

SOFTWARE ENGINEERING COURSEWORK 2

MAINTENANCE GUIDE

BLUE TEAM

*Christian Bentley, Aimee Dowell, Oscar Gee, Jaafar Ghaddar, Rohith
Kanjirappara, Justin Lam, Vishnu Vardhan and Abel Varghese*

Fruiteaser Maintenance Guide

Table of Contents:

Introduction	1
Installation Guide	2
System Requirements	2
Downloading and Installing Unity Hub.....	2
Downloading and Installing Unity	3
Setting Up the Fruiteaser Codebase.....	3
Installing Git MacOS	3
Installing Git Windows.....	4
Cloning the Repository	4
Opening the Project in Unity Editor	5
File organisation	7
System Design UML Class Diagram.....	8
Frontend UI Scripts	9
LevelController Class	9
Class Variables.....	9
Start()	11
Awake()	11
Update()	11
ContinueButtonClicked()	12
RetryButtonClicked()	12
ExitButtonClicked()	12
ResumeButtonClicked()	12
ReturnButtonClicked()	13
QuitWithoutSavingClicked().....	13
SaveAndQuitClicked()	13
SaveGame()	13
SelectTile(GameObject obj)	14
GetCoords(GameObject obj)	14
OnQuestionInputChanged().....	14
OnQuestionCorrect()	15
UpdateVegetablesRemaining()	15

FadeSpeechBubble()	16
Reset()	16
RemoveCarrotFromBoard()	16
RemoveBroccoliFromBoard()	16
PushVegetablesBack()	17
CheckConsecutiveEmptyTiles()	17
FindACarrotOnRow(GameObject[] row)	17
ShowLevelComplete()	18
ShowLevelFailed()	18
QuestionPopUpManager Class	19
Class Variables	19
Start()	19
HideQuestionPopUp()	20
HideAnswerNotification()	20
ResetQuestion()	20
SetQuestion(string questionText, double questionAnswer)	20
IsQuestionCorrect(string userAnswer)	21
UpdateScoreText()	21
IsNumeric(String str)	21
ShowQuestion()	21
ResetQuestionInput()	22
GetInputString()	22
MainMenuControllerClass	23
StartNewGameButtonClicked ()	23
LoadGameButtonClicked ()	23
QuitToDesktopButtonClicked()	23
LoadGame()	23
OpeningSceneController	24
StartButtonClicked()	24
Backend Scripts	25
BoardModel Class	25
Class Variables	25
Start()	25
Level	26
GridBoard	26

initializeBoard()	26
getGridSizeX().....	26
getGridSizeY().....	27
addCarrot().....	27
addBroccoli()	27
addBanana()	28
getDistance()	28
isCarrotPresent()	28
isBananaPresent()	28
isBroccoliPresent().....	28
makeGuess()	29
moveVegetables()	29
GetNewCarrotPosition()	30
GetNewBroccoliPosition().....	30
GetVegetablesPosition()	30
GetPreviousVegPosition().....	31
PushVegetablesBackOneMove().....	31
ArithmeticQuestionGenerator Class	31
Class Variables.....	31
Level.....	32
setUpperRange()	32
generateQuestion()	32
getRandomNum().....	32
isEven().....	33
isOdd()	33
UserScore Class	33
Class Variable	33
Start()	33
incrementScore().....	34
halveScore()	34
newQuestion().....	34
StaticVariables Class	35
Score	35
Level.....	35
StartingLevel	35

GameStatus.....	36
SteveQuotes Class	37
Class Variables.....	37
TileSelection.....	37
TileEmpty	37
CarrotFound	37
BroccoliFound	38
BananaFound	38
Free.....	38

Introduction

This documentation provides a thorough explanation of the existing FruiTeaser codebase and describes the necessary means to clone and contribute to the codebase, with instructions on hardware and software requirements and the use of git. It outlines the importance and role of each class, then details the variables and methods that they each contain. Through reading this document, a future developer of FruiTeaser should achieve a rounded understanding of the structural aspects of the code, and how and where to add future developments.

The document is divided into frontend and backend scripts. In general, the frontend scripts control the visibility and management of UI objects. The backend controls non-user facing aspects, such as question generation, and game board management. To visualise the relationship between the frontend and backend scripts, a visual depiction can be seen in the System Design UML Class Diagram section, below. Each class is represented as a UML class diagram, with arrows showing which classes own objects of other classes, the private and public variables, and methods contained within each class; static classes are indicated in bold.

Each class is described in detail, with a brief overview of its purpose and examples of suitable future expansions that could be made to that script. Tables are utilised to document class variables, along with their type and description, and each method is analysed with a description of its intended use, return type, access modifier, and examples of future work which could expand upon the existing functionalities.

Installation Guide

This section provides detailed instructions on how to install the Unity game editor, setup the codebase and run the FruiTeaser Game.

System Requirements

Operating System	Windows	macOs
Operating System Version	Windows 7 (SP1+), Windows 10 and Windows 11	High Sierra 10.13+
CPU	x86, x64 architecture with SSE2 instruction set support	x64 architecture with SSE2
Graphics API	DirectX 10, 11, 12 capable GPUs	Metal capable Intel and AMD GPUs
Additional Requirements	Hardware vendor officially supported drivers	Apple officially supported drivers

Downloading and Installing Unity Hub

1. Go to <https://unity3d.com/get-unity/download> and download the correct version of Unity Hub based on your operating system.

1. Download the Unity Hub

Follow the instructions onscreen for guidance through the installation process and setup.



[Download Hub V3 for Windows](#)^{BETA}

[Download Hub V3 for Mac](#)^{BETA}

[Download for Linux](#)^{BETA}


[Download for Windows](#)

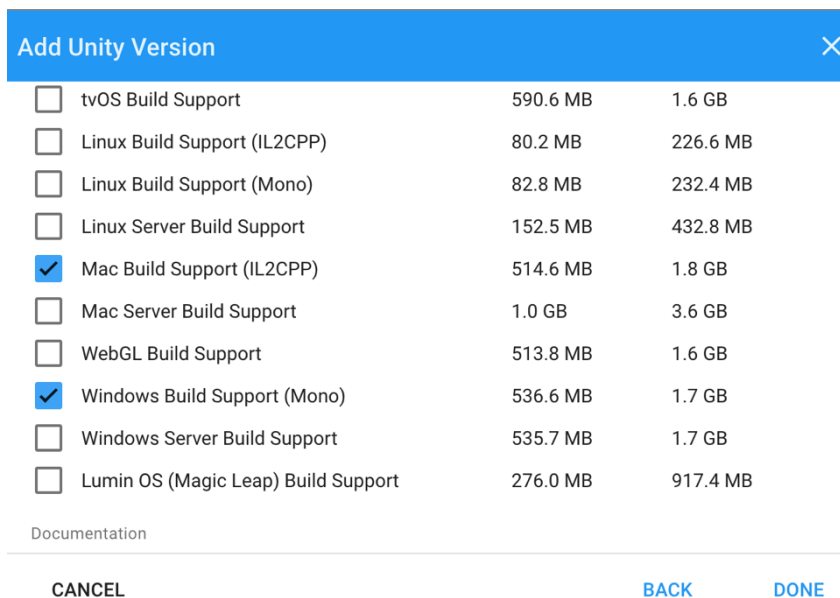
[Download for Mac](#)

2. Open the **UnityHubSetup.exe (Windows)** or **UnityHubSetup.dmg (Mac OS)** and follow the on-screen instructions through the installation process and setup.
3. Launch Unity Hub and click on the **profile icon**  on the top right of the page, then click on **sign in** and proceed to do so; if you do not have a Unity account, then please create one using the sign-up link and then sign in.
4. If you are a new user or have not activated a license, click on the **gear icon**  on the top right of the page. Go to the License Management tab and proceed to activate a new license that best fits your purpose for downloading of Unity.

