SOFTWARE DESIGN DOCUMENT

for

Virtual tour based serious Game

July 18, 2018

Prepared by
Inderpreet Singh Chera-160101035
Shubhendu Patidar-160101068
Siddharth Sharma-160101071

IIT GUWAHATI

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to give a detailed description of the design for the Virtual tour based serious game. This includes use case models, sequence diagrams, class diagrams and other supporting information. It will provide us with complete understanding of what is to be built and how it is expected to be built. This document is primarily intended to as a reference for developing the first version of the system for the development team.

1.2 Project Scope

This software is meant to serve as a newcomer's guide for familiarizing the user to the map and layout of the CSE department, IITG. Our goal is to develop a user friendly interface which promotes the memorization of all the pathways of CSE department in a fun way. The user can visit CSE department, IITG even if they are not physically available in IITG. Even physically challenged person who are unable to go on 1st and 2nd floor due to lack of lifts in the department can visit through the game.

1.3 References

1. Software Requirement Specification for virtual based serious game.
2. Share Latex learn documentation https://www.sharelatex.com/learn
3. Stack overflow forum https://stackoverflow.com/
4. Doc on GameMaker https://docs.yoyogames.com/
5. Stack Exchange Forum https://gamedev.stackexchange.com

2 Use Cases

Use Case model survey

2.1 Actor

2.1.1 Player

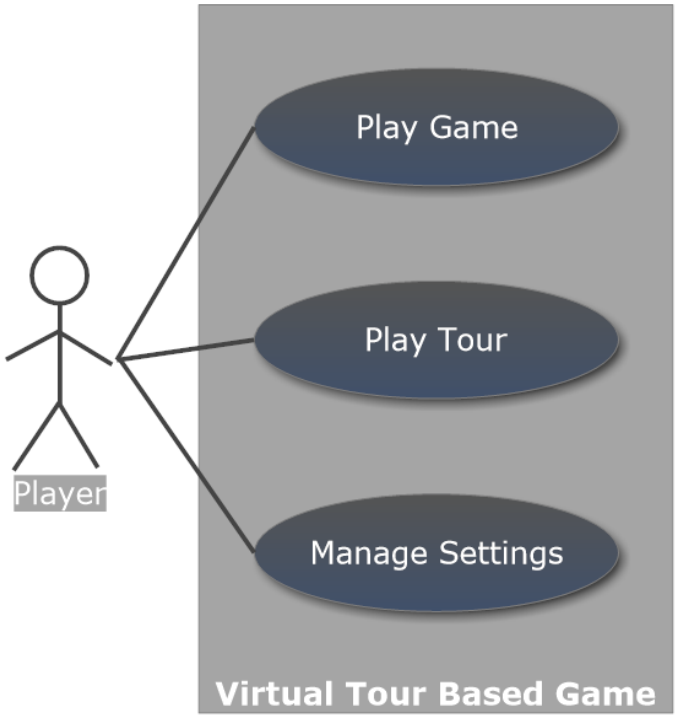
- **Information:** Anyone who wants to familiarize himself/herself with the map of CSE department IITG through a virtual tour of the department.

2.2 List of Use Cases

Player User use cases:

1. Play Game
2. Play Tour
3. Manage Settings

2.3 Use Case diagram



2.4 Description of Use Cases

2.4.1 U1: Play Game

- **Actors:** Player
- **Description:** Using this case player can start playing the game.
- **Scenario 1**: Mainline Sequence
  1. Player: Select "Play Game" option.
  2. System: Start the game by randomly positioning player in the map of CSE department, IITG and giving a task to player to reach at some position.
  3. Player: Start playing the game by pressing keys on keyboard and by movement of mouse.
  4. System: Changes the user's position/view showing some movement in the game.
  5. Player: Keep playing by making moves through keyboard and mouse until he has reached the final position.
  6. System: Evaluates and display score on basis of time taken to reach the destination.Then gives a prompt to choose from the following options:

- "Play Questionnaire": To start questionnaire round so that player can increase the final score.
- "New Game": To start a new game
- "Main Menu": To return to main menu and hence stop playing.

7. Player: Chooses "Play Questionnaire".
8. System: Displays all the questions with options.
9. Player: Chooses options for all the questions and presses "Submit".
10. System: Evaluates score on basis of answers given by player and displays the score as well as correct answers. Then it gives a prompt to choose from following options:
    - "New Game": To start the game again with different starting and final positions.
    - "Main Menu": To return to main menu and hence stop playing.
11. Player: If player chooses "New Game" then return to step(2) else if player chooses "Main Menu" option, return to main menu(step 0).

- **Scenario 2**: At step 5 and 7 of mainline sequence.
  1. Player: Chooses "Main Menu" option to end the current game and return to main menu.
  2. System: Ends current game and opens main menu without calculating score of player.
- **Scenario 3**: At step 7 of mainline sequence.
  1. Player: Chooses "New Game" option to play a new game.
  2. System: Return to Step(2) of mainline sequence.

