24-780 B—ENGINEERING COMPUTATION

Assigned: Wed. Sept. 13, 2023
Due: Tues. Sept. 19, 2023, 11:59pm

Problem Set 3: Circles and Rectangles

In this assignment, you'll create a simple graphics program, so simple that it contains only two kinds of items: circles and rectangles. That's it, nothing else. Well, perhaps there's a little more to it.

Task 1: Develop data structures

Create a struct type called *GraphicalItem* to hold the information for each graphical item. You should include whether the item is a circle or a rectangle (use enum), the center coordinates of the item (int), the height and width of the item (int) (I guess the circle could be an oval 😊), and the color described as three values representing red, green, and blue (RGB as float number from 0.0 to 1.0).

Provide the following function that takes a string and returns a *GraphicalItem* struct with the information in the string:

```
GraphicalItem createItem(const string &input);
```

The string will have a format like this:

```
CIRCLE: 10 20 30 40 0.5 0.6 0.7
RECT: 80 90 100 110 120 0.1 0.2 0.3
```

Where the numbers represent (respectively) the x-coord of center, the y-coord of center, the height, the width, the red color value, the green color value and the blue color value.

Provide a means of storing an indeterminate number of graphical items and develop a console interface for creating graphical items using the *createItem()* function. The user should be able to add a circle or rectangle, clear a single item by providing its index, clear all items, or exit the program. Please be sure that all clearing operations ask for confirmation. [Note: This part got a bit intense, so please use the function provided below as a starting point rather than spending precious time on it.]

You should probably also provide a function that prints out the data stored in a graphical item. The printing should duplicate the format above.

```
void printItem(const GraphicalItem &anItem, ostream &output = std::cout);
```

Note that the declaration above includes a default value for the second parameter which means that if the second parameter is omitted in a function call, the default value will be used (i.e., prints the data to console).

Task 2: Provide Graphical Display

Use FSsimplewindow libraries to provide a graphical display of the items in your model in OpenGL. The window size should be 800x600. Since we tend to think of the origin (0,0) as the lower left corner and then have y-coord go up from there, be sure to allow for this in your graphical output. (e.g., a circle at position 80, 500 should be higher on the screen than a circle at position 80, 300).

You will likely make use of a function like this:

```
void drawItem(const GraphicalItem &anItem);
```

Task 3: Add Permanent Storage

Create a function to read several graphical items from an input file stream (which is created and initialized elsewhere):

```
void readItems(vector<GraphicalItem> &theItems, ifstream &inFile);
```

Provide a function to write all the graphical items in memory to an output stream (which is created and initialized elsewhere):

```
void writeItems(const vector<GraphicalItem> &theItems, ostream &output);
```

Note that the second parameter is of type ostream, which includes cout, thus allowing you to test your function by writing to the console.

Complete the code for user options (L)oad a file and (S)ave to a file. Note that reading from a file does NOT delete the graphical items already existing.

Some sample graphical item collection files are attached to the assignment.

Deliverables

1 file, very appropriately named (or perhaps 3, based on how you feel about header files):

ps03graphicBlocks_yourAndrewID.cpp

or

ps03graphicBlocks_yourAndrewID.h

ps03graphicBlocks_yourAndrewID.cpp

ps03graphicBlocksMain_yourAndrewID.cpp

Upload the file(s) to the class Canvas page before the deadline (Tuesday, Sept. 19, 11:59pm).

Hint: Even if you name your file appropriately, be sure to include your full name within the code itself (perhaps as a comment block at the top of the file). It is also appropriate to include date and course info, plus a short description of what the program does. (Think about what part of the text in the assignment write-up can be copy/pasted or adapted for this purpose.)

Learning Objectives

Thinking about data structures (enum, struct, vector)

Introduction to OpenGL

Making use of console input/output

File input and output

Using functions effectively

Searching references (online and/or textbook) for C++ library functions.

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>

using namespace std;


bool getUserInput(vector<GraphicalItem>& theItems)
{   // guides user through data input and makes changes to
    // the collection of graphical items

    char userChoice;
    int inputInt;
    string longInput, shapeDescription;
    bool validResponse = false, stayInProgram = true;

    while (!validResponse) {
        cout << endl;
        cout << "Make a selection: add (C)ircle, add (R)ectangle, " << endl;
        cout << "                  (D)elete item, clear (A)ll, " << endl;
        cout << "                  (L)oad file, (S)ave file, e(X)it >> ";
        cin >> userChoice;

        cin.clear(); cin.ignore(); // to clear up the input buffer

        if (userChoice == 'C' || userChoice == 'c'
            || userChoice == 'R' || userChoice == 'r') { // add a new shape

            // initialize description string
            if (userChoice == 'C' || userChoice == 'c')
                shapeDescription = "CIRCLE: ";
            else
                shapeDescription = "RECT: ";

            // ask for parameters
            cout << "                 Center of shape (x y) >> ";
            getline(cin, longInput);
            shapeDescription += longInput + " ";

            cout << "          Size of shape (height width) >> ";
            getline(cin, longInput);
            shapeDescription += longInput + " ";

            cout << "      Color of shape (red green blue) >> ";
            getline(cin, longInput);
            shapeDescription += longInput;

            // create and add the item to collection
            theItems.push_back(createItem(shapeDescription));
            validResponse = true;
        }
        else if (userChoice == 'D' || userChoice == 'd') { // delete single item
            if (theItems.size() < 1)
                cout << endl << "                 No items to delete" << endl;
            else {
                cout << "                 Select index of item to delete (1 to "
                    << theItems.size() << ") >> ";
                cin >> inputInt;
                if (0 < inputInt && inputInt <= theItems.size()) {
                    // confirm first
                    cout << "Are you sure you want to delete the item:" << endl;
                    cout << "     ";
                    printItem(theItems.at(inputInt - 1));
                    cout << "   (Y/N) >> ";
                    cin >> userChoice;
                    if (userChoice == 'Y' || userChoice == 'y') {
                        theItems.erase(theItems.begin() + inputInt - 1);
                        validResponse = true;
                    }

                }
            }
        }
        else if (userChoice == 'A' || userChoice == 'a') { // clear all items
            if (theItems.size() < 1)
                cout << endl << "                 No items to delete" << endl;
            else {
                // confirm first
                cout << "Are you sure you want to clear all items? (Y/N) >> ";
                cin >> userChoice;
                if (userChoice == 'Y' || userChoice == 'y') {
                    theItems.clear();
                    validResponse = true;
                }
                else
                    validResponse = false;
            }
        }
        else if (userChoice == 'X' || userChoice == 'x') {
            stayInProgram = false;
            validResponse = true;
        }
        else if (userChoice == 'L' || userChoice == 'l') {
            // ask user for file name
            // create and open an ifstream, checking if it was able to open
            // call the readItems() function

            validResponse = true;
        }
        else if (userChoice == 'S' || userChoice == 's') {
            // ask user for file name
            // create and open an ofstream, checking if it was able to open
            // call the writeItems() function

            validResponse = true;
        }

        if (!validResponse)
            cout << endl << "No changes were recorded " << endl;
    }

    // return true if program should continue
    return stayInProgram;
}
```