Downloading and Installing Unity

The game was developed and runs on 2020.3.22f1 (LTS) version of Unity. To download the game and its code base, the correct version of Unity needs to be installed. This is downloaded via the Unity Hub.

1. Visit <https://unity3d.com/get-unity/download/archive> and locate the **2020.3.22f1 (LTS) version** of unity as labelled below.
2. Once located, click on the Unity Hub button  and the installation will launch in Unity Hub.
3. Tick the relevant build modules, as seen below depending on the system that you are running (Windows or Mac, etc.) and then click on **DONE** to begin the installation.



Build Module	Size (MB)	Size (GB)
<input type="checkbox"/> tvOS Build Support	590.6 MB	1.6 GB
<input type="checkbox"/> Linux Build Support (IL2CPP)	80.2 MB	226.6 MB
<input type="checkbox"/> Linux Build Support (Mono)	82.8 MB	232.4 MB
<input type="checkbox"/> Linux Server Build Support	152.5 MB	432.8 MB
<input checked="" type="checkbox"/> Mac Build Support (IL2CPP)	514.6 MB	1.8 GB
<input type="checkbox"/> Mac Server Build Support	1.0 GB	3.6 GB
<input type="checkbox"/> WebGL Build Support	513.8 MB	1.6 GB
<input checked="" type="checkbox"/> Windows Build Support (Mono)	536.6 MB	1.7 GB
<input type="checkbox"/> Windows Server Build Support	535.7 MB	1.7 GB
<input type="checkbox"/> Lumin OS (Magic Leap) Build Support	276.0 MB	917.4 MB

Documentation


CANCEL BACK DONE

Setting Up the FruiTeaser Codebase

The codebase of the current FruiTeaser product is managed via Github.com. The most recent working branch is located in the 'delivery' branch in the form of a git repository. To manage and contribute to the code of FruiTeaser, you should first clone the repository. The following guide will show how to clone and pull a local copy to your computer.

Installing Git MacOS

1. Check if Git is already installed on your Mac.
2. Open **Terminal** on your Mac by doing one of the following:
 - i. Using Spotlight Search (Command + Spacebar) and type in **Terminal**

- ii. Open Finder, , and navigate to **Applications > Utilities folder > Terminal**
3. Type **git --version** and hit enter. If you already have Git installed, the Terminal will show you which version you have, like the picture below. Skip to **Step 6** if you have Git installed.

```
$ git --version git version 2.7.0 (Apple Git-66)
```



4. If you do not have Git installed. You can download a stand-alone installer to install Git. Go to <https://sourceforge.net/projects/git-osx-installer/files/> and download the latest version. Open the dmg file and follow the prompts to install Git.
5. Repeat **Step 2 and 3** to ensure Git has been successfully downloaded.
6. In the **Terminal**, you can configure your Git username and email by using the following commands:
 - i. **git config --global user.name "type your name here"**
 - ii. **git config --global user.email "type your email here "**

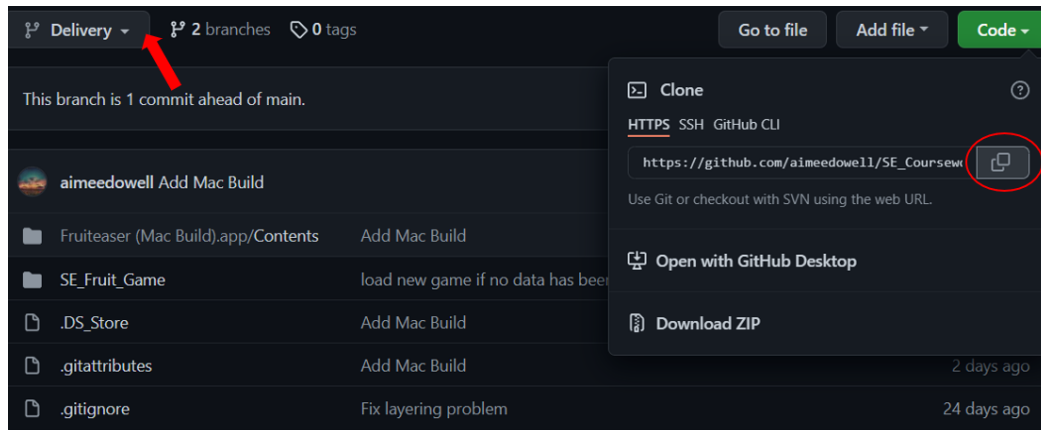
Installing Git Windows

1. Go to <https://git-for-windows.github.io/> and download the latest version.
2. Open the downloaded application and follow the prompts to finish the installation. The default options will be sensible for most users.
3. Open **Command Prompt** by typing **cmd** and hit enter in the windows search bar. Then type in the following commands to configure your username and email:
 - i. **git config --global user.name "type your name here"**
 - ii. **git config --global user.email "type your email here "**

Cloning the Repository

1. Go to the following link on a browser of your choice:
https://github.com/aimeedowell/SE_Coursework2_Fruit_Game/tree/delivery

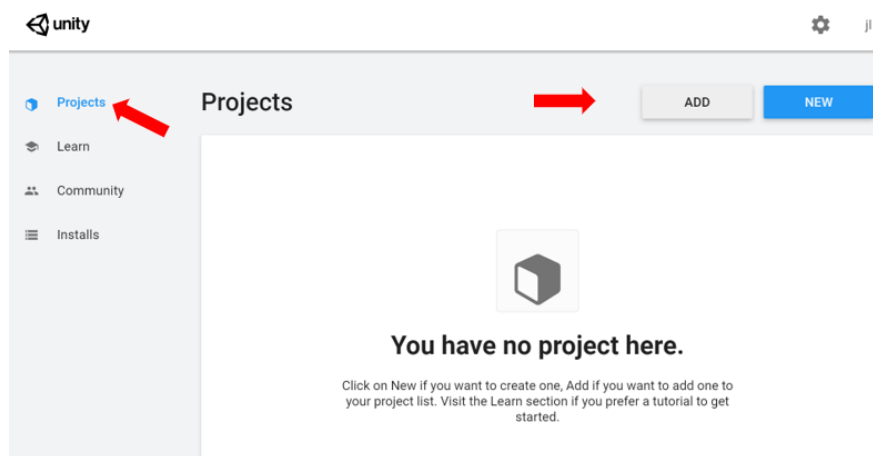
2. Make sure you are on the **delivery** branch. Click on the button  and select **HTTPS** and copy the URL by clicking the copy button  .



3. Open Terminal (Mac) or Command Prompt (Windows) and navigate to the path you want to locally store the repository using **cd name_of_directory**. Note that this method cannot jump to a directory nested two levels down. If you want to go back to the directory one level up, type: 'cd ../'. **Alternatively**, you may drag the selected directory into the Terminal or Command Prompt window after typing 'cd ' and the path will autocomplete.
4. Once you are at your preferred folder location. Type in the following command: **git clone copied-URL**
5. Your specified directory will now contain a repository called **SE_Coursework2_Fruit_Game**.

Opening the Project in Unity Editor

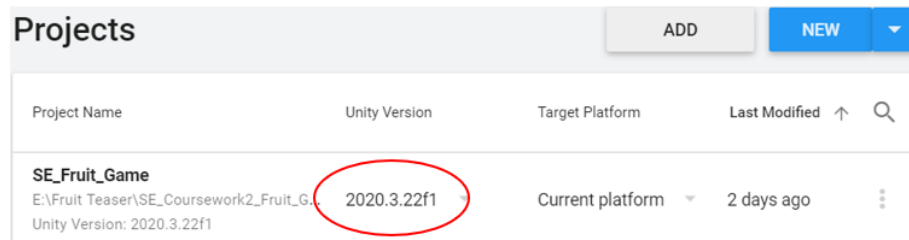
1. Open Unity Hub. Proceed to the **Projects** tab and click **ADD** button located on the top right corner.



