C++ Starter (Games Track) - Lesson 2

Penglais Competitive Programming Club

TL;DR

We introduce two fundamental features of C++: variables and conditional blocks.

Lesson

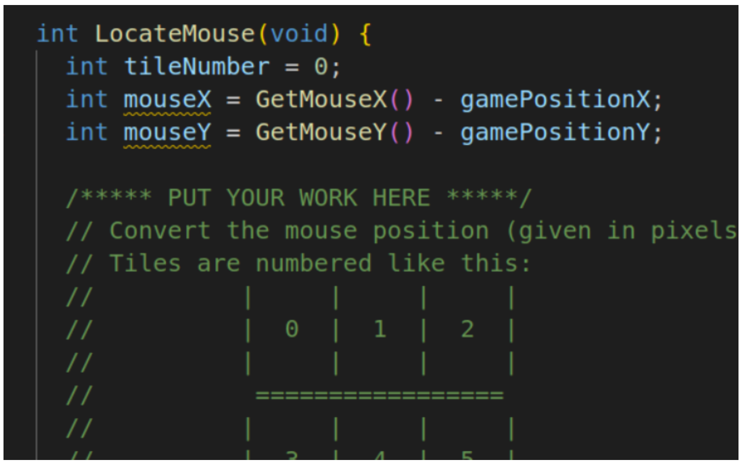
Once you have launched VS Code, please follow the following procedure:

1. Clone the lesson repository. Remember from Lesson 1 that you do this from the Source Control icon on the bar to the left (looks vaguely like a gear shift pattern). Once you have entered the Source Control pane, click the three dots and choose “Clone”. The URL for the repository is

   https://github.com/hightechhasbeen/raylib_lesson_two

   You should clone the repository in a new folder and then enter the project.

2. Build the application (CTRL+SHIFT+B) and run it (CTRL-F5). You’ll notice some differences but also that the puzzle does not work. The assignment today is to process mouse clicks.
3. Open the Document Explorer (Top Icon on the left) and open the **src > main.cpp** file. You will be making changes to this file in a moment. Scroll down until you see the green comments with the words **PUT YOUR WORK HERE**.



```
int LocateMouse(void) {
  int tileNumber = 0;
  int mouseX = GetMouseX() - gamePositionX;
  int mouseY = GetMouseY() - gamePositionY;

  /***** PUT YOUR WORK HERE *****/
  // Convert the mouse position (given in pixels
  // Tiles are numbered like this:
  //          |     |     |     |
  //          |  0  |  1  |  2  |
  //          |     |     |     |
  //           =================
  //          |     |     |     |
  //          |  3  |  4  |  5  |
```

Figure 1 - Navigate to this point in main.cpp

4. We will start analysing the program line by line starting with

   int LocateMouse(void) {

This line starts a function, which is a topic we shall return to later. For the moment, you can think of a function as a way to organize the program into smaller units that each perform a specific task.

5. The next three lines declare (create) **variables**.

```
int tileNumber = 0;
int mouseX = GetMouseX() - gamePositionX;
int mouseY = GetMouseY() - gamePositionY;
```

Variables are specific bits of information your program will process. They usually contain text or numbers or complex structures whose components are other variables.

The first bit, *int*, tells the compiler what kind of variable you want. This is a characteristic of C++ that is sometimes hard for beginners but is beneficial in the long run because it makes you think about the kinds of information you are processing. Here, we are asking the computer to set aside some memory for three integer numbers. This means we can use them to store things like 371 or -65529 but not 2.71828 or "Iestyn".

You only need to put the *int* in the declaration of the variable — that is, the first time you mention it in its scope. That is because the declaration is what asks the compiler to make a space to store the information associated with the variable. If you use *int* a second time with the same name, you will get an error because the compiler thinks you are trying to call two different variables by the same name.

After *int* you will see the name of the variable. Here, the names are *tileNumber*, *mouseX*, and *mouseY*. Good programmers try to choose names that make clear what the information is: here, we are going to select one of the tiles of our puzzle based on the X and Y coordinates of the last mouse press.

