

Week 03 (Programming Workshop)

Implementing Title Screen and Game Menu

Module Leader: Dr. Sud Sudirman

James Parsons Building, Byrom Street

Room BS/7.27

s.sudirman@ljmu.ac.uk

|  |
| --- |
| Liverpool John Moores University |

|  |
| --- |
| Contents  [Implementing Title Screen 3](#_Toc474333393)  [Game Menu 8](#_Toc474333394)  [Additional Notes 22](#_Toc474333395) |

# Implementing Title Screen

**Introduction**

In this section, we are going to create a new class that displays a title screen as shown in Figure 1 below. The title screen will be a simple image display as a background and a simple message to press any key to continue. The message will be a flashing bitmap.



**Figure 1.** The Title Screen

You need to have completed Workshop 02 from last week to start working on this workshop.

**Expected Learning Outcome**

1. Able to load an image file using SFML
2. Able to display bitmap images to the screen as SFML Sprite.
3. Able to alter display properties of the rendered images.

**Creating SFML\_TitleScreen class**

1. Using the steps detailed in week 01, create a duplicate of the solution from last section of last week workshop and rename the project from **SFML\_4109COMP\_Week\_02** to **SFML\_4109COMP\_Week\_03**.
2. Open the new solution with Visual Studio.
3. Add a new class to the project by right-clicking on the project name in the Solution Explorer and select Add » Class. Select C++ Class from the list of template classes and press Add.
4. In the next window, enter **SFML\_TitleScreen** in the Class name textbox. This class will be derived publicly from the **sf::Drawable** class, so add sf::Drawable in the Base class textbox. Make sure the Access textbox is set to **public**. Press the Finish button.
5. Two new files should be created namely **SFML\_TitleScreen.h** and **SFML\_TitleScreen.cpp**. Open SFML\_TitleScreen.h and delete the #include line:



And replace it with:



1. Change the prototype of the default class constructor to:



This will allow the constructor to take the texture file name as well as the dimension of the window.

1. Add a **public** member function update() with the following prototype

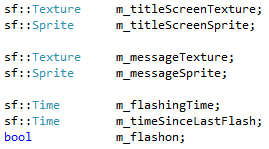


1. Add a **private** member function draw() with the following prototype



This function is needed when the object of a class that has been derived from sf::drawable is passed on to sf::window.draw() function.

1. Declare also the following private member attributes:



The member attributes names should be self-explanatory. However, if you are unsure, please ask the tutor about the purpose of any of them.

1. Open **SFML\_TitleScreen.cpp**. Change the class constructor interface to match the prototype in step 6.
2. Inside the constructor, load the image from texturefilename to m\_titleScreentexture.



1. Assign this texture to m\_titleScreenSprite



1. Set the texture rectangle to cover the entire texture area



1. Set the sprite’s position to the top-left corner



Note that the default sprite’s origin is the top-left corner of the sprite. Hence, the above code will align the top-left corner of the sprite to the top-left corner of the screen.

1. Next, we load the image that we want to use to display the “Press any key” message. At the moment we hard-coded this to continue.png.



1. Assign this texture to m\_messageSprite



1. We want to place this message bitmap at (½ width, ¾ height) on the screen.



And we want it to be centred on the texture’s area.



1. Set the texture rectangle to cover the entire texture area.



1. We can set the initial tint of the image to red colour

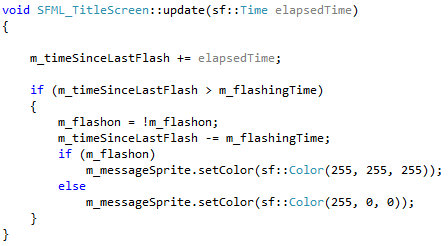


1. Next, we set the parameters to control the flashing of the message sprite. First, we reset the time counter to zero and set the flashing time to 0.2 seconds.

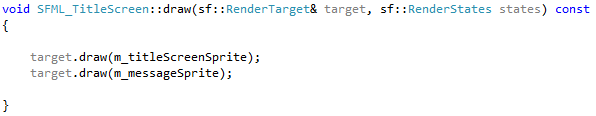


This completes the implementation of the class constructor.