2. Locate your folder cloned from Github. The folder should be named **SE_Coursework2_Fruit_Game**. Inside the folder, select the folder named

'SE_Fruit_Game' to add into Unity Hub. You should see the project 'SE_Fruit_Game' in the **Project** tab.

3. Locate your extracted folder. Inside the folder, select the folder named 'SE_Fruit_Game' to add into Unity Hub. You should see the project 'SE_Fruit_Game' in the **Project** tab.



Projects				ADD	NEW	▼
Project Name	Unity Version	Target Platform	Last Modified	↑	Q	
SE_Fruit_Game E:\Fruit Teaser\SE_Coursework2_Fruit_G... Unity Version: 2020.3.22f1	2020.3.22f1	Current platform	2 days ago			⋮

4. Make sure you are using the correct version of Unity, then click on the project to launch the Editor. It will take a few minutes to launch when opening the project for the first time.

You are now ready to start contributing to FruiTeaser. It is common practice to work on topic-related branches, and use pull requests reviewed by at least one other member of the FruiTeaser developer team, to ensure good coding practices are in place and that the **main** branch always contains a production level standard.

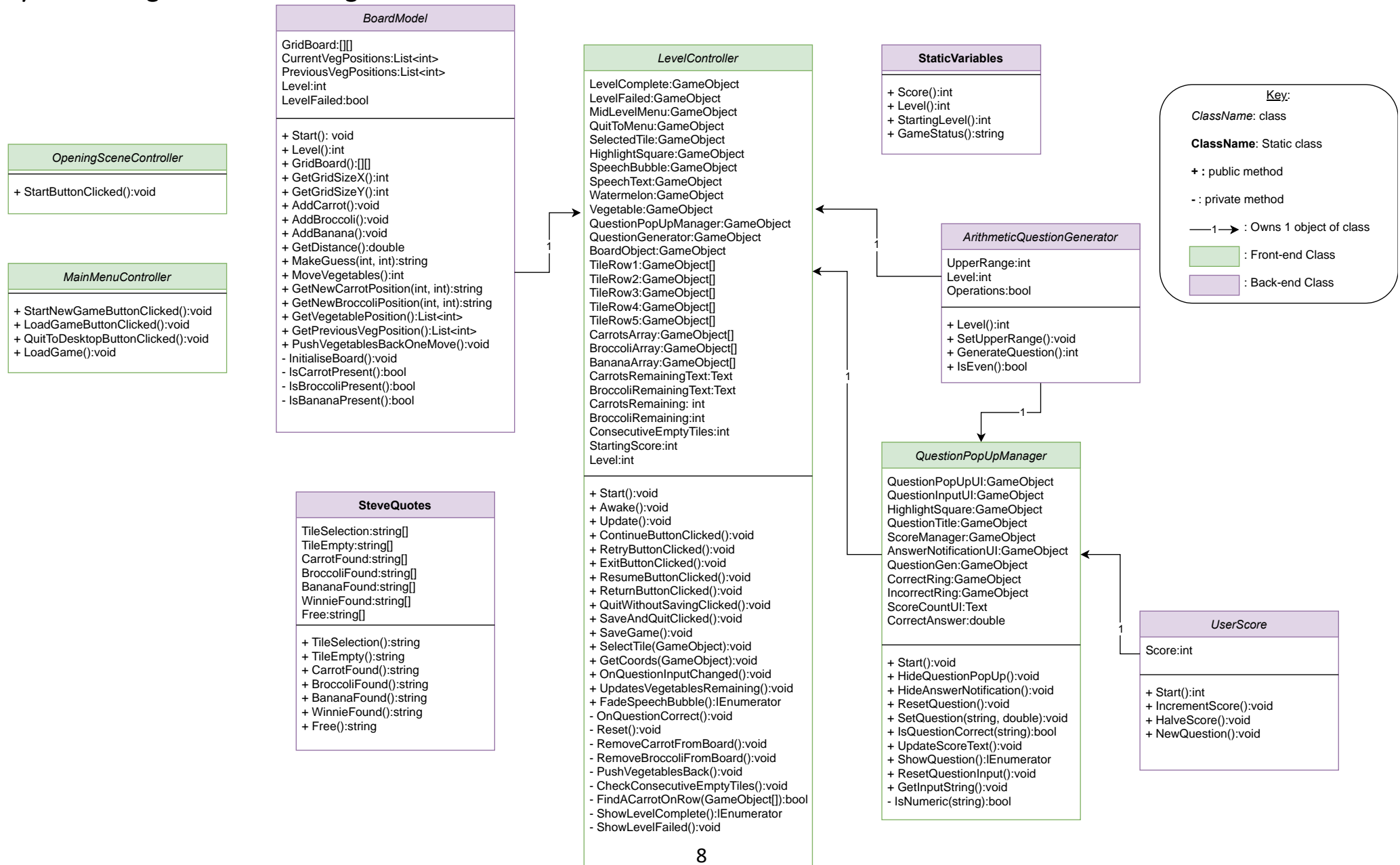
File organisation

When new files are added to the FruiTeaser project, it is important to maintain readability and easy navigation. All class names should reflect their intended purpose from the name only, and file types must be organised into specific folders, so that all developers have a uniform path to finding specific resources - time is not wasted in trying to locate them.

The main three file types and their paths are described below. It is not uncommon for these paths to contain multiple subdirectories, such as the UI folder containing a subdirectory for buttons. For files in which purpose can be clustered, it makes sense to contain these within their own subdirectory.

- When creating new scripts, these must be added to the project files via
.../SE_Coursework2_Fruit_Game/SE_Fruit_Game/Assets/Scripts
- When creating new test files, these must be added to the project files via
.../SE_Coursework2_Fruit_Game/SE_Fruit_Game/Assets/Tests
- When creating new UI assets, these must be added to the project files via
.../SE_Coursework2_Fruit_Game/SE_Fruit_Game/Assets/UI
- When creating new scenes, these must be added to the project files via
.../SE_Coursework2_Fruit_Game/SE_Fruit_Game/Assets/Scenes along with their corresponding .meta file.

System Design UML Class Diagram



Frontend UI Scripts

LevelController Class

The central frontend script used to manage the visually changing dynamic game objects/UI and the current state within the game. It controls the incrementation of levels and what happens for each game status. It holds BoardModel and QuestionPopUpManager class objects to uncover the objects underneath each tile and manage question pop ups.

The levels are as follows:

- Level 1: 3 x 3 grid
- Level 2: 5 x 5 grid
- Level 3: 7 x 3 grid

This class can be extended to include more levels, in which the developer must include new TileRow Game Objects arrays if the grid size extends 5 rows, and these should be initialised in the Unity editor. As a new scene must be created for each level, methods such as ContinueButtonClicked() must also be updated to allow it to move to these new Level scenes.

A further development of each level may be to create small animations, attached to specific Game Objects. An example of this would be a tile cracking and exploding when a user has clicked it and answered the question correctly. The start trigger of the animation must be controlled by this class, adding Animation.Play() during the correct GameStatus.

Any other UI which needs scripted control may also be managed in this class however, for UI which needs one than one method to control, it is best practise to create a new script, and create an object of this script within the LevelController.