After the variable name you can end the declaration with a semicolon or you can combine the declaration with an *assignment*. An assignment puts information into the space created by the declaration — that is, it gives the variable a value. It is generally considered good practice in C++ to combine declarations with assignments because a variable is usually undefined until it is assigned. This means that if you accidentally use it, you will get unpredictable results. You assign a value to a variable with the equals sign. Here, we assign each variable a value in the same line that we declare it.

To the right side of the equals sign is an expression that gives the variable's new value. This must be of the correct variable type (that is, you can't assign 3.14 to a variable of type *int*) or the compiler will give you an error. The expression can contain a literal value (as in *int tileNumber = 0;*) or other variables and tokens (as in *int mouseX = GetMouseX() - gamePositionX;*). The computer will evaluate the expression and put the result into the memory specified by the variable name.

An assignment (like most lines in C++) must end with a semicolon.

Consider, for a moment, what the second line does. We are finding the X-component of the mousepress, as given by the difference between the value of *GetMouseX()* and *gamePositionX*. This diagram may help:

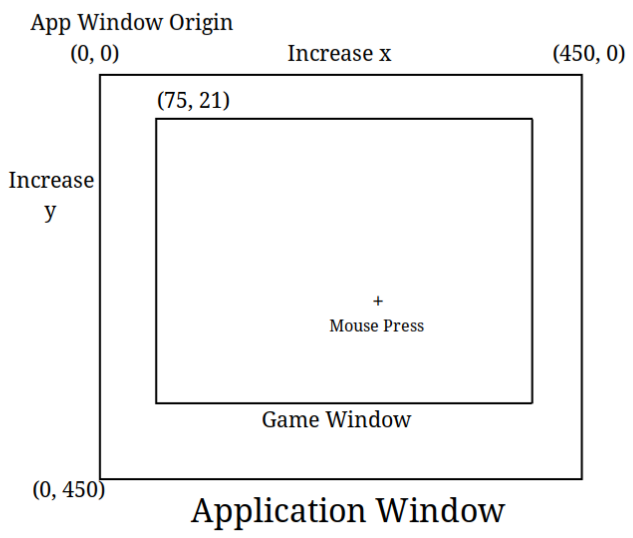


Figure 2 - Coordinate Schema

GetMouseX() gives us the X coordinate of the mouse press, relative to the application window origin. Say that this is at 300. We then subtract the X coordinate of the origin of the game window (75), so that mouseX is set to 300 - 75 = 225. We do this because we want the relative position of the mouse press from the origin that directly affects the puzzle tiles.

6. Your job is to convert the coordinates into a tile number so that we know which tile the user wants to move. You can do this by knowing that the size of each tile is 100x100. For example, if you want the column of the chosen tile, you could write

   ```
   int tileColumn = mouseX / tileWidth
   ```

   (you don't have to do it this way, by the way). Here, you are relying on integer division: if mouseX is 270 and tileWidth is 100, then tileColumn would be 270 / 100 = 2 (not 2.7).

   Use mouseX and mouseY to determine which of the tiles 0 through 8 is being clicked. Assign the value you find to the variable tileNumber.

   You can build and test your changes. You should see the tile you click change colour.

7. Once you get the basic task of tile determination working, check what happens when you click outside the game window. Are tiles changing colour anyway? How can you fix this? Can you see that you must check to make sure the clicks are in the game window and not outside?

8. To see if the clicks are outside the game window, you can compare the coordinates to the boundaries of the screen. You should do this in an “if” statement:

```
if (mouseX < gameWidth) {
        tileNumber = tilesPerRow * (mouseY / tileHeight) + (mouseX / tileWidth);
}
```

   When the program gets to this point, the computer will compare the value of mouseX to gameWidth and only execute the lines between the curly brackets if mouseX is smaller.

   Here, we need to compare the click coordinates to all four boundaries. To combine the results of two comparisons, we can use a boolean operator “AND” (with the symbol “&&”). The if statement now looks like

```
if (mouseX > 0 && mouseX < gameWidth) {
        tileNumber = tilesPerRow * (mouseY / tileHeight) + (mouseX / tileWidth);
}
```

   Can you add the checks for the y boundaries?

9. You will find a solution at the bottom of the program. There are many ways to do this task, so don’t worry if your solution is different. The main thing is that the solution works!

Recap

We have made our first changes to the program and seen the results. The key language elements we have used are variables and conditional blocks.