1. As for the class deconstructor, we leave it blank as before.
2. The class’ update() member function will essentially change the tint of the message sprite from red to white every 0.2 second (as was set earlier). Here is the implementation:



1. The class’ draw() member function simply draws the m\_titleScreenSprite and m\_messageSprite (in that order).



**Using the SFML\_TitleScreen class**

We are going to use the SFML\_TitleScreen class to display a more interesting title screen than just a simple text as before.

1. Open **SFML\_Application.h**. Add the following #include line



1. Add the following private member attribute to the SFML\_Application class



1. Open **SFML\_Application.cpp**. In the SFML\_Application class constructor add the following to the Member Initializers List



This will create the title screen object using the specified image.

1. This would be a good time to download the **title-screen.png** and **continue.png** from the Blackboard and save them to the **/Media/Textures/** folder.
2. Still in the SFML\_Application.cpp file, find the **update()** function. Find the part in the switch-case statement for TITLESCREEN



And replace the line



with



1. Find the **render()** function and replace the statement



with



1. You should now be able to build the program without any errors. If there are any, try to fix them yourself. If you need assistance let the tutor know.
2. Run the program and observe the result.

**Additional Challenges**

1. Try designing your own title and message images using any image editing software. You can use the internet to browse any images for inspiration. The theme for the game would be zombies and horror.
2. Use the images in the application you created.

# Game Menu

**Introduction**

In this part of the workshop we are going to add a menu to our application. The menu will look like Figure 2 below and will replace the bland on-screen text message we currently have.



**Figure 2**. The look of the Game Menu we will be creating

You need to have completed the previous section of this workshop before attempting this one.

**Expected Learning Outcome**

1. Able to display multiple images on the screen at different locations
2. Able to display texts with different colour to simulate highlight
3. Able to implement a simple menu system using SFML

**Creating simple SFML\_GameMenu class**

1. Add a new class to the project by right-clicking on the project name in the Solution Explorer and select Add » Class. Select C++ Class from the list of template classes and press Add.
2. In the next window, enter **SFML\_GameMenu** in the Class name textbox. This class will be derived publicly from the **sf::Drawable** class, so add sf::Drawable in the Base class textbox. Make sure the Access textbox is set to public. Press the Finish button.
3. Two new files should be created namely **SFML\_GameMenu.h** and **SFML\_GameMenu.cpp**. Open SFML\_GameMenu.h and replace the #include line



with



1. We are going derive SFML\_Game Menu publicly from sf::Transformable as well so that we can scale the menu’s size.



1. Change the class default constructor to the following



The constructor will now also take a filename that contains the properties and menu items for our menu. The file is a text file which will be parsed by our program for the information. We will cover more about the structure of this file shortly.

1. Open the working folder using File Explorer and go to the /Media sub-folder. Create folder in the /Media sub folder called **MenuInfo**. Inside this folder, create a new text file called **main-menu.txt** with the following contents.

Background image filename



Logo image filename

The first line is the filename (including its relative path) of the background image. The second line is for the logo.

Important: You need to press **Enter** to add a **Carriage-Return** at the end of the second line once. This will prevent our program to fail reading the last line when parsing the file.

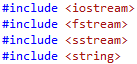
1. Save the file. We will leave this file for now. We will revisit it again when we are initialising our game menu object.
2. Back in Visual Studio, inside SFML\_Game.h add a private member function draw()



1. Add also the following private member attributes



1. Open **SFML\_GameMenu.cpp** and add the following #include statements



1. Find the pre-generated class constructor implementation. Change the operand to match the prototype as shown in step 39.
2. Inside the class constructor, declare an input file stream object with menufilename as input.



1. Create a check if the file is opened correctly:

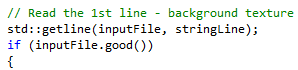




1. Inside the above pair of curly brackets, declare a text string object. This will be used to store each line in the file as it is parsed one line at a time.