2.4.2 U2: Play Tour

- **Actors:** Player
- **Description:** Using this use case player can either free roam and can know various things about the department, like shortest path from one position to another, various things about each room in the department, etc. or see a saved tour video
- **Scenario 1**: Mainline Sequence
  1. Player: Select "Play Tour" option.
  2. System: Starts the Tour by positioning player at the entrance of CSE department, IITG and allowing player to free roam.
  3. Player: Start playing the Tour by pressing keys on keyboard and by movement of mouse.

4. System: Changes the user's position/view showing some movement in the game.
5. Player: Keep playing by making moves through keyboard and mouse until he presses "Main Menu" Option.
6. System: End Tour and opens main menu(step 0).

- **Scenario 2**: from Mainline Sequence step 1
  1. Player: Select "Play Tour Video" option.
  2. System: Plays the saved video of Tour of CSE department with descriptions of places in the video.

2.4.3 U3: Manage Settings

- **Actors:** Player
- **Description:** Using this use case, the player can manage the game environment,for examples, game music, game sounds, movement controls and see a "HOW TO PLAY" tutorial .
- **Scenario 1**: Mainline Sequence
  1. Player: Select "Settings" option.
  2. System: Displays a sub-menu containing the followings:
     - "Movement": To allow the player to change the movement controls of the game.
     - "Sound": To allow the player to adjust the game sound volume.
     - "Music": To allow the player to adjust the game music volume.
     - "How to Play": To lead the player in an interactive session to teach him how to play the game and make various moves in the game.
     - "Main Menu": To allow the user to return to main menu.
  3. Player: Chooses one of 4 options.
  4. System: If the player selects
     - "Movement": Opens a sub-menu with options to change various movement controls of the game.
     - "Sound": Display a sidebar to increase/decrease sound.
     - "Music": Display a sidebar to increase/decrease music.
     - "How to Play": Open an interactive session where the player is instructed to make moves in a Tour or game and is shown the scoring criteria for the moves he makes or answer he gives.
     - "Main Menu": Return player to Main menu.

3 Domain Modelling

3.1 Boundary Objects

Finding boundary objects for different use cases:

- **U1: Play Game**
  1. PlayGame Boundary
- **U2: Play Tour**
  1. PlayTour Boundary
- **U3: Manage Settings**
  1. ManageSettings Boundary

3.2 Entity Objects

Finding Entity objects for each use case.

- **U1: Play Game**
  1. Map
  2. QuestionAnswers
- **U2: Play Tour**
  1. Map
  2. Video
- **U3: Manage Settings**
  1. Environment

3.3 Controller Objects

Finding Controller objects for each use case.