Class Variables

Class variable	Type	Role
LevelComplete	GameObject	Stores a reference to the Game Object which holds the Level Complete Pop Up (background and buttons).
LevelFailed	GameObject	Stores a reference to the Game Object which holds the Level Failed Pop Up (background and buttons).
MidLevelMenu	GameObject	Stores a reference to the Game Object which holds the Mid-Level Pop Up (background and buttons).
QuitToMenuWarning	GameObject	Stores a reference to the Game Object which holds the Quit to Menu Warning Pop Up (background and buttons).
SelectedTile	GameObject	Stores the GameObject which is currently being selected during play in the corresponding game level.
SelectedTileCoords	Array of type Int	Stores the tile coordinates, for the grid, for the current selected tile during play in the corresponding game level.
SelectedTilePos	Vector3	Stores the position of the currently selected tile on the current scene as a Vector3 (x, y and z), for the corresponding game level.
TileRow1	Array of GameObjects	Stores a reference to the Game Objects on the first row of the game grid as a dynamic array.

Class variable	Type	Role
TileRow2	Array of GameObjects	Stores a reference to the Game Objects on the second row of the game grid as a dynamic array.
TileRow3	Array of GameObjects	Stores a reference to the Game Objects on the third row of the game grid as a dynamic array.
TileRow4	Array of GameObjects	Stores a reference to the Game Objects on the fourth row of the game grid as a dynamic array.
TileRow5	Array of GameObjects	Stores a reference to the Game Objects on the fifth row of the game grid as a dynamic array.
Vegetable	GameObject	Stores a reference to the vegetable sprite which has been discovered under the selected tile.
CarrotsArray	Array of GameObjects	Stores a reference to the Game Objects containing carrot sprites used within the corresponding level as an array.
BroccoliArray	Array of GameObjects	Stores a reference to the Game Objects containing broccoli sprites used within the corresponding level as an array.
BananaArray	Array of GameObjects	Stores a reference to the Game Objects containing banana sprites used within the corresponding level as an array.
CarrotsRemaining	Int	Sets the number of carrots for each level and stores a reference to current remaining carrots within the corresponding level.
BroccoliRemaining	Int	Sets the number of broccolis for each level and stores a reference to current remaining broccolis within the corresponding level.
BananasRemaining	Int	Sets the number of bananas for each level and stores a reference to current remaining bananas within the corresponding level.
HighlightSquare	GameObject	Stores a reference to the Game Object which holds the HighlightSquare sprite.
SpeechBubble	GameObject	Stores a reference to the Game Object which holds the SpeechBubble (sprite and text).
SpeechText	GameObject	Stores a reference to the Game Object which holds the Text for the SpeechBubble Game Object.
CarrotsRemainingText	Text	Stores a reference to the Text object which holds the UI text for displaying the number of remaining carrots in the corresponding level.
BroccoliRemainingText	Text	Stores a reference to the Text object which holds the UI text for displaying the number of remaining broccolis in the corresponding level.
QuestionPopUpManager	GameObject	Stores the reference for a new QuestionPopUpManager Script which gets initialised in the Start() function.
QuestionGenerator	GameObject	Stores the reference for a new ArithmeticQuestionGenerator Script which gets initialised in the Start() function.
BoardObject	GameObject	Stores the reference for a new BoardObject Script which gets initialised in the Start() function.

Class variable	Type	Role
Watermelon	GameObject	Stores a reference to the Game Object which holds the sprite for the watermelons within a corresponding game level.
consecutiveEmptyTiles	Int	Stores a reference to current number of consecutive empty tiles the user has pressed in the corresponding game level.
startingScore	Int	Sets the starting level of the games.
level	Int	Stores a reference to the current game level of the controller.

Start()

Method Name	Start()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	When a script is enabled, Start is called on the frame shortly before any of the Update methods are called for the first time. This function finds and assigns references to specific GameObjects from the class variables and sets the starting visibility of the GameObjects. It also sets the starting string for the variable GameStatus (StaticVariables).
Future Developments	All configurations for future development in the game should be initialised here
Access Modifier	Private
Return Type	Void

Awake()

Method Name	Awake()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	When the script instance is loaded, Awake is called. This function sets the variable StartingLevel (StaticVariables) to 1.
Future Developments	N/A
Access Modifier	Private
Return Type	Void

Update()

Method Name	Update()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	When the script instance is loaded, Awake is called. This function sets the variable StartingLevel (StaticVariables) to 1.
Future Developments	N/A
Access Modifier	Private
Return Type	Void

ContinueButtonClicked()

Method Name	ContinueButtonClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the continue buttons in each level. When the button is clicked, the next level is called, and the static variable Level is incremented.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

RetryButtonClicked()

Method Name	RetryButtonClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the retry buttons in each level. When the button is clicked, the Reset() method is called which resets all the variables in the level and removes all pop ups so the level has the impression of starting from the beginning,
Future Developments	N/A
Access Modifier	Public
Return Type	Void

ExitButtonClicked()

Method Name	ExitButtonClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the exit button in each level. When the button is clicked, the mid-level menu is made active and the GameStatus is set to 'Paused'. The function is guarded so it can only be run if the game is not completed or failed.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

ResumeButtonClicked()

Method Name	ResumeButtonClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the resume buttons in each level. When the button is clicked, the static variable GameStatus is set to 'TileSelection' and the mid-level menu is set to inactive.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

ReturnButtonClicked()

Method Name	ReturnButtonClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the return buttons in each level. When the button is clicked, level complete, failed, and mid-level menu are made invisible, and the quit to main menu game object is made visible.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

QuitWithoutSavingClicked()

Method Name	QuitWithoutSavingClicked ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the quit without saving button in each level. When the button is clicked, the scene is changed to the MainMenu and the LevelController script is reset using the Reset() method.
Future Developments	Tile animations could be played from this function.
Access Modifier	Public

SaveAndQuitClicked()

Method Name	SaveAndQuitClicked()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached via the button EventTrigger to the save and quit button in each level. When the button is clicked the SaveGame method is called, then the scene is changed to the MainMenu and the LevelController script is reset using the Reset() method.
Future Developments	Tile animations could be played from this function.
Access Modifier	Public

SaveGame()

Method Name	SaveGame()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Called from the Save and Quit button and stores the current score and game level to PlayerPrefs.
Future Developments	Vegetable locations and uncovered tiles could also be saved.
Access Modifier	Public

SelectTile(GameObject obj)

Method Name	SelectTile(GameObject obj)
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Attached to each tile in each level via an EventTrigger, passing the reference to itself as the function input. When the tile is clicked and the GameStatus is in 'TileSelection', coordinates, and position of the selected tile are stored, the highlight square object is set to active and positioned to the location of the selected tile. The GameStatus is then set to 'InQuestion' and a new question is generated via the QuestionGenerator, and SetQuestion is called via the QuestionPopUp object.
Future Developments	Tile animations could be played from this function.
Access Modifier	Public
Return Type	Void

GetCoords(GameObject obj)

Method Name	SelectTile(GameObject obj)
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Called from SelectTile, returns the coordinates of a selected game object based on the size of the grid and the position of the obj in the TileRow arrays.
Future Developments	Update this function for new grid sizes and levels
Access Modifier	Public
Return Type	Void

OnQuestionInputChanged()

Method Name	OnQuestionInputChanged()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Called from Update() every frame if the GameStatus is 'InQuestion'; it tracks changes to the answer input Text UI. If the enter key has been pressed, the script calls the IsQuestionCorrect method from the QuestionPopUpManager object. If this returns true, the GameStatus progresses to 'QuestionCorrect' and the OnQuestionCorrect method is called.
Future Developments	Update or expand this function if multiple choice questions are implemented
Access Modifier	Public
Return Type	Void

OnQuestionCorrect()