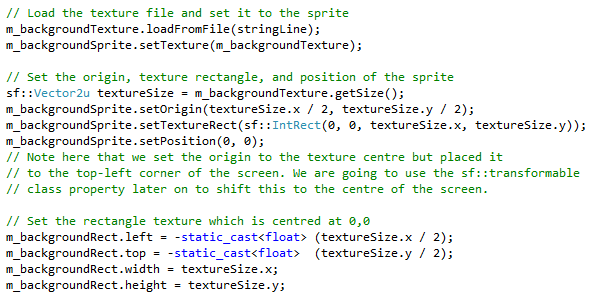


1. Now we read the first line, which is the filename of the background texture image we want to use.

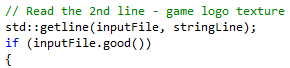




1. Inside the above pair of curly brackets, add:

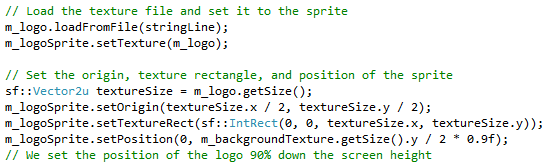


1. Now, we read the second line, which is the filename of the logo texture image we want to use.

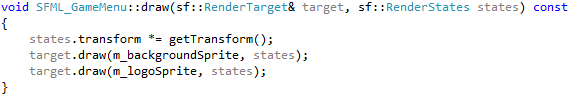




1. Inside the above pair of curly brackets, add:



1. Implement the draw() function by first getting the transform state and use it when drawing the background and logo images.



**Using the SFML\_GameMenu class**

1. Open **SFML\_Application.h** and add the following #include line



1. Add SFML\_Menu object as a **private** member attribute



1. Open **SFML\_Application.cpp** and add the following #include lines



1. In the SFML\_Application class constructor add the following to the Member Initializers List



1. At the end of the constructor, we set the scale of the menu to 85% and its position to the centre of the screen.



1. Find the render() member function. Find the part in the switch-case statement for MAINMENU



Replace the line that draws the on-screen message



With rendering the main menu object.



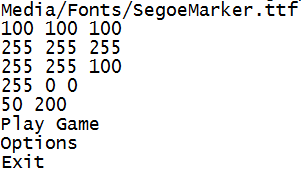
1. You should now be able to build the program without any errors. If there are any, try to fix them yourself. If you need assistance let the tutor know.
2. Run the program and observe the result.

**Adding Menu Item**

In this part, we are going to add three menu items. They are text strings which are highlighted as the user makes their selection. The menu item texts are Play Game, Option, and Exit. This information will be stored in the main-menu.txt we created earlier.

1. Open **main-menu.txt** and append the following lines at the end (just after the first 2 lines we had previously)

Font to use when rendering the text



Menu item texts (there are 3 of them)

Border width and height (in pixels)

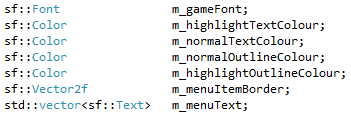
Highlighted Outline Colour (red)

Highlighted Fill Colour (dark yellow)

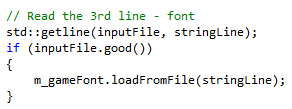
Normal Outline Colour (white)

Normal Fill Colour (grey)

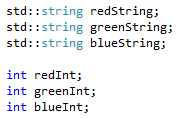
1. Save the file
2. Open **SFML\_GameMenu.h** add the following private member attributes



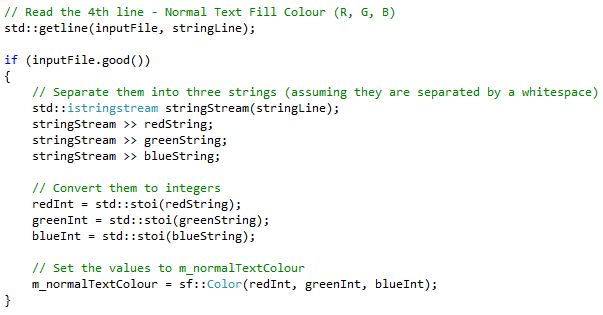
1. Open **SFML\_GameMenu.cpp**. In the class’ constructor add the following line to read the third line – which is the font.



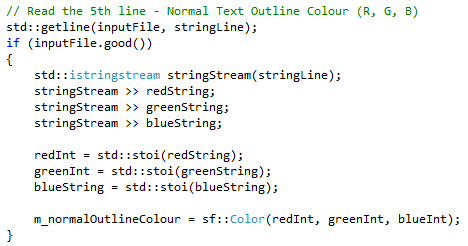
1. Next we are going to read four lines containing the RGB values of the text colours. Declare some variables to temporarily store the information.



