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	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

ButtonController 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 ⇒ 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: Input.anyKeyDown ⇒ *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

~~$lifetime > 0$~~
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
SpawnNormalEffect	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$		
SpawnGoodEffect	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$		
SpawnPerfectEffect	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$		
SpawnMissedtEffect	$\mathbb{R}$, $\mathbb{R}$, $\mathbb{R}$		

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* || < 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~~

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*: 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 ≥ 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: $\forall(i : \mathbb{N} | 0 \leq i < |s| : \text{UnityEngine.PlayerPrefs.SetInt}(nameMap[i], s[i]))$~~
- ~~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~~

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
~~Start~~			
~~Update~~			

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
~~Start~~ ~~SpawnNote~~			

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

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.~~

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	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{R}$	
GoodHit	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{R}$	
PerfectHit	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{R}$	
NoteMissed		$\mathbb{R}$	
LongHit	$\mathbb{Z}, \mathbb{Z}$	$\mathbb{R}$	
AccuracyCalculation	$\mathbb{R}, \mathbb{R}, \mathbb{R}, \mathbb{R}$	$\mathbb{R}$	

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*: ℕ~~
~~*totalNotes*: 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* $\rightarrow \mathbb{N}$
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 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~~