- **U1: Play Game**
  1. Navigation

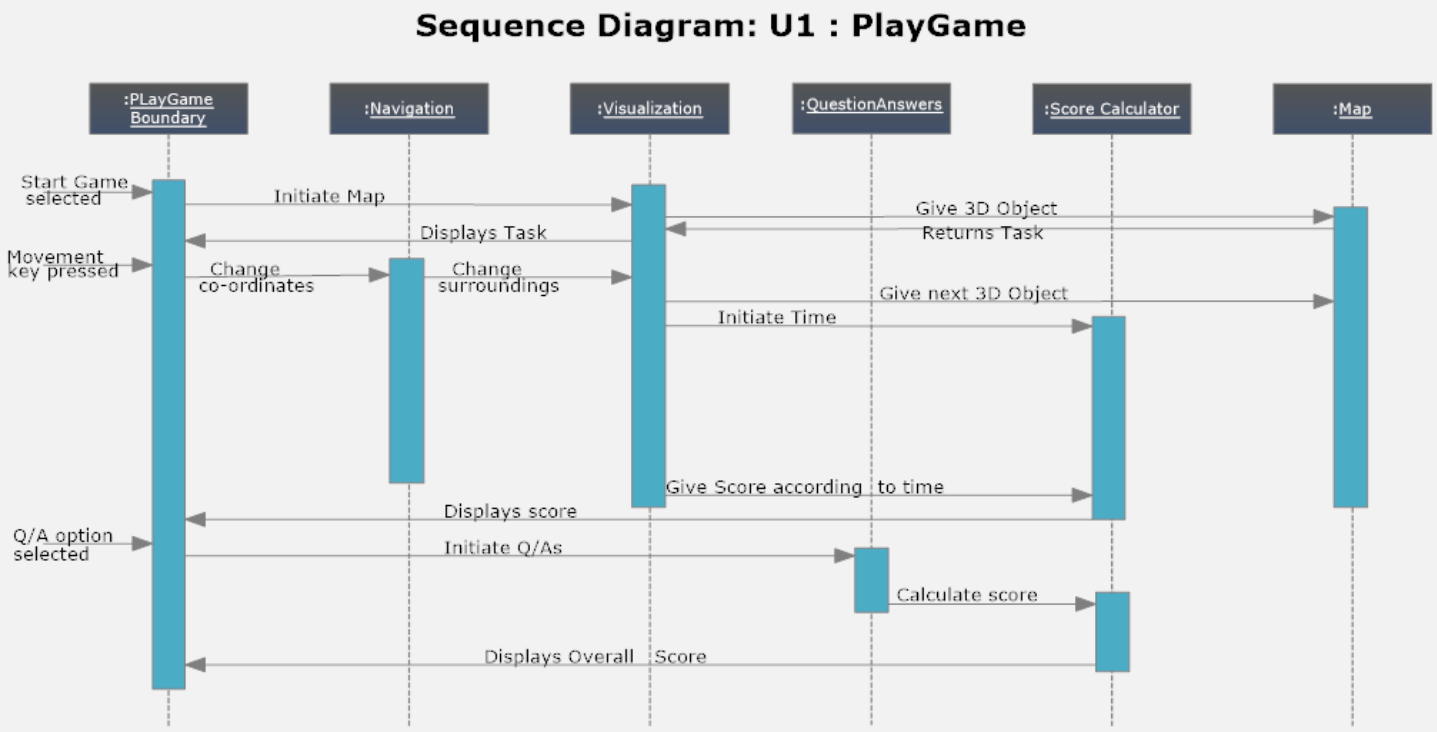
2. Visualization
3. Score Calculator

- **U2: Play Tour**
  1. Navigation
  2. Visualization
- **U3: Manage Settings**
  1. ManageSettings Controller

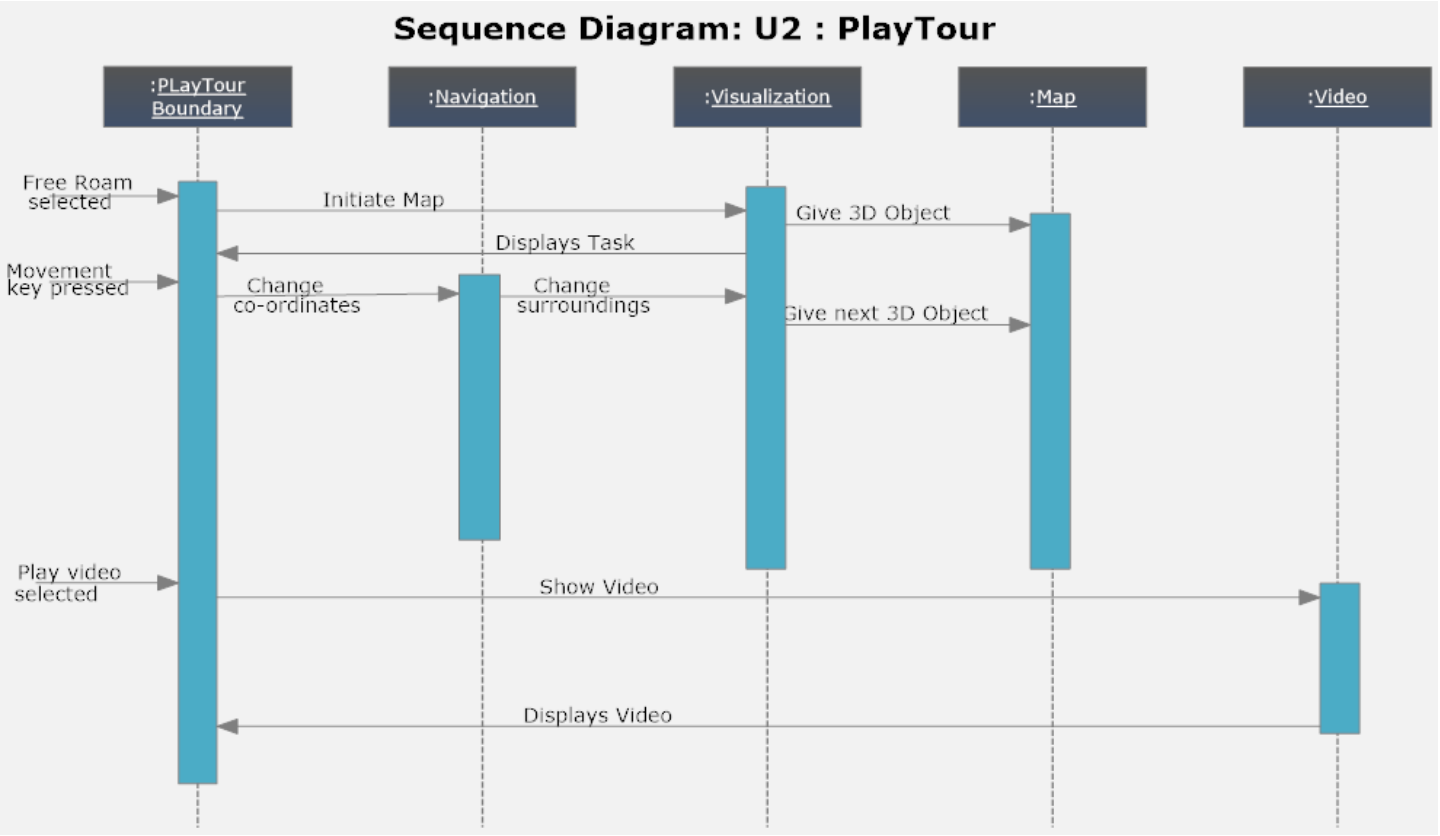
4 Sequence Diagram

Sequence diagram for all the use cases:

