



Introduction to Programming

Pass Task 5.1: Circle Moving in C

Overview

In Topic 3 you developed a **Circle Moving Program** using Pascal. To check your reference sheet, and to see how similar Pascal and C programs really are, you will now implement another version of this program using C.

Purpose: Start to learn a new language.

Task: Create the circle moving program using the new language, and your reference sheet.

Time: This task should be completed before the start of week 9.

Resources:

- Programming Arcana
- Google
- Your language reference sheet
- Swinburne CodeCasts ([YouTube Channel](#), [iTunesU](#))
 - [Learning a new language](#)
 - [Introducing Objects](#)

Submission Details

You must submit the following files to Doubtfire:

- Your circle moving code in C/C++

Make sure that your task has the following in your submission:

- The program must move the circle, and ensure it remains on the screen.
- Code must follow the C coding convention used in the unit (layout, and use of case).
- The code must compile and the screenshot show it working.

Instructions

Here is a list of C functions and procedures available in C to help you with this program.

| <i>Function / Procedure</i> | <i>Does</i> |
|---|--|
| <code>void open_graphics_window(title, width, height);</code> | Opens a new window, with indicated title and size |
| <code>void load_default_colors ();</code> | Loads the default colours such as COLOR_RED. |
| <code>void clear_screen ();</code> | Clears the screen to a color, e.g. COLOR_WHITE |
| <code>void delay (time);</code> | Delays for a number of milliseconds |
| <code>void process_events ();</code> | Listens for user input - required for any input |
| <code>int screen_width (); int screen_height ();</code> | The width and height of the window |
| <code>float mouse_x(); float mouse_y();</code> | X or Y location of the mouse |
| <code>bool mouse_clicked(button)</code> | Was the mouse button clicked? Buttons are LEFT_BUTTON and RIGHT_BUTTON |
| <code>bool window_close_requested();</code> | Has the user asked to close the Window? |
| <code>bool key_down(key);</code> | Is the key currently held down? Keys are in the format A_KEY, NUM_1_KEY, LEFT_KEY etc |
| <code>bool key_typed(key);</code> | Was the key typed? (Pressed then released) |

1. Download the starter code for this task.

Note: This contains the C/C++ version of SwinGame.

2. Extract the zip file to your code directory (e.g. Documents/Code)
3. Rename the **Project Template** folder to **CharacterMoving2**
4. Open a **Terminal** window and navigate to your *CharacterMoving2* directory.
5. Write the code to implement a basic *SwinGame* program using the following code.

```
1  #include <stdio.h>
2  #include "SwinGame.h"
3
4  int main()
5  {
6      open_graphics_window("Circle Moving 2 - C/C++", 800, 600);
7
8      clear_screen(ColorWhite);
9      refresh_screen(60);
10
11     delay(5000);
12
13     return 0;
14 }
```

6. Switch back to the Terminal and compile and run your program.
7. As with Pascal, the window closes after 5 seconds when the program's instructions end. Implement an **event loop** to allow you to control the life of the program.
 - The event loop will be located in Main, and will loop until the user closes the window.
 - **Process Events** needs to be called *once* each event loop to update *SwinGame* with the actions that have occurred since the last time through the loop.

Procedure: **Main**

Steps:

- 1: **Open** a **Graphics Window** with title 'Character Moving' that is 800x600
- 2: Do
- 3: **Process Events**
- 4: While **Window Close** is not **Requested**

8. Switch back to the Terminal and compile and run the program. It should now remain open until you close the window.
9. Create a constant named **CIRCLE_RADIUS** and set it to 90.

10. Alter Main to draw a circle on the screen, using variables for the circle's x and y location.

```
-----  
Procedure: Main  
-----  
Local Variables:  
- x, y: Single data for the circle's location  
Steps:  
1: Open a Graphics Window with title 'Character Moving'  
   that is 800x600  
2: Assign x the value 400  
3: Assign y the value 300  
4: Do  
5:     Process Events  
6:     Clear the Screen to COLOR_WHITE  
7:     Fill a Circle using COLOR_GREEN, at location x,y  
       with a radius CIRCLE_RADIUS  
8:     Refresh the Screen limiting it to 60 FPS  
9: While Window Close is not Requested
```

Note: To ensure a consistent game speed you can use RefreshScreen(60) to limit the refresh rate to 60 frames per second (FPS)

WARNING: C/C++ is **case sensitive!** So this means you really need to following the case recommendations. So make sure you use **int main()** in C for example.

11. Switch to the Terminal, compile and run the program. You should be able to see a green circle in the centre of the screen.

- The x and y variables in Main store all of the data for this game: the location of the circle. As these are the only variables in the "game" (which is being run by the steps in the Main procedure), these are the only things that can change.

The next step will involve using **if** statements to selectively run sections of your code.

12. Alter Main to update the x variable when the user is holding down the arrow keys. The following pseudocode shows the changes for moving left and right. Include the code to move in all four directions.

```
-----  
Procedure: Main  
-----
```

Local Variables:

- x, y: Single data for the circle's location

Steps:

```
1: Open a Graphics Window with title 'Character Moving'  
   that is 800x600  
2: Assign x the value 400  
3: Assign y the value 300  
4: Repeat  
5:     Process Events  
  
6:     if the LEFT_KEY Key is Down then  
7:         Assign x, the value x - 1  
8:     if the RIGHT_KEY Key is Down then  
9:         Assign x, the value x + 1  
  
10:    Clear the Screen to ColorWhite  
11:    Fill a Circle using ColorGreen, at location x,y  
       with a radius 90  
12:    Refresh the Screen limiting it to 60 FPS  
13: Until Window Close is Requested
```

Tip: C/C++ has operators to add/subtract one from a variable; use ++ and --.

13. Switch back to the Terminal and compile and run the program. You should be able to move the circle using the left and right arrow keys. If you keep your finger on the one arrow key long enough, the circle will disappear off the edge of the screen.

14. Now, adjust the if statements so that the program will ensure that the circle remains on the screen.

Tip: To find the width of the current screen you can call the *SwinGame* function `screen_width()`. Similarly, C/C++ has a `screen_height()` function to give you the height of the screen.

15. Switch to the Terminal. Compile and run the program, and test that you cannot move the circle off the screen to the left or right.

Compare your final program to your Pascal program... Notice how similar they are. If you can see that they are basically the same you are well on the way with your understanding of programming.