Method Name	OnQuestionCorrect ()
Location (Directory)\SE_Fruit_Game\Assets\Scripts\LevelController.cs
Description	<p>Called from OnQuestionInputChanged() if the question is correct. This method has a delay of 1 second for added suspense.</p> <p>The answer input UI is emptied, so each time a new question is shown the answer box starts empty and make the question pop up inactive.</p> <p>The function interacts with BoardModel object, using the selected tile coordinates to check the tile for vegetables or fruit. From this:</p> <ol style="list-style-type: none"> 1) A carrot is found and RemoveCarrotFromBoard() is called. 2) A broccoli is found and RemoveBroccoliFromBoard() is called. 3) A banana is found and the carrots are pushed back to the previous tile using PushVegetablesBack() 4) Tile is empty <ol style="list-style-type: none"> a. The consecutive empty tile variable is updated and calls CheckConsecutiveEmptyTiles() to see if Winnie the Watermelon should be implemented b. The vegetables are moved one step towards Steve the Strawberry using the Board Model method moveVegetables(). c. The positions of the moved vegetables are checked. If the new position is the position of Steve, ShowLevelFailed() is called <p>The static variable GameStatus is changed back to 'TileSelection' unless the level is complete or failed.</p> <p>Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component.</p>
Future Developments	Update this function for to check for new fruit and vegetables and add their functionalities
Access Modifier	Private
Return Type	IEnumerator

UpdateVegetablesRemaining()

Method Name	UpdateVegetablesRemaining ()
Location (Directory)\SE_Fruit_Game\Assets\Scripts\LevelController.cs
Description	<p>Updates the Text UI for CarrotsRemainingText and BroccoliRemainingText. The name objects are found and the Text component extracted and updated using the int variables which track carrots or broccoli remaining and converts this value to a string.</p>
Future Developments	Update or expand this function for more vegetables in a level.
Access Modifier	Public
Return Type	Void

FadeSpeechBubble()

Method Name	UpdateVegetablesRemaining ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	This function sets SpeechBubble object to active and uses CrossFadeAlpha to fade the bubble object in and out during a period of 2.5 seconds.
Future Developments	Update the fade/in out length.
Access Modifier	Public
Return Type	IEnumerator

Reset()

Method Name	Reset()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	All variables which need to be set back to their initial values, and objects which should no longer be visible should be re-initialised in this function. For example, carrotsRemaining should be set to its initial value, GameStatus should be set to 'TileSelection' etc.
Future Developments	All configurations which need to be set back to their initial values should be re-initialised here.
Access Modifier	Private
Return Type	Void

RemoveCarrotFromBoard()

Method Name	RemoveCarrotFromBoard ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	A carrot sprite to the set to the position of the selected tile and activated. The remaining carrot int is updated. If this variable is now 0 (and all other vegetables remaining are 0), set GameStatus to 'LevelComplete' and show Level Complete popup. Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component to CarrotFound.
Future Developments	Add animations
Access Modifier	Private
Return Type	Void

RemoveBroccoliFromBoard()

Method Name	RemoveBroccoliFromBoard ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	A broccoli sprite to the set to the position of the selected tile and activated. The remaining broccoli int is updated. If this variable is now 0 (and all other vegetables remaining are 0), set GameStatus to 'LevelComplete' and show Level Complete popup. Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component to BroccoliFound.
Future Developments	Add animations
Access Modifier	Private
Return Type	Void

PushVegetablesBack()

Method Name	PushVegetablesBack ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	A banana sprite to the set to the position of the selected tile and activated. The remaining banana int is updated. The BoardModel object method PushCarrotsBackOneMove() is called to push the carrots back to their previous position. set GameStatus to 'LevelComplete' and show Level Complete popup. Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component to BananaFound and FadeSpeechBubble() is called.
Future Developments	Add animations
Access Modifier	Private
Return Type	Void

CheckConsecutiveEmptyTiles()

Method Name	CheckConsecutiveEmptyTiles()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	If the number of consecutive empty tiles is greater than 3, then Winnie the Watermelon is positioned to the selected tile and activated. A carrot position is found and coordinates are returned using FindACarrotOnRow(GameObject[] row) and the TileRow Game Object Arrays. Once a carrot is found, it is removed from the board using the RemoveCarrotFromBoard() method. A check to see if the remaining vegetables are now all empty is conducted and ShowLevelComplete() called if this is true. Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component to WinnieFound and FadeSpeechBubble() is called.
Future Developments	N/A
Access Modifier	Private
Return Type	Void

FindACarrotOnRow(GameObject[] row)

Method Name	FindACarrotOnRow(GameObject[] row)
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Takes in a TileRow GameObject array and for each GameObject it calls the makeGuess() method from the BoardModel script. If a "Carrot" is returned, the SelectedTilePos is updated to match the carrots position and the method returns true.
Future Developments	N/A
Access Modifier	Private
Return Type	Bool

ShowLevelComplete()

Method Name	ShowLevelComplete ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	Updates the Text UI for SpeechBubble object using the SpeechText object and getter and setter functions from SteveQuotes to update the SpeechText Text component to Free and FadeSpeechBubble() is called. A delay of one second is implemented before the LevelComplete Game Object is set to active, and the Game Status is changed to 'LevelComplete'
Future Developments	Add animations
Access Modifier	Private
Return Type	IEnumerator

ShowLevelFailed()

Method Name	ShowLevelFailed ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/LevelController.cs
Description	The LevelFailed Game Object is set to active, and the Game Status is changed to 'LevelFailed' The static variable score is set back to its original value as it was when the level first started.
Future Developments	Add animations and loss of life
Access Modifier	Private
Return Type	Void

QuestionPopUpManager Class

This script controls the UI for question pop-ups, managing the user input and answer correct/incorrect visuals. It owns an ArithmeticQuestionGenerator class object to generate questions and pass them to the LevelController and determine whether the player's inputs are correct.

Anything relating to the visual aspect of the Question Popup may be extended in this class. For example, this script may be expanded to deal with multiple choice questions, for which new UI assets should be added to the QuestionPopUpUI GameObject which holds the containing UI for a Question Popup. This method may require the IsQuestionCorrect to become overloaded, adding a separate implementation based on which answer the user has selected.

Class Variables

Class variable	Type	Role
QuestionPopUpUI	GameObject	Stores a reference to the Game Object which holds all the UI for the Question Pop Up UI.
QuestionInputUI	InputField	Stores a reference to the InputField object which holds the inputUI for the question answer.
HighlightSquare	GameObject	Stores a reference to the Game Object which holds the HighlightSquare sprite.
QuestionTitle	GameObject	Stores a reference to the Game Object object which holds the Text for a given question.
ScoreManager	GameObject	Stores the reference for a new UserScore Script which gets initialised in the Start() function.
ScoreCountUI	Text	Stores a reference to the ScoreCountUI Text object which holds the current score as a Text sprite.
correctAnswer	double	Stores a reference to the correct answer of the current generated question as an Int.
AnswerNotificationUI	GameObject	Stores a reference to the Game Object which holds the AnswerNotificationUI sprite (backgroundf and text).
QuestionGen	GameObject	Stores the reference for a new ArithmeticQuestionGenerator Script which gets initialised in the Start() function.
CorrectRing	GameObject	Stores a reference to the Game Object which holds the CorrectRing sprite.
IncorrectRing	GameObject	Stores a reference to the Game Object which holds the IncorrectRing sprite.