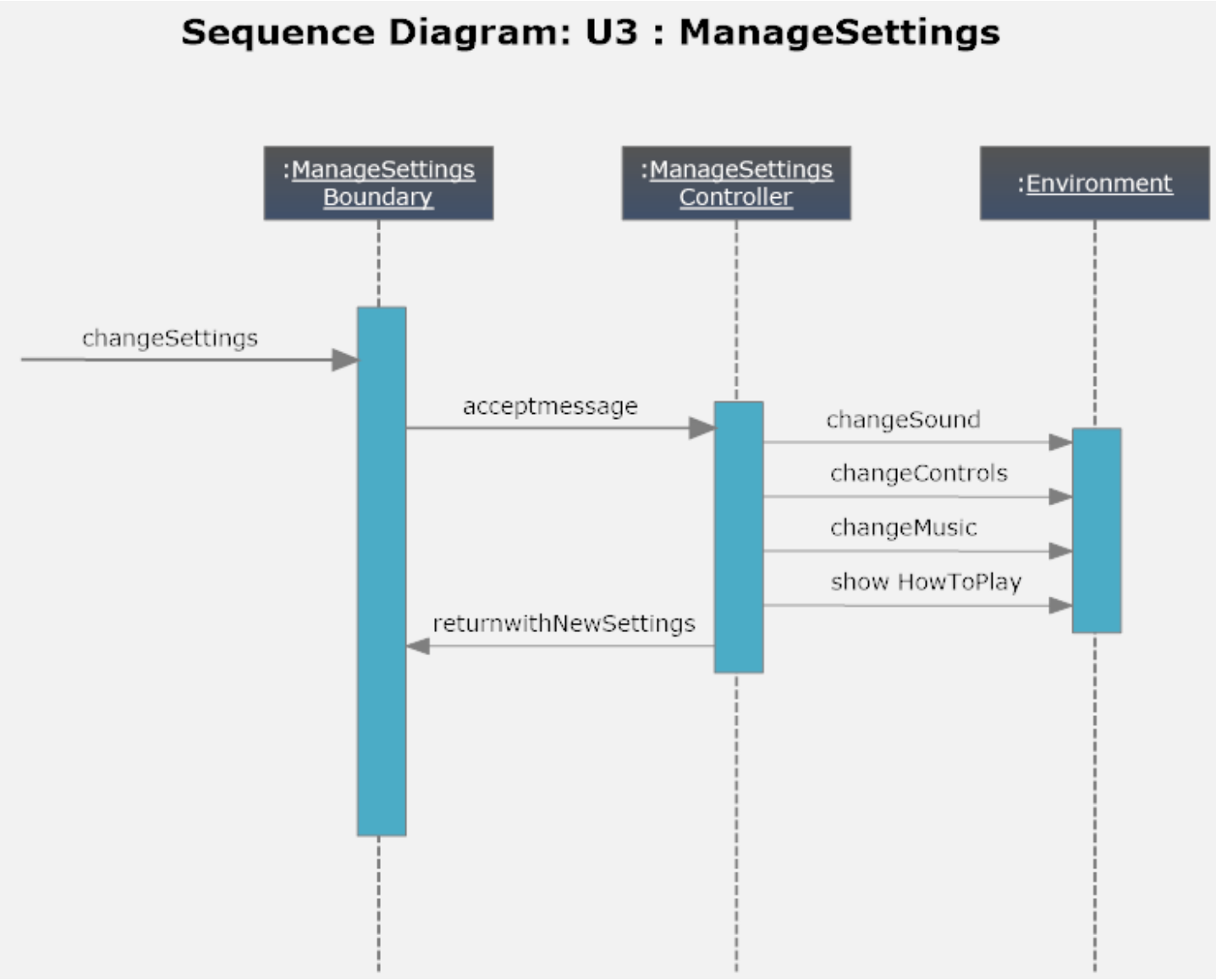
4.1 U1: Play Game



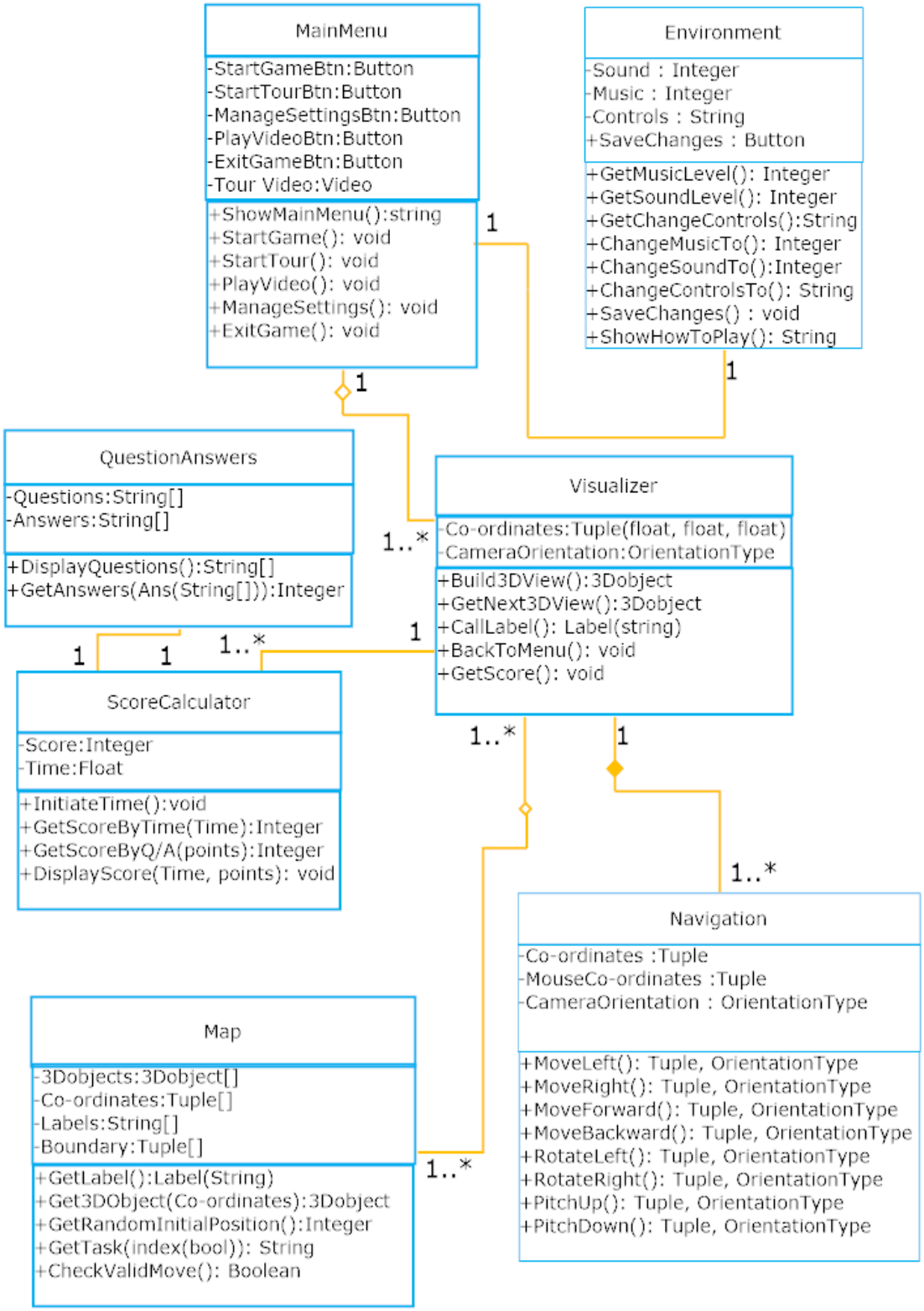
4.2 U2: Play Tour



4.3 U3: Manage Settings



5 Class Diagram


MainMenu
-StartGameBtn:Button
-StartTourBtn:Button
-ManageSettingsBtn:Button
-PlayVideoBtn:Button
-ExitGameBtn:Button
-Tour Video:Video
+ShowMainMenu():string
+StartGame(): void
+StartTour(): void
+PlayVideo(): void
+ManageSettings(): void
+ExitGame(): void
Environment
-Sound : Integer
-Music : Integer
-Controls : String
+SaveChanges : Button
+GetMusicLevel(): Integer
+GetSoundLevel(): Integer
+GetChangeControls():String
+ChangeMusicTo(): Integer
+ChangeSoundTo():Integer
+ChangeControlsTo(): String
+SaveChanges() : void
+ShowHowToPlay(): String
QuestionAnswers
-Questions:String[]
-Answers:String[]
+DisplayQuestions():String[]
+GetAnswers(Ans(String[])):Integer
Visualizer
-Co-ordinates:Tuple(float, float, float)
-CameraOrientation:OrientationType
+Build3DView():3Dobject
+GetNext3DView():3Dobject
+CallLabel(): Label(string)
+BackToMenu(): void
+GetScore(): void
ScoreCalculator
-Score:Integer
-Time:Float
+InitiateTime():void
+GetScoreByTime(Time):Integer
+GetScoreByQ/A(points):Integer
+DisplayScore(Time, points): void
Navigation
-Co-ordinates :Tuple
-MouseCo-ordinates :Tuple
-CameraOrientation : OrientationType
+MoveLeft(): Tuple, OrientationType
+MoveRight(): Tuple, OrientationType
+MoveForward(): Tuple, OrientationType
+MoveBackward(): Tuple, OrientationType
+RotateLeft(): Tuple, OrientationType
+RotateRight(): Tuple, OrientationType
+PitchUp(): Tuple, OrientationType
+PitchDown(): Tuple, OrientationType
Map
-3Dobjects:3Dobject[]
-Co-ordinates:Tuple[]
-Labels:String[]
-Boundary:Tuple[]
+GetLabel():Label(String)
+Get3DObject(Co-ordinates):3Dobject
+GetRandomInitialPosition():Integer
+GetTask(index(bool)): String
+CheckValidMove(): Boolean
1
1..*