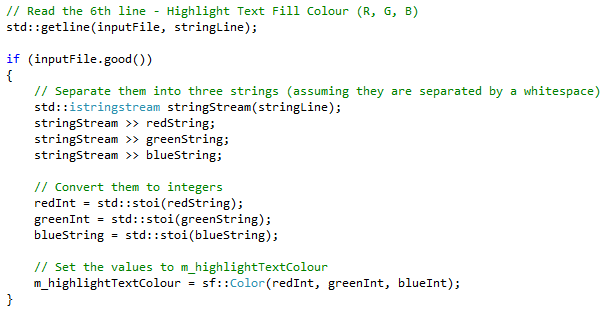
1. Now, we read the 4th line to get the normal fill colour.



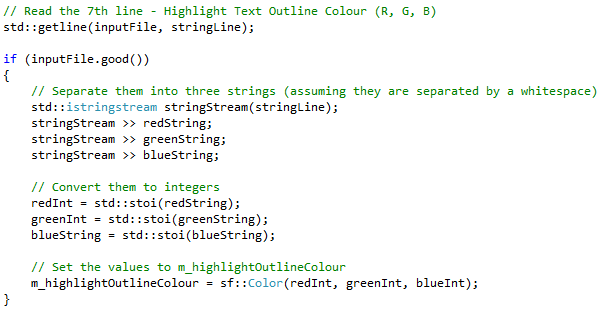
1. Do the same for the 5th line to get the normal outline colour



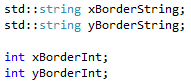
1. Do the same for the 6th line to get the highlighted fill colour



1. And the same for the 7th line to get the highlighted outline colour



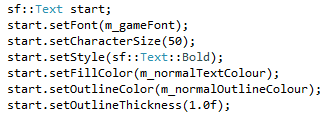
1. Next we are going to read the 8th line for the border width and height. Declare some variables to temporarily store the information.



1. Now, we read the 8th line to get the border width and height.



1. Now, we are going to read the remainder of the lines (there will be three of them) to get the menu item texts. First, we set the text properties as follows:

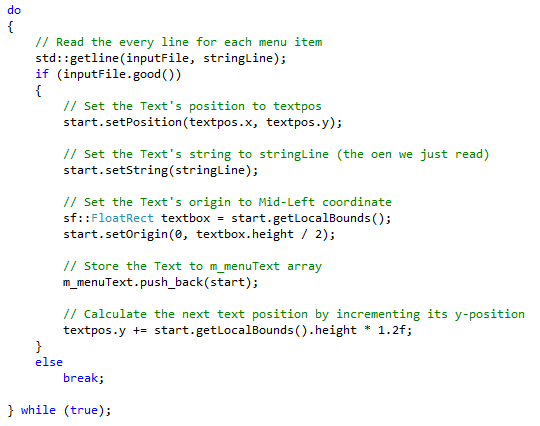


1. Then we set the position (for the first menu item)





1. Then we iteratively read the line, for each we going to calculate the new position, assign the text (from the read line), and store it in the m\_menuText array.



1. Close the file



You need to have the above line **inside and at the end** of the pair of curly bracket when inputFile is properly opened.

1. Since, we have populated the m\_menuText array with items, we need to make sure that it is cleared when the object is destroyed. Add the following instruction to the class **deconstructor**.



1. Lastly, you need to add instructions inside the **draw()** function to render each menu item.

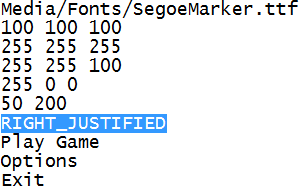


1. You should now be able to build the program without any errors. If there are any, try to fix them yourself. If you need assistance let the tutor know.
2. Run the program and observe the result.

**Text Alignment of the Menu Item**

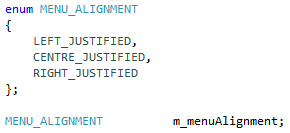
In this part we are going to add a feature to let us decide whether the menu items will be displayed on the left, at the centre, or on the right of the menu. We will let the program know by inserting one extra line in the main-menu.txt we created earlier.