Start()

Method Name	Start()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	When a script is enabled, Start is called on the frame shortly before any of the Update methods are called for the first time. This function initialises the ScoreManager and Question Gen Game Objects from their equivalent scripts, UserScore and ArithmeticQuestionGenerator, respectively.
Future Developments	All configurations for future development in the game should be initialised here
Access Modifier	Public
Return Type	Void

HideQuestionPopUp()

Method Name	HideQuestionPopUp ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Sets the QuestionPopUpUI game object to false, which hides the entire Question Pop Up UI.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

HideAnswerNotification()

Method Name	HideAnswerNotification ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Sets the AnswerNotificationUI game object to false, which hides the notification which tells you if the inputted answer is incorrect. AnswerNotificationUI (background and text).
Future Developments	N/A
Access Modifier	Public
Return Type	Void

ResetQuestion()

Method Name	ResetQuestion()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Re-initialises objects within the QuestionPopUpUI Object, such as setting the AnswerNotificationUI and IncorrectRing to inactive. It also generates a new question from ArithmeticQuestionGenerator and clears the answer input.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

SetQuestion(string questionText, double questionAnswer)

Method Name	SetQuestion()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Uses the tuple result from QuestionGen.generateQuestion(), to set the text of the Question title UI to the generated question and update the correctAnswer variable with the right answer.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

IsQuestionCorrect(string userAnswer)

Method Name	IsQuestionCorrect()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Converts the users answer to a float. Returns true if the given input from the Answer InputField is correct (matches the correctAnswer variable) or false if it does not match the correctAnswer variable, or if the string cannot be converted to a float (answer is not numeric). Sets the visibility of the incorrect or correct UI rings. Connects with the UserScore, to increment the score if the answer is correct, or halve the score when it is wrong.
Future Developments	N/A
Access Modifier	Public
Return Type	Bool

UpdateScoreText()

Method Name	UpdateScoreText ()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Locally stores the current score from StaticVariables.Score. Changes the score UI Text output to match the score received from the StaticVariables class.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

IsNumeric(String str)

Method Name	IsNumeric()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Checks a given string and returns true if it can be converted into an Int or a Float, returns false if this conversion fails.
Future Developments	N/A
Access Modifier	Private
Return Type	Bool

ShowQuestion()

Method Name	ShowQuestion()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Waits for one second before setting the QuestionPopUI to active, and any unwanted active objects within QuestionPopUI such as AnswerNotificationUI to inactive. This will be what is shown to the user once a tile is selected.
Future Developments	N/A
Access Modifier	Public
Return Type	IEnumerator

ResetQuestionInput()

Method Name	ResetQuestionInput()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Sets the answer InputField to an empty string, to clear the input box ready for a fresh question.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

GetInputString()

Method Name	GetInputString()
Location (Directory)SE_Fruit_Game/Assets/Scripts/QuestionPopUpManager.cs
Description	Returns the current string held within the question answer InputField.
Future Developments	N/A
Access Modifier	Public
Return Type	String

MainMenuControllerClass

This script will control the attached events to UI buttons for during the main menu.

UI displayed in the main menu scene which needs scripted control should be managed in this class, for example, if a high score chart was displayed using data containing all score values of FruiTaser users, held by a server, this should be displayed and updated here.

StartNewGameButtonClicked ()

Method Name	StartNewGameButtonClicked()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/ MainMenuControllerClass.cs
Description	Called from the Start new game button and uses the scene manager to switch to the first level of the game, resetting the static variables, level and score to 1 and 0, respectively.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

LoadGameButtonClicked ()

Method Name	LoadGameButtonClicked()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/ MainMenuControllerClass.cs
Description	Called from the Load game button and uses the scene manager to switch to the previous level and score of the game, accessed via LoadGame(). Changes the scene using SceneManager to the returned Level variable.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

QuitToDesktopButtonClicked()

Method Name	QuitToDesktopButtonClicked()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/ MainMenuControllerClass.cs
Description	Called from the Quit game button and quits the application.
Future Developments	Warning message before quitting.
Access Modifier	Public
Return Type	Void

LoadGame()

Method Name	LoadGame()
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/ MainMenuControllerClass.cs
Description	Called from the LoadGameButtonClicked() method and accesses the current stored score and game level from PlayerPrefs if they exist. If they do not exist, a new game is started.
Future Developments	Vegetable locations and uncovered tiles could also be loaded.
Access Modifier	Public
Return Type	Void

OpeningSceneController

This script controls the first scene which appears when the game application is opened.

UI displayed in this opening scene which needs scripted control should be managed in this class, for example, if online question banks are utilised in future, a loading bar could be incorporated here, to account for the time taken to retrieve question packages from a server on start up.

Other expansions could include animations such as a short video explaining the back story of the game, i.e., how Steve the Strawberry became trapped in the game board and showing his threat of vegetables hunting him down.

StartButtonClicked()

Method Name	StartButtonClicked()
Location (Directory)\SE_Fruit_Game\Assets\Scripts\ OpeningSceneController.cs
Description	Called from the Opening Scene when a start game button is clicked and uses the scene manager to switch to the main menu scene.
Future Developments	N/A
Access Modifier	Public
Return Type	Void

Backend Scripts

BoardModel Class

The central backend script used to manage the different pieces & the current state within the game. Based on a specified level, the script manages the board dimensions, positioning & movement of pieces as well as the general dynamics of the game.

The levels are as follows:

- Level 1: 3 x 3 grid
- Level 2: 5 x 5 grid
- Level 3: 7 x 3 grid

When a new level is added to the game, its grid size must be added to the initialisation of this class, to adequately track and position vegetables. Other extensions to this class include new fruit and vegetables. The number of fruit and vegetables must be initialised for each in the level during initialisation. As vegetables movement are unique, each vegetables specific movement must be implemented and called from the MoveVegetables() method. If new fruit powers involve movement or positioning of other fruit and vegetables, then their functionality should be included in this class and called from the Level Controller script.

Class Variables

Class variable	Type	Role
gridBoard	Two-dimensional string array	Initially the values in the gridboard are set to null. Once the different pieces are set, the null values are replaced with the name. Examples include Carrot, Broccoli, Banana, Strawberry.
currentVegPositions	List of integers	This stores the current grid coordinates for all vegetables within the game.
previousVegPosition	List of integers	This stores a list of the previous positions of each vegetable in the game. This is then used when moving the vegetables back 1 position when a fruit is uncovered.
Level	Int	Variable to keep track of the current level. Based on this figure, the grid size & question package is determined.
levelFailed	Boolean	A Boolean variable set to check whether the current level has been failed.

Start()

Method Name	Start()
Location (Directory)\SE_Fruit_Game\Assets\Scripts\BoardModel.cs
Description	This is the function called by the controller scripts when the game is started. This function in turn calls the initializeBoard function.
Future Developments	All configurations for future development in the game should be initialised here
Access Modifier	Public
Return Type	Void

Level

Method Name	Level
Location (Directory)SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description	Getter and setter method for the front end controller scripts to set the level for a new game. This will affect the size of the grid & the number of pieces on the board.
Future Developments	Extra checks can be placed here to ensure the value set for level is between a particular range
Access Modifier	Public
Return Type	Int

GridBoard

Method Name	GridBoard
Location (Directory)SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description	Getter method to return the gridBoard class variable
Future Developments	N/A
Access Modifier	Public
Return Type	Two-dimensional Array of strings

initializeBoard()

Method Name	initializeBoard()
Location (Directory)SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description	Strawberry is placed at the centre of the board. The add carrot function is called which places the carrot on the grid board. If the grid size is more than 3, this function adds another vegetable – Broccoli & another fruit – banana.
Future Developments	This is the function to be modified when new fruits or vegetables are added in the future
Access Modifier	Private
Return Type	Void

getGridSizeX()

Method Name	getGridSizeX()
Location (Directory)SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description	Returns the number of tiles in the y direction based on the current level.
Future Developments	New values for the y dimension size of the grid can be set in this method. When new map shapes are added to the game, the dimensions can be set here.
Access Modifier	Public
Return Type	Int

getGridSizeY()