6 Class Descriptions

6.1 MainMenu (Boundary : Class)

- **Attributes-:**
  - $\langle Button \rangle$ StartGameBtn
  - $\langle Button \rangle$ StartTourBtn
  - $\langle Button \rangle$ ManageSettingsBtn
  - $\langle Button \rangle$ ExitGameBtn
  - $\langle Button \rangle$ PlayVideoBtn
  - $\langle Video \rangle$ TourVideo
- **Methods-:**
  - **ShowMainMenu() : Constructor (public)**
    1. Parameters : void
    2. Return Value : names (Datatype : string[])
    3. Description : ShowMainMenu() method is a constructor which displays the list of options in main menu.
    4. Called By : When the game starts.
    5. Calls : NA
  - **StartGame() (public)**
    1. Parameters : void
    2. Return Value : void
    3. Description : StartGame() method is used to trigger the event start game on the instance StartGameBtn is Clicked.
    4. Called By : It is called when user clicks on Start Game Button.
    5. Calls : Visualizer class.
  - **StartTour() (public)**
    1. Parameters : void
    2. Return Value : void
    3. Description : StartTour() method is used to trigger the event start tour on the instance StartTourBtn is Clicked.

4. Called By : It is called when user clicks on Start Tour Button.
5. Calls : Visualizer class.

- **PlayVideo() (public)**
  1. Parameters : void
  2. Return Value : void
  3. Description : PlayVideo() method is used to start the saved TourVideo on the instance PlayVideoBtn is Clicked.
  4. Called By : It is called when user clicks on Play Video button.
  5. Calls : NA
- **manageSettings() (public)**
  1. Parameters : void
  2. Return Value : void
  3. Description : ManageSettings() method is used to trigger the event manage settings on the instance ManageSettingsBtn is Clicked.
  4. Called By : When user clicks on Manage Settings button.
  5. Calls : Environment Class.
- **ExitGame() (public)**
  1. Parameters : void
  2. Return Value : void
  3. Description : ExitGame() method is used to trigger the event exit game on the instance ExitgameBtn is Clicked.
  4. Called By : When user clicks on Exit Game button.
  5. Calls : NA

6.2 Score Calculator (Controller : Class)

- **Attributes-:**
  - int Score
  - float Time
- **Methods-:**
  - **InitiateTime() (public)**
    1. Parameters : void
    2. Return Value : void
    3. Description : It is used to set time attribute to 0.

4. Called By : Constructor of ScoreCalculator.
5. Calls : NA

– **GetScoreByTime() (public)**

1. Parameters : NA
2. Return Value : Returns a integer
3. Description : It is used to get score according to time elapsed to do the given task.
4. Called By : GetScore() in Visualizer class, when player reaches final position.
5. Calls : DisplayScore() in ScoreCalculator Class.

– **GetScoreByQ/A() (public)**

1. Parameters : points from Q/A(datatype:integer)
2. Return Value : Returns a integer
3. Description : It is used to get score according to number of correct answers.
4. Called By : GetAnswers in QuestionAnswers, when Submit button is clicked.
5. Calls : GetOverallScore() in ScoreCalculator.

– **DisplayScore() (public)**

1. Parameters : points from Q/A(datatype:integer)
2. Return Value : void
3. Description : It is used to print overall score according to performance till now.
4. Called By : GetScoreByTime() in ScoreCalculator Class, GetScoreByQ/A in ScoreCalculator Class.
5. Calls : void

6.3 Navigation (Controller : Class)

- **Attributes-:**
  - tuple $\langle float, float, float \rangle$ Coordinates
  - tuple $\langle float, float \rangle$ MouseCoordinates
  - $\langle Orientation_Type \rangle$ CameraOrientation
- **Methods-:**
  - **MoveLeft() (public)**

1. Parameters : NA
2. Return Value : Returns (Co-ordinates, Camera Orientation)
3. Description : Shift the user left with respect to current line of sight hence changing the coordinate variables.
4. Called By : Called when user presses left key on the keyboard.
5. Calls : Boundary Class to ensure user doesn't cross any boundary and Visualizer to construct surroundings for changed coordinates.