1. Open **main-menu.txt** and insert the line as shown below:

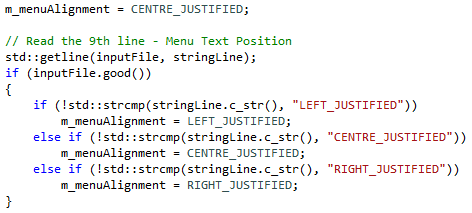


We add this to inform the program to display the menu text on the right

1. Open **SFML\_GameMenu.h** add the following enumeration and its object as private member attributes.



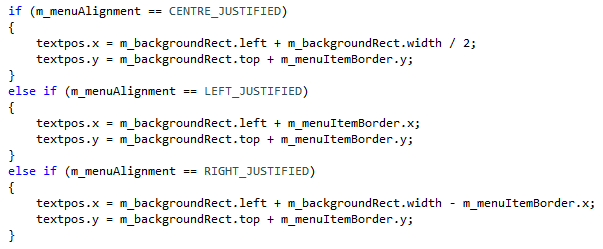
1. Open **SFML\_GameMenu.cpp**. In the class’ constructor add the following line after the part where we completed parsing the 8th line.



1. Then we need to replace the instructions that we wrote to calculate the menu item’s location



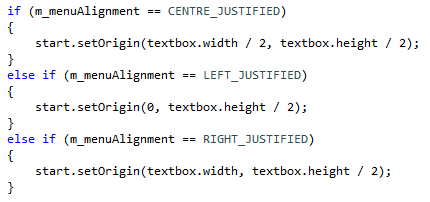
With another that depends on the alignment.



1. We also need to replace the instructions that we wrote to set the menu item’s origin:



with



1. You should now be able to build the program without any errors. If there are any, try to fix them yourself. If you need assistance let the tutor know.
2. Run the program and observe the result.

Next we are going to modify the program to allow us to select a menu item (the currently selected item will be highlighted with a different colour than the others).

1. Open **SFML\_GameMenu.h** add the add the following private member attribute to store the index of the currently selected menu item



1. And the following private member attributes to store previous state of Down, Up, and Return key presses.



1. Add public member function update()



1. Open **SFML\_GameMenu.cpp**. In the class constructor, after the loop that reads the menu items (shown below)



Add the following instructions that initialise the highlight index and set the colour of the highlighted text.



1. We will also set initial values for the previous key press state variables.



1. Now, we are going to implement the update() member function. Let’s start with a blank skeleton of the function.

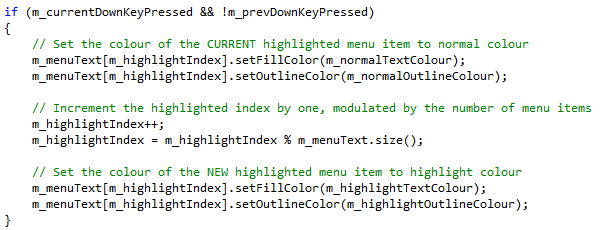




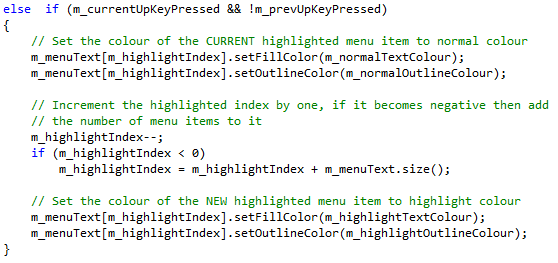
1. Inside curly bracket pair get the key press states



1. And check if the **Down Arrow** key has been **recently** pressed



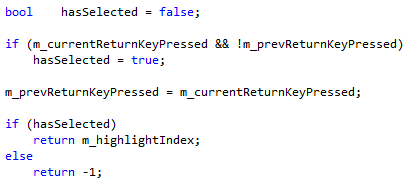
1. Else check if the **Up Arrow** key has been **recently** pressed



1. Save the current states of the key presses to be used in the next iteration of the loop



1. Next we handle the event when the user presses the Enter key. This is when the user has decided to make the current selection as his choice.



