3XA3 Module Interface Specification
Rhythm Master

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

March 18, 2021

Contents

List of Tables

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

  UnityEngine.Input.GetKeyDown(keyToPress) $\Rightarrow$ *button.z* := *button.z* + *distance*
  UnityEngine.Input.GetKeyUp(keyToPress) $\Rightarrow$ *button.z* := *button.z* − *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: $\forall(i : \mathbb{N} | 0 \leq i < |s| : \text{UnityEngine.PlayerPrefs.SetInt}(nameMap[i], s[i]))$
- exception: None

getKeyBinds():

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

getVolumeLevel():

- output: *out*:=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 $\langle$ $\mathbb{N}$, 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* || < name score >
- 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

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: $\mathbb{N}$
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 $s =$ LeaderboardCalculator.Sort($saveFile$.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

$lifetime \geq 0$

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¿¿ 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
OnTriggerEnter	Collider		
OnTriggerExit	Collider		

Semantics

State Variables

canBePressed: $\mathbb{B}$

Environment Variables

noteCollider: A ColliderObject repersenting 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 $< \mathbb{N} >$
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

$scorePerNote$: $\mathbb{N}$
$scorePerNote \equiv 100$

$scorePerGoodNote$: $\mathbb{N}$
$scorePerGoodNote \equiv 125$

$scorePerPerfectNote$: $\mathbb{N}$
$scorePerPerfectNote \equiv 150$