– **MoveRight() (public)**

1. Parameters : NA
2. Return Value : Returns (Co-ordinates, Camera Orientation)
3. Description : Shift the user right with respect to current line of sight hence changing the coordinate variables.
4. Called By : Called when user presses right key on the keyboard.
5. Calls : Boundary Class to ensure user doesn't cross any boundary and Visualizer to construct surroundings for changed coordinates.

– **MoveForward() (public)**

1. Parameters : NA
2. Return Value : Returns (Co-ordinates, Camera Orientation)
3. Description : Shift the user forward with respect to current line of sight hence changing the coordinate variables.
4. Called By : Called when user presses forward key on the keyboard.
5. Calls : Boundary Class to ensure user doesn't cross any boundary and Visualizer to construct surroundings for changed coordinates.

– **MoveBackward() (public)**

1. Parameters : NA
2. Return Value : Returns (Co-ordinates, Camera Orientation)
3. Description : Shift the user back with respect to current line of sight hence changing the coordinate variables.
4. Called By : Called when user presses back key on the keyboard.
5. Calls : Boundary Class to ensure user doesn't cross any boundary and Visualizer to construct surroundings for changed coordinates.

– **RotateLeft() (public)**

1. Parameters : NA
2. Return Value : Returns (Co-ordinates, Camera Orientation)

3. Description : It will change the camera orientation in the 3D visual certain degrees left about the vertical axis.
4. Called By : Called when mouse is moved left.
5. Calls : Visualizer to construct surroundings for changed mouse coordinates.

- **RotateRight() (public)**
  1. Parameters : NA
  2. Return Value : Returns a (Co-ordinates, Camera Orientation)
  3. Description : It will change the camera orientation in the 3D visual certain degrees right about the vertical axis.
  4. Called By : Visualizer to construct surroundings for changed mouse coordinates.
- **PitchUp() (public)**
  1. Parameters : NA
  2. Return Value : Returns a (Co-ordinates, Camera Orientation)
  3. Description : Changes user line of sight in the 3D visual certain degrees up.
  4. Called By : Visualizer to construct surroundings for changed mouse coordinates.
- **PitchDown() (public)**
  1. Parameters : NA
  2. Return Value : Returns a (Co-ordinates, Camera Orientation)
  3. Description : Changes user line of sight in the 3D visual certain degrees down.
  4. Called By : Visualizer to construct surroundings for changed mouse coordinates.

6.4 Visualizer (Controller : Class)

- **Attributes-:**
  - tuple $\langle float, float, float \rangle$ Coordinates
  - $\langle OrientationType \rangle$ CameraOrientation
- **Methods-:**
  - **Build3DView() (public)**
    1. Parameters : void

2. Return Value : Returns a 3D object
3. Description : It is use to initiate 3D surroundings according to the initial position and Camera orientation of the user.
4. Called By : Constructor of Visualizer class.
5. Calls : Map class to get 3D view associated to surroundings.

– **GetNext3DView() (public)**

1. Parameters : void
2. Return Value : Returns a 3Dobject
3. Description : GetNext3DView() method is used to change surroundings whenever user moves.As the surroundings change whenever the user moves, this function is needed to constantly calculate the 3D surroundings setup for each and every position and orientation of the user's view..
4. Called By : Navigation whenever any method of Navigation is called.
5. Calls : Map Class to get the next 3Dobject, Calls ScoreCalculator to initiate time.

– **CallLabel() (public)**

1. Parameters : void
2. Return Value : Returns a Label (datatype : string)
3. Description : CallLabel() method is used to get the label associated with given co-ordinates.
4. Called By : When LabelButton is pressed.
5. Calls : Map Class to get the label.

– **BackToMenu() (public)**

1. Parameters : void
2. Return Value : void
3. Description : BackToMenu() method is used to return to main menu.
4. Called By : Called when MainMenuButton is pressed.
5. Calls : ShowMainMenu() in MainMenu Class.

– **GetScore() (public)**

1. Parameters : void
2. Return Value : void
3. Description : GetScore() method is used to call ScoreCalculator to stop time and display score.
4. Called By : Called when user reaches final position is pressed.
5. Calls : GetScoreByTime in ScoreCalculator to display score according to time elapsed.