Method Name	getGridSizeY()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description	Returns the number of tiles in the x direction based on the current level.
Future Developments	New values for the x dimension size of the grid can be set in this method. When new map shapes are added to the game, the dimensions can be set here.
Access Modifier	Public
Return Type	Int

addCarrot()

Method Name:	addCarrot()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	<p>This function places the carrots randomly across the board. There needs to be conditions satisfied for the placement of carrot to be successful. The following conditions must be satisfied –</p> <ol style="list-style-type: none"> 1. Coordinate in the grid must not have any other vegetable/fruit 2. Distance of the vegetable from the strawberry is determined by the current level & the distance is calculated using the getDistance() function. This ensures that the game is configured accordingly to fit different board sizes. <p>Found cell is the flag is used in the function to check whether a suitable cell has been found.</p>
Future Developments	Different size or types of carrots can be added using this method.
Access Modifier	Public
Return Type	Void

addBroccoli()

Method Name:	addBroccoli()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	<p>This function places the Broccoli randomly across the board. There needs to be conditions satisfied for the placement of broccoli to be successful. The following conditions must be satisfied –</p> <ol style="list-style-type: none"> 1. Coordinate in the grid must not have any other vegetable/fruit 2. Distance of the vegetable from the strawberry is determined by the current level & the distance is calculated using the getDistance() function. This ensures that the game is configured accordingly to fit different board sizes. <p>Found cell is the flag is used in the function to check whether a suitable cell has been found.</p>
Future Developments	Different size or types of Broccoli can be added using this method.
Access Modifier	Public
Return Type	Void

addBanana()

Method Name:	addBanana()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function places the Banana randomly across the board. There needs to be conditions satisfied for the placement of banana to be successful. The following conditions must be satisfied – 1. Coordinate in the grid must not have any other vegetable/fruit Found cell is the flag is used in the function to check whether a suitable cell has been found.
Future Developments	Different size or types of Bananas can be added using this method.
Access Modifier	Public
Return Type	Void

getDistance()

Method Name:	getDistance(int xCoord, int yCoord)
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	Helps to calculate the distance of the input from the position of the strawberry. It takes two inputs as x & y coordinates.
Future Developments:	N/A
Access Modifier	Public
Return Type	Double

isCarrotPresent()

Method Name:	isCarrotPresent()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function takes x & y coordinates and returns true value if a carrot is present underneath the block.
Future Developments:	N/A
Access Modifier	Private
Return Type	Boolean

isBananaPresent()

Method Name:	isBananaPresent()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function takes x & y coordinates and returns true value if a Banana is present underneath the block.
Future Developments:	N/A
Access Modifier	Private
Return Type	Boolean

isBroccoliPresent()

Method Name:	isBroccoliPresent()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function takes x & y coordinates and returns true value if a Broccoli is present underneath the block.
Future Developments:	N/A
Access Modifier	Private
Return Type	Boolean

makeGuess()

Method Name:	makeGuess()
Location (Directory):/SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	<p>It takes the x & y coordinates of the block the user has clicked and then returns a cellflag to the front end which tells whats underneath the block.</p> <p>CellFlag:</p> <ul style="list-style-type: none"> • “Null” – No vegetable/fruit underneath the block the user has clicked, • “Carrot” – A carrot is underneath the block the user has selected, • “Banana” – A Banana is underneath the block the user has selected, • “Broccoli” – A Broccoli is underneath the block the user has selected, • “Invalid” – do nothing, kept as failproof in case the user clicks on the same block more than once. <p>The values of the grid board are updated to “found” if the user clicks on block having the carrot and guessed if there was no carrot underneath the block the user has guesses.</p>
Future Developments:	When new fruits are added, the range of string values can be updated in this method.
Access Modifier	Public
Return Type	String

moveVegetables()

Method Name:	moveVegetables()
Location (Directory):/SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	<p>This function when called, moves the vegetables. It sweeps each position in a grid and if that position has a vegetable, then it passes the grid position to a function corresponding to the vegetable in that position. It then calls a function to determine the next position of a vegetable when it moves .</p> <p>It also saves the position of carrot prior to moving.</p>
Future Developments:	Currently, it works for carrots but it can be altered using else if statements to include other vegetables as well.
Access Modifier	Public
Return Type	Int

GetNewCarrotPosition()

Method Name:	GetNewCarrotPosition (int xcor, int ycor)
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function when called, moves the carrot by one position – either up, down, left or right. This function has the current x & y coordinates of the carrot on the gridboard as the input. If all the positions are blocked for single movement of carrots, the carrot can jump more than 1 cell. The distance of the carrot is calculated for all the movement of the carrots. The movement having the greatest change in the distance is the movement chosen for the carrot. The change in distance, where the distance between carrot and strawberry is reduced is the only one taken into consideration. This new position is updated as “carrot” on the original gridboard and the previous position is set as null.
Future Developments:	Can be refactored as a template method where specific sub methods can be created which facilitate the movement of different vegetables.
Access Modifier	Public
Return Type	String

GetNewBroccoliPosition()

Method Name:	GetNewBroccoliPosition (int xcor, int ycor)
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	This function when called, moves the broccoli by one position – diagonally left or right. This function has the current x & y coordinates of the broccoli on the gridboard as the input. If all the positions are blocked for single movement of broccolis, the broccoli can jump more than 1 cell. The distance of the broccoli is calculated for all the movement of the broccoli. The movement having the greatest change in the distance is the movement chosen for the broccoli. The change in distance, where the distance between broccoli and strawberry is reduced is the only one taken into consideration. This new position is updated as “broccoli” on the original gridboard and the previous position is set as null.
Future Developments:	Can be refactored as a template method where specific sub methods can be created which facilitate the movement of different vegetables.
Access Modifier	Public
Return Type	String

GetVegetablesPosition()

Method Name:	GetVegetablesPosition()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	Returns a list of the current position (coordinates) of the vegetables within the gridBoard. If the position is centre of the board (where the Strawberry resides), it sets the global level failure variable to true.
Future Developments:	currentVegPositions is a list of integers, in future iterations this can be refactored into a list of tuples for simpler traversal.
Access Modifier	Public
Return Type	List of integers

GetPreviousVegPosition()

Method Name:	GetPreviousCarrotPosition()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	Returns the list of the previous vegetable positions
Future Developments:	N/A
Access Modifier	Public
Return Type	List of integers

PushVegetablesBackOneMove()

Method Name:	PushCarrotsBackOneMove()
Location (Directory):SE_Fruit_Game/Assets/Scripts/BoardModel.cs
Description:	Using the previousVegPositions list, it moves all the carrots and broccolis to their previous positions.
Future Developments:	Could also be used to push other types of vegetables back to their previous position.
Access Modifier	Public
Return Type	Void

ArithmeticQuestionGenerator Class

The purpose of this class is to create arithmetic equations on basis of levels. Questions are generated based on the number of a level and are programmed to get progressively harder the with every level. The difficulty of these equations is described as follows:

- Level 1: +, -, /, * operations with even numbers up to 10
- Level 2: +, -, * operations with odd numbers up to 10
- Level 3: +, - operations with odd or even numbers up to 50

Advancements to arithmetic question generation should be added to this class and the type of question should be initialised for each level. For example, algebraic may be generated from this class when the level exceeds 6.

Other types of question packages, such as riddles, or pattern matching, should be created in a similar manner but within separate classes. These can be added as an object in the Level Controller class who will manage will type of question to output based on the chosen question package.

Class Variables

Class variable	Type	Role
upperRange	Int	Variable stores the upper range of the set of numbers used for mathematical equations. The upper range is derived using the level.
level	Int	Variable to keep track of the current level. Based on this figure, the difficulty of questions are configured.
operations	Boolean	A string array which defines all 4 operations (Multiplication, division, addition, subtraction)..

