CAB201 Programming Principles

Assignment: Major Project – Space Race

Part B: GUI Implementation

Semester 2, 2018

Due Date: 28/10/2018 20:00

Weighting: 40%

Assessment type: Individual, with optional pairs

Specification version: Version 1.1B (th September 2018)

INTRODUCTION TO PART B

Do not start on the GUI implementation if your Console implementation cannot play at least 1 game of Space Race correctly or at least appears to play the game correctly.

Both implementations of the game must work correctly for you to gain full marks. The GUI implementation will not involve making any changes to either the **Game Logic Class** or the **Object Classes.**

This specification and its associated documents are fairly detail and do not expect to understand or remember all that is contained in them in a single read .

THE TASK FOR PART B

Having implemented the Console version of the game, you will now implement the GUI version of the same game. It is required that you use **Windows Form,** rather than any other GUI technologies such as WPF, XNA, or web pages. This means that Visual Studio for Mac and MonoDevelop will likely be insufficient – you will need to ensure that you have access to Visual Studio for Windows to do this assignment.

You must test your program in one of the CAB201 labs prior to submission. Markers will attempt to compile and run your code in that environment only. It is your responsibility to ensure that your code compiles and runs on PCs in the QUT computer labs of CAB201 as the target platform.

You will implement the GUI version as part of the **GUI Class** within the same VS2017 solution as the Console version.

Remember you are not to develop a GUI of your own design. Though you are not expected to position the various controls in the location and size correct to the last pixel, your layout should look like the layout in the **Fig 3** in the document **Screenshots of GUI**.

Building GUI Implementation

Open the **Solution Explorer**, and right-click on the ***GUI Class*** project and select ***Set as StartUp Project***.

If you want to run the Console version again, then do the same to the ***Console Class*** project.

The **GUI Class** contains the form, ***SpaceRaceForm.cs***, which is to be used to display GUI version of the game. The form is at an early stage, you can ***Start*** (run) the form to see a blank form apart from one button is the lower right-hand corner which can be clicked to close the form.

Details of the form: The form size is set to 900 x 700 pixels. This is a good size for using **Design View** to layout the form and shouldn't be changed. The form's precise size will be altered by program code that I've already written, so that each of the board's squares is displayed with the same size.

The form contains two controls so far, see **Fig 1** in **Screenshots of GUI**. Both controls are essential.

- A **SplitContainer** which divides the form into two panels: one on the left, for the board; and one on the right, for all the other controls. You can't really see the **SplitContainer**, just its two panels. The SplitContainer is docked in its parent container, the form, so that its panels occupy the whole of the form, i.e. its Dock property = Fill. (This is the default behaviour when you add a SplitContainer to a form, so you shouldn't have to do anything. But if you accidentally change this property, your form will look a mess.) This control is complete. Its position on the form is correct. **Do not move it.**
- The **Exit** button. It has an event-handler that terminates the game. This control is complete.

Now add a **TableLayoutPanel** to the left-hand panel. Refer to the document **Creating a TableLayoutPanel**. When completed, your form should look like **Fig 2** in **Screenshots of GUI. Do not proceed further until you have placed the TableLayoutPanel correctly on your form.**

Look at **Fig 3**, where there are now three **Labels**, a **ComboBox**, a **DataGridView** , three **Buttons**, a **GroupBox** containing two **RadioButtons** in the right-hand panel

- The two larger Labels, **Space Race** and **Players,** are 16pt. All three labels are **bold**.

- The additional two Buttons, **Game Reset** and **Roll Dice**, need to be given reasonable names, so that you can easily refer to them in your code later, rather than default names likes **button1**, etc. Suggest looking at the **(Name)** property of the **Exit** button.

- The ComboBox, beside the **Number of Players** Label, needs the following properties set:

    - **Size: 35, 21** is recommended
    - **Items**: left click the value **(Collection),** left click the greyed box to the right which will open the String Editor. Enter the values 2, through to 6, one per line and click OK.
    - **Text**: **6** (This will ensure that the ComboBox will initially display this value, otherwise it will be blank.)
    - Now **run** your Form, only the controls that you have added in the right-hand panel will be visible and you should be able to select anyone of the five values from your ComboBox.

- From the **Data** group in the **Toolbox**, select **DataGridView and** place it roughly below the **Players** label, do not be concern if the control extends beyond the right-hand edge of the Form, that will be fixed later.

- Now refer to the document, **Setting up a DataGridView**, noting that the first two screenshots are from an older assignment so there will be some small difference from what you will see initially. When finished run your form and it will look like the last screenshot in the **Setting up a DataGridView** document.