The update() function will return the index of the selected menu item if the user has pressed the Enter key, otherwise it will return -1. We can then use this in our main application to check the user selection and act according to it.

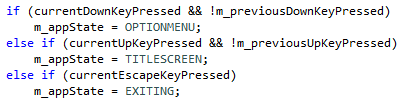
1. Open **SFML\_Application.cpp**, and find the update() member function. Add a declaration of an integer variable for the menu selection



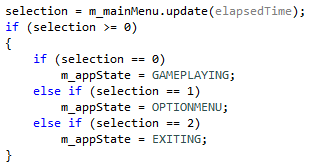
1. Find the part in the switch-case statement for MAINMENU



And replace these instructions



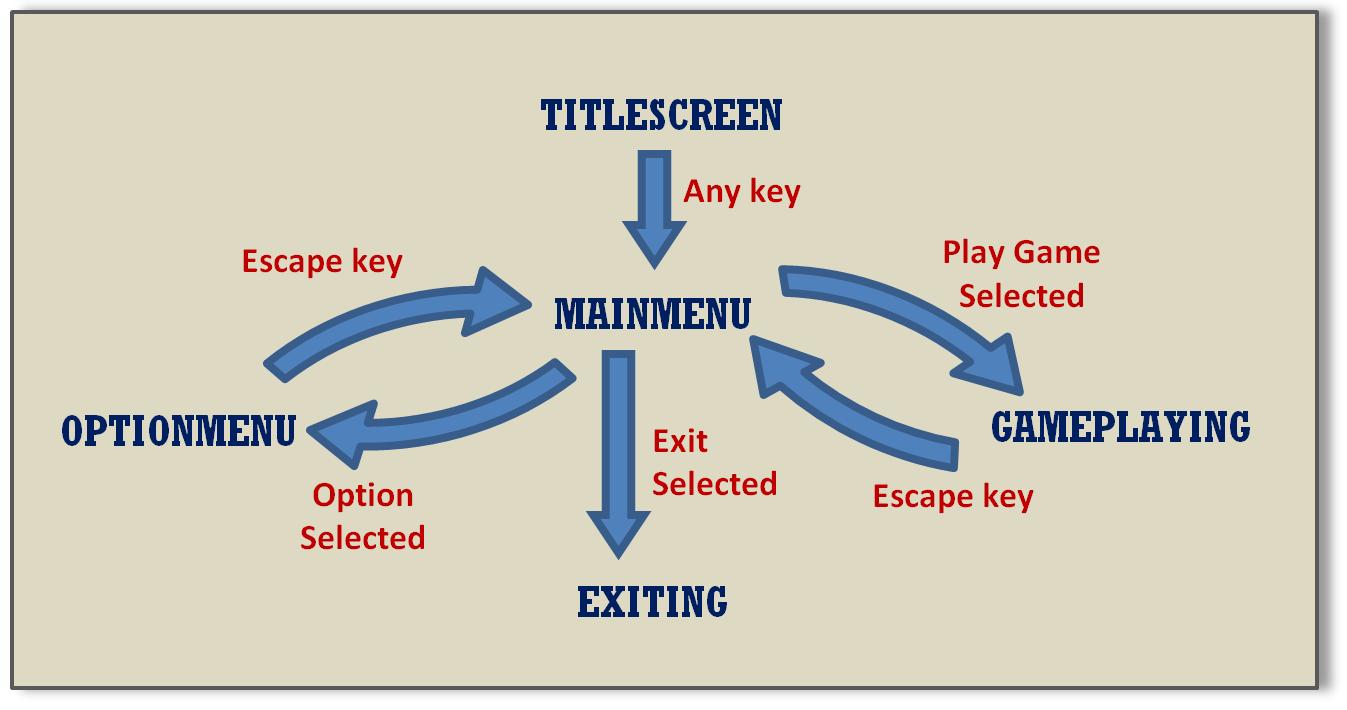
With



1. You should now be able to build the program without any errors. If there are any, try to fix them yourself. If you need assistance let the tutor know.
2. Run the program and observe the result.

**Additional Challenges**

Modify the program such that its state logic flow follows the FSM diagram shown below:



# Additional Notes

Please use this space to add your own notes.