := \text{button.z} + \text{distance}$

$\text{UnityEngine.Input.GetKeyUp}(\text{keyToPress}) \Rightarrow \text{button.z} := \text{button.z} - \text{distance}$

- exception: None

Local Functions/Constants

N/A

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

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

setVolumeLevel(v):

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

setKeyBinds(s):

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

getKeyBinds():

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

getVolumeLevel():

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

Local Functions/Constants

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

Save File Handler Module

Template Module

SaveFileHandler

Uses

UnityEngine.PlayerPrefs

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
writeUserData	\mathbb{N} , String		
getAllData		seq of $\langle \mathbb{N}, \text{String} \rangle$	

Semantics

State Variables

filename: String

Environment Variables

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

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

writeUserData(score, name):

- transition: $file := file \parallel < \text{name score} >$
- exception: None

getAllData():

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

Local Functions/Constants

N/A

Settings Menu Module

Template Module

SettingsMenu inherits UnityEngine.MonoBehaviour

Uses

SettingsData

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			
saveSettings			
sliderUpdate			

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

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

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

textfield_yellow: Text field where the user chooses which keyboard key controls the yellow button during gameplay

textfield_blue: Text field where the user chooses which keyboard key controls the blue button during gameplay

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

Start():

- transition:
value of *slider* := *data*.getVolumeLevel()
value of *textfield_green* := *data*.getKeyBinds()[0]
value of *textfield_red* := *data*.getKeyBinds()[1]
value of *textfield_yellow* := *data*.getKeyBinds()[2]
value of *textfield_blue* := *data*.getKeyBinds()[3]
value of *textfield_pink* := *data*.getKeyBinds()[4]

- 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

value: $textfield \rightarrow \mathbb{N}$

value \equiv returns the value that the textfields contains

slider_value: $slider \rightarrow \mathbb{N}$

slider_value \equiv returns the value of a slider

Leaderboard Module

Template Module

Leaderboard inherits UnityEngine.MonoBehaviour

Uses

SaveFileHandler, LeaderboardCalculator

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 $\langle String, \mathbb{N} \rangle$

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.

State Invariant

N/A

Assumptions

saveFile should have been assigned referenced to a SaveFileHandler component

Access Routine Semantics

Start():

- transition:
 $\forall(i : \mathbb{N} | 0 \leq i < |s| : table[i][0] = i + 1 \wedge table[i][1] = s[i][1] \wedge table[i][2] = s[i][2])$
where $s = \text{LeaderboardCalculator.Sort}(\text{saveFile.getAllData}())$
- exception: None

Local Functions/Constants

N/A

Game Manager

Template Module

GameManager inherits UnityEngine.MonoBehaviour

Uses

System.Collections, Systems.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

startPlaying: boolean

music: AudioSource

instance: GameManager

Environment Variables

theNS: NoteScroller controlling the movement of notes along the game screen.

State Invariant

N/A

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

Local Functions/Constants

N/A

Effects

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			
Update			

Semantics

State Variables

lifetime: \mathbb{R}
missEffect: GameObject
okEffect: GameObject
goodEffect: GameObject
perfEffect: GameObject

Environment Variables

N/A

State Invariant

lifetime ≥ 0

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

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_{LL} 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

Note Spawner

Template Module

NoteSpawner inherits UnityEngine.MonoBehaviour

Uses

UnityEngine

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start SpawnNote			

Semantics

State Variables

spawnDelay: \mathbb{R} *spawnStartTime*: \mathbb{R} *note*: GameObject

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

Start():

- transition:
Repeatedly calls SpawnNote, starting after *spawnStartTime* and repeating every *spawnDelay*.
- exception: None

SpawnNote():

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

Local Functions/Constants

N/A

Pause Menu

Template Module

PauseMenu 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
Update			
Resume			
Pause			

Semantics

State Variables

gamePaused: \mathbb{B}

Environment Variables

pauseMenuUI: GameObject that shows the pause menu

State Invariant

None

Assumptions

The environment variables are initialized manually through the Unity interface.

Access Routine Semantics

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

Resume():

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

Pause():

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

Local Functions/Constants

N/A

Main Menu

Template Module

MainMenu inherits UnityEngine.MonoBehaviour

Uses

UnityEngine.Input, UnityEngine.GameObject, 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

Collision Detector

Template Module

Collision Detector inherits `UnityEngine.MonoBehavior`

Uses

`UnityEngine.GameObject`, `UnityEngine.Collider`

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
<code>OnTriggerEnter</code>	<code>Collider</code>		
<code>OnTriggerExit</code>	<code>Collider</code>		

Semantics

State Variables

canBePressed: \mathbb{B}

Environment Variables

noteCollider: A `ColliderObject` representing a note

State Invariant

N/A

Assumptions

The environment variables are initialized manually through the Unity interface.

Access Routine Semantics

OnTriggerEnter(noteCollider)

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

OnTriggerExit(noteCollider):

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

Local Functions/Constants

N/A

Note Scroller

Module

N/A

Uses

UnityEngine.GameObject

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
Update			

Semantics

State Variables

hasStarted: \mathbb{B}

Environment Variables

N/A

State Invariant

N/A

Assumptions

N/A

Access Routine Semantics

Start():

- Transition: Sets *beatTempo*
- Exception: None

Update():

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

Local Functions/Constants

beatTempo: \mathbb{N}

beatTempo \equiv 126

Score Calculator

Template Module

ScoreCalculator

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
Start			
NoteHit			
NormalHit			
GoodHit			
PerfectHit			
NoteMissed			

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

N/A

State Invariant

N/A

Assumptions

multiplierThresholds are set manually within the Unity interface.

Access Routine Semantics

Start():

- Transition: Initializes *currentScore* to 0, *currentMultiplier* to 1, and *multiplierTracker* to 0. Sets *totalNotes* to the number of Note GameObjects.
- 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

NormalHit():

- Transition: Updates *currentScore* by adding *scorePerNote* multiplied by *currentMultiplier*. Increments *normalHits* each time this function is called.
- Exception: None

GoodHit():

- Transition: Updates *currentScore* by adding *scorePerGoodNote* multiplied by *currentMultiplier*. Increments *GoodHits* each time this function is called.
- Exception: None

PerfectHit():

- Transition: Updates *currentScore* by adding *scorePerPerfectNote* multiplied by *currentMultiplier*. Increments *GoodHits* each time this function is called.
- Exception: None

NoteMissed():

- Transition: Resets the *CurrentMultiplier* and *multiplierTracker* to its initial values, 1 and 0 respectively. Increments *missedHits* by one each this this function is called.
- Exception: None

Local Functions/Constants

scorePerNote: \mathbb{N}

scorePerNote $\equiv 100$

scorePerGoodNote: \mathbb{N}

scorePerGoodNote $\equiv 125$

scorePerPerfectNote: \mathbb{N}

scorePerPerfectNote $\equiv 150$