- From the **Container** group, place a **GroupBox** below the **DataGridView**. Change its **Text** property to **Single Step?** and set its **Enabled** property to **False** and its **BackColor** to **ControlDark.**

- From the **Toolbox** now add two **RadioButtons** to the **GroupBox,** side-by-side with the text **Yes** and **No** respectively, as well as renaming each radio button.

- Now drag the **GroupBox** borders to fit inside the panel and change its **Size** to approximately **140 by 55.** You may have to move the radio buttons around within the **GroupBox** so that the control looks reasonable.

- This **GroupBox will be disabled until advised later in this specification.** When completed, your form, in Design View, should look like **Fig 3** in **Screenshots of GUI.**

Adding code to the Space Race Form

No code from any of the other classes should be copied in the Space Race Form.

Before writing any code in **SpaceRaceForm.cs**, search for the word "**Unco**mment", which occurs in five places and follow the instruction that occur given there. In one place, the instruction in the code say to delete a line, as well as doing the uncommenting. Make sure that this class still compiles without error.

The constructor of **SpaceRaceForm** consists of a series of methods calls, currently each call is commented out. You can now uncomment the first three calls, ***Board.SetUpBoard()***, ***ResizeGUIGameBoard()*** and ***SetUpGUIGameBoard()***. Compile (Build) the ***GUI Class*** and then run.

If successfully, the board will be displayed with all 56 squares coloured Slate Grey colour and a number in the bottom right-hand corner of each square, though the numbering will appear to be in some random pattern, **Screenshots of GUI, Fig 4**.

Trace the call to ***SetUpGUIGameBoard*** then through a number of method calls to the method ***MapSquareNumToScreenRowAndColumn*** which maps an individual square's number to the correct row and column position on the **TableLayoutPanel**. The top left corner of the **TableLayoutPanel** is position (0,0) and the bottom right corner is (6, 7).

Whereas the board "Start" square is to be in the bottom left corner and the "Finish" square in the top right corner, with numbering going left-to right and then right-to-left on alternate rows. Implement code within the method to achieve this correct mapping. **Do not proceed further until the board is displayed with squares in the correct sequence as per Screenshots of GUI, Fig 5**.

The **SquareControl** class is used to display a square on the GUI game board. Each square on the board, has a **Square** object and its corresponding **SquareControl** object. A **SquareControl** object is a **PictureBox** object which will enable each square to contain a picture as well as displaying a player's token when the player is on that square. It is not necessary that you fully understand this class. **This class is complete.**

Uncomment the last line on the constructor method in **SquareControl** class, recompile and the board should be displayed with appropriate images in each square as per **Screenshots of GUI, Fig 6**.

Now uncomment the calls to ***SetUpPlayersDataGridView()*** and ***DetermineNumberOfPlayers()*** in the constructor of **SpaceRaceForm.** The body of ***DetermineNumberOfPlayers()*** needs to be implemented. A suggested algorithm is provided.

One difference between the GUI implementation and the Console implementation is that for the GUI each player has a token colour (**playersTokenColour**). In ***SetUpPlayers*** in the ***Game Logic Class*** add a line to set each player's **playersTokenColour** to the corresponding colour in the array **playerTokensColours** of **SpaceRaceGame.**

Now uncomment the call to ***SpaceRaceGame.SetUpPlayers(),*** recompile and run the form, the board will look like **Fig 7** of **Screenshots of GUI**.

The method ***UpdatePlayersGuiLocations*** of **SpaceRaceForm** needs to be completed along with the method ***GetSquareNumberOfPlayer***. Once done uncomment the call to ***PrepareToPLayGame().***

The method ***PrepareToPlayGame()*** contains a single statement which is sufficient for completing the look of the GUI game board for the start of a game. Recompile and run the form. Do not proceed until your form looks like **Fig 8** of **Screenshots of GUI**.

Create an event handler for the **Roll dice** button which will call a method to "play a round". Remember to call ***UpdatePlayersGuiLocations*** to "remove the players' tokens" before calling ***PlayOneRound*** of ***SpaceRaceGame*** and once the round is completed, call ***UpdatePlayersGuiLocations*** to "add the players' tokens" and then calling ***UpdatePlayersDataGridView()*** to ensure that the players' information within the DataGridView is updated accordingly.

Once you have written "play a round" method, run your form and play one round of the game, noting the position of the six tokens and amount of fuel for each.