6.5 Map (Entity : Class)

- **Attributes :**
  - $\langle 3DObject \rangle$ 3Dobject
  - tuple $\langle float, float, float \rangle$ Co_ordinates
  - tuple $\langle float, float, float \rangle$ boundary[ ]
  - string[ ] Labels
- **Methods :**
  - **GetLabel() (public)**
    1. Parameters : Void
    2. Return Value : Returns a Label(datatype : String)
    3. Description : GetLabel() method is used to get label associated with the given co-ordinates
    4. Called By : CallLabel() from Visualizer.
    5. Calls : NA
  - **Get3Dobject() (public)**
    1. Parameters : Co-ordinates(datatype : int[3])
    2. Return Value : Returns a 3Dobject
    3. Description : Get3Dobject() method is used to get the 3D view associated with the given co-ordinates
    4. Called By : Build3DView() and GetNext3DView() from Visualizer.
    5. Calls : NA
  - **GetRandomInitialPosition() (public)**
    1. Parameters : Void
    2. Return Value : Returns a integer
    3. Description : GetRandomInitialPosition() method is used to get a random initial position index for array of tuples.
    4. Called By : GetTask() in Map Class .
    5. Calls : void.
  - **GetTask() (public)**
    1. Parameters : Index(datatype : bool)
    2. Return Value : Returns a string
    3. Description : GetTask() method is used to get the task if the index value is TRUE.In case of TRUE, call GetRandomInitialPosition() and returns a string and if its FALSE, it returns void.

4. Called By : Constructor of Map Class.
5. Calls : GetRandomInitialPosition() in Map.

– **CheckValidMove() (public)**

1. Parameters : void
2. Return Value : returns a bool value.
3. Description : It is used to check whether the move user made is within the boundary[ ] list.If not, return error saying button press is invalid.
4. Called By : Either one function in Navigation class.
5. Calls : NA

6.6 Environment (Entity : Class)

- **Attributes-:**
  - int Sound
  - int Music
  - string[ ] Controls
  - ⟨*Button*⟩ SaveChanges
- **Methods-:**
  - **GetMusicLevel() (public)**
    1. Parameters : void
    2. Return Value : MusicVolume(datatype: integer)
    3. Description : This method is used to get the music volume which user has typed.
    4. Called By : It is called when save changes button is clicked in settings.
    5. Calls : NA
  - **GetSoundLevel() (public)**
    1. Parameters : void
    2. Return Value : SoundVolume(datatype: integer)
    3. Description : This method is used to get the sound volume which user has typed.
    4. Called By : It is called when save changes button is clicked in settings.
    5. Calls : NA
  - **GetChangeControls() (public)**
    1. Parameters : void

2. Return Value : ControlString(datatype: string)
3. Description : This method is used to get the string which user has typed.
4. Called By : It is called when save changes button is clicked in settings.
5. Calls : NA

- **ChangeMusicTo() (public)**
  1. Parameters : NA
  2. Return Value : CurrentMusiclevel(datatype: integer)
  3. Description : ChangeMusicTo() method is used to change the music settings to given choice option.
  4. Called By : It is called when save changes button is clicked in settings.
  5. Calls : GetMusicLevel() in Environment Class.
- **ChangeSoundTo() (public)**
  1. Parameters : void
  2. Return Value : CurrentSoundlevel(datatype: integer)
  3. Description : ChangeSoundTo() method is used to change the sound settings to given choice option.
  4. Called By : It is called when save changes button is clicked in settings.
  5. Calls : GetSoundLevel() in Environment Class.
- **ChangeControlsTo() (public)**
  1. Parameters : void
  2. Return Value : NewControls(datatype : String)
  3. Description : ChangeControlsTo() method is used to change the controls to new set of controls.
  4. Called By : It is called when save changes button is clicked in settings.
  5. Calls : GetChangeControls() in Environment Class.
- **SaveChanges() (public)**
  1. Parameters : void
  2. Return Value : void
  3. Description : It is used to save changed settings and go back to Main Menu.
  4. Called By : It is called when save changes button is clicked in settings.
  5. Calls : ChangeMusicTo(), ChangeSoundTo(), ChangeControlsTo() in Environment Class and ShowMainMenu() of MainMenu Class.
- **ShowHowToPlay() (public)**

1. Parameters : void
2. Return Value : Returns a String
3. Description : ShowHowToPlay() method is used to show basic instructions of game.
4. Called By : It is called when ShowHowToPlay button is clicked in settings.
5. Calls : void

6.7 QuestionAnswers (Entity : Class)

- **Attributes-:**
  - String[ ] Questions
  - String[ ] Answers
- **Methods-:**
  - **DisplayQuestions() (public)**
    1. Parameters : void.
    2. Return Value : Returns a Array of Strings.
    3. Description : DisplayQuestions() method is used to display questions and wait for user to input answers and click Submit button.
    4. Called By : Constructor of QuestionAnswers Class.
    5. Calls : GetAnswers() in QuestionAnswers Class, when user presses Submit button.
  - **GetAnswers() (public)**
    1. Parameters : Strings of answers.
    2. Return Value : Returns a integer.
    3. Description : GetAnswers() method is used to give number of correct answers after comparing given input answers with the predefined answers.
    4. Called By : When user presses Submit Button on Questionnaire.
    5. Calls : GetScoreByQA() in ScoreCalculator.

NOTE : In all classes, attributes of classes are private. Also, classes for the actual display on the screen has not been made in this document because there are libraries(modules) which have all the required classes predefined. We only need to set the values to the instances of these classes.