Level

Method Name	Level
Location (Directory)SE_Fruit_Game/Assets/Scripts/ArithmeticQuestionGenerator.cs
Description	Getter & Setter method for the front end controller scripts to set the level for a new game. This will affect the upper range for the generation of questions.
Future Developments	Extra checks can be placed here to ensure the value set for level is between a particular range
Access Modifier	Public
Return Type	Int

setUpperRange()

Method Name:	setUpperRange()
Location (Directory):SE_Fruit_Game/Assets/Scripts/ ArithmeticQuestionGenerator.cs
Description:	This Function is used to declare the upper range defined according to different levels.
Future Developments:	Can be refactored to set the range where both lower & upper ranges are determined.
Access Modifier	Public
Return Type	Void

generateQuestion()

Method Name:	generateQuestion()
Location (Directory):SE_Fruit_Game/Assets/Scripts/ ArithmeticQuestionGenerator.cs
Description:	<p>This is the algorithm used for generating math questions. It randomly selects the operand and the operations to be done. The function returns a tuple (question, answer) which has the question generated as a string and answer as a double.</p> <p>In case of division where the divisor is even, the numbers are not rounded off since there is an exact answer. However, in case of odd operators, the answer is rounded off to 2 decimal points.</p> <p>For now, only 2 operands and a single operation is done.</p>
Future Developments:	Larger equations can be created which make use of multiple operations. This will assess the users ability in applying BIDMAS/BODMAS concepts.
Access Modifier	Public
Return Type	Tuple of type string & double where the question is stored as a string & the answer is double.

getRandomNum()

Method Name:	getRandomNum(int lowerBound, int upperBound)
Location (Directory):SE_Fruit_Game/Assets/Scripts/ ArithmeticQuestionGenerator.cs
Description:	Function to get a random integer based on the passed in parameters
Future Developments:	N/A
Access Modifier	Public
Return Type	Int

isEven()

Method Name:	isEven(int num)
Location (Directory):SE_Fruit_Game/Assets/Scripts/ ArithmeticQuestionGenerator.cs
Description:	Function to check if the value of argument passed is even.
Future Developments:	N/A
Access Modifier	Public
Return Type	Boolean

isOdd()

Method Name:	isOdd(int num)
Location (Directory):SE_Fruit_Game/Assets/Scripts/ ArithmeticQuestionGenerator.cs
Description:	Function to check if the value of argument passed is odd.
Future Developments:	N/A
Access Modifier	Public
Return Type	Boolean

UserScore Class

This class is responsible for maintaining the scoring functionality within the game. At the time of writing the maintenance guide, the game will award 100 points for each correct answer. However, if an incorrect attempt is made, the available score is halved each time.

Future advancements to this class could include, adding scores at the end of each level completion depending on the number of correct questions and number of tiles selected before all the vegetables were found. For example, a player who answers more questions correctly could have their score bumped at the end of level completion. Likewise, a player who used less tiles before finding all the vegetables could also get their score bumped at the end of completion. In this manner, the LevelController could track the number of tiles selected, and number of correct and incorrect question answers, storing them in the StaticVariables class so they can be accessed by all scripts.

Class Variable

Class variable	Type	Role
score	Int	This variable stores the current points available to the user if a correct answer is provided.

Start()

Method Name	Start()
Location (Directory)SE_Fruit_Game/Assets/Scripts/UserScore.cs
Description	This function initializes the starting score to 0 & sets the available points to 100.
Future Developments	All configurations & changes to scoring functionality should be added here
Access Modifier	Public
Return Type	Void

incrementScore()

Method Name	incrementScore()
Location (Directory)SE_Fruit_Game/Assets/Scripts/ UserScore.cs
Description	Function increments the players total score using the available points.
Future Developments	All changes to scoring functionality should be added here
Access Modifier	Public
Return Type	Void

halveScore()

Method Name	halveScore()
Location (Directory)SE_Fruit_Game/Assets/Scripts/ UserScore.cs
Description	Function halves the available points each time an incorrect attempt is made for a question. The point is converted to an integer to ensure there are no fractional amounts.
Future Developments	In the future when different question packages are added, it would prove logical to add different scoring functionality in cases where the answer is partly correct. Therefore, if answers are partly correct, logic to award different amounts of points should be added here.
Access Modifier	Public
Return Type	Void

newQuestion()

Method Name	newQuestion()
Location (Directory)SE_Fruit_Game/Assets/Scripts/ UserScore.cs
Description	Function sets the available points to the initial value of 100.
Future Developments	All configurations & changes to initial score should be added here
Access Modifier	Public
Return Type	Void

StaticVariables Class

This class maintains the static variables which should not be re-initialised across multiple scenes. Examples of such variables include the game score and level number. This class is called from the UserScore class once the points are processed to award the player and LevelController to increment the Level or change the GameStatus.

Further expansion of this class includes key information for a given level which need to be accessed by multiple classes or managed across multiple scenes.

Such expansion of the game may include giving the user lives. Once a level is failed, they may lose a life, the number of lives needed to be accessed over multiple levels so should be stored within this class using a getter/setter method.

Following on from the example in the UserScore class, the number of tiles selected, and number of correct questions may be stored for use in the UserScore class.

Score

Method Name	Score
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/StaticVariables.cs
Description	Getter & setter method to track the current score of the game
Future Developments	N/A
Access Modifier	Public
Return Type	Int

Level

Method Name	Level
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/StaticVariables.cs
Description	Getter & setter method for the current level of the game
Future Developments	N/A
Access Modifier	Public
Return Type	Int

StartingLevel

Method Name	StartingLevel
Location (Directory)	.../SE_Fruit_Game/Assets/Scripts/StaticVariables.cs
Description	Getter & setter method for the initial starting level for the game on load
Future Developments	N/A
Access Modifier	Public
Return Type	Int

GameStatus

Method Name	GameStatus
Location (Directory)SE_Fruit_Game/Assets/Scripts/ StaticVariables.cs
Description	Getter & setter method to track the current game status of a given level
Future Developments	N/A
Access Modifier	Public
Return Type	String

SteveQuotes Class

This class stores & returns the different messages that the strawberry should output based on the players latest moves.

Arrays and methods to this class may be expanded for different Steve Quotes. These could include quotes based on finding different fruits or warnings when a vegetable is close to the strawberry.

Class Variables

Class variable	Type	Role
tileSelection	String array	List of Steve quotes to store messages on prompting user to select tiles
tileEmpty	String array	List of Steve quotes to inform the user a tile is empty
carrotFound	String array	List of Steve quotes to inform the player that a carrot has been found
broccoliFound	String array	List of Steve quotes to inform the player that a broccoli has been found
bananaFound	String array	List of Steve quotes to inform the player that a banana has been found
Free	String array	List of Steve quotes to inform that he is free & game has been completed

TileSelection

Method Name	TileSelection
Location (Directory)SE_Fruit_Game/Assets/Scripts/ SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String

TileEmpty

Method Name	TileEmpty
Location (Directory)SE_Fruit_Game/Assets/Scripts/ SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String

CarrotFound

Method Name	CarrotFound
Location (Directory)SE_Fruit_Game/Assets/Scripts/ SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String

BroccoliFound

Method Name	BroccoliFound
Location (Directory)SE_Fruit_Game/Assets/Scripts/SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String

BananaFound

Method Name	BananaFound
Location (Directory)SE_Fruit_Game/Assets/Scripts/SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String

Free

Method Name	Free
Location (Directory)SE_Fruit_Game/Assets/Scripts/SteveQuotes.cs
Description	Getter method to randomly select one of the messages from the list
Future Developments	N/A
Access Modifier	Public
Return Type	String