Now you should make the **Console Class** the **"StartUp Project"** and play one round to ensure that the code of GUI implementation has not affected the Console implementation and that the six tokens end at the same position and fuel after one round as in the GUI.

Finer details of the Space Race Controls

(a) The **Reset** button is disabled at the start of a game. It will be enabled at the end of any round including when the game is finished. This means at the end of any round, if the user clicks the **Reset** button the players' tokens return to the Start square and a new game is ready to be played.

(b) The **Exit** button is enabled at the start of a game, disabled during any round and enabled at the start of any round.

(c) The **Players DataGridView** is enabled at the start of the game to allow the user to enter other values for the players' names to replace the default names. As the other columns are read-only, the user is unable to change those values. The **DataGridView** is disabled as soon as the **Roll dice** button is clicked to play the first round. It remains disabled until a new game is commenced.

(d) The **ComboBox** is enabled at the start of a game. If the user selects any number of players, the **ComboBox** is disabled and the correct number of tokens for the newly selected number of players appear on the Start square. If the user decides not to change the number of players, the **ComboBox** will be disabled as soon as the **Roll dice** button is clicked to play the first round. In either case the **ComboBox** will remain disabled until a new game is commenced.

    Note that the **Players DataGridView** will still show all six players after the user selects a smaller number of players, but during the game only the values for the selected number of players will be updated. See **Fig 11 & 12** of **Screenshots of GUI**

(e) The **Roll dice** button is enabled at the start of any round and disabled during a round of play as well as being disabled when the game is finished. It will be enabled after the **Reset** button is clicked.

(f) The user can resize the game by dragging on its borders. (Don't worry if doing this causes some of the square numbers to display strangely.) To support this, most controls in the right-hand panel should have default set of values for their **Anchor** property, which is:

    Anchor = Top, Left

    So that the three buttons stay in the bottom right corner, each should have:

    Anchor = Bottom, Right

(g) When the game finishes, a **MessageBox** will show which players have finished the game, similar to Console version. There is no need to report the details of all players at the end of the game as this is provided by the **DataGridView**. See **Fig 9 & 10** of **Screenshots of GUI**.

(h) You should now ensure that your GUI version can play multiple games according to the specification so far. Always checking that the Console version still plays correctly. **Do not proceed if both implementations are not correct.**

(i) At the start of the GUI version, now make the **GroupBox** enabled and the **Roll dice** button disabled. So, at the start of a game the user chooses between **Single Step** mode or not. In **Single Step** mode only one player's token moves each time the **Roll dice** button is clicked, so six clicks would be required to complete a single round for a six-player game. If user selects the **No** radio button all tokens will move on a click of the **Roll dice** button. See **Figs 13 – 16** of **Screenshots of GUI** for two rounds of **Single Step** mode.

(j) Do not use the default event handler for either radio button (**CheckedChanged**) but use **Click** instead.

(k) Once either radio button has been clicked, the **GroupBox** is disabled and the **Roll dice** button is enabled. At this stage the **ComboBox** and **DataGridView** controls are still enabled. The **GroupBox** will remain disabled until a new game is commenced.

This change in the specification will require you to add additional methods or substantial redesign one or more existing methods in both the **SpaceRaceForm** as well as in the **SpaceRaceGame.**

One example of an additional method is as follows: in **SpaceRaceForm,** do not change the method **UpdatePlayersGuiLocations(…)** but rather add another method like it which would "update a single player's location". This new method would need to be passed which player in addition to the **TypeOfGuiUpdate** to occur. This will enable the board to be updated after a token is moved in **Single Step** mode.

Ensure that any change made to the **SpaceRaceGame** does not adversely affect the Console Version from playing multiple games successfully.

The challenge is to complete what you can before the due date. It is better to hand in a working project which does something correctly than one which does not play the game correctly.

Academic Integrity

Please read and follow the guidelines in QUT's Academic Integrity Kit, which is available from the Blackboard site on the Assessment page.

Students are reminded of the following MOPP statement C/5.3.7 which I have paraphrased as follows:

"***To assist in identification of potential breaches, unit/course coordinators may require students to authenticate their learning on the assessment item (for example, by showing notes/drafts/resource materials used in the preparation of the item, or by undertaking a viva or practical based exercise***)".

Programs submitted for this assignment will be analysed by the MoSS (Measure of Software Similarity) plagiarism detection system (http://theory.stanford.edu/~aiken/moss/).

Final Comment

Though all care has been taken in the production of this specification and related documentation, there may be a need to notify by email any alterations/clarifications to this specification and related documentation. **SO, CHECK YOUR QUT EMAIL DAILY**