CS-224 Object Oriented Programming and Design Methodologies

Homework 03

Fall 2022

1 Submission Policy

You need to submit this homework on 07th October at 11:59pm, on LMS. Late submissions are allowed until 09th October 11:59pm, which will be penalized by 20%. Your work will not be accepted once the submission is closed on LMS.

2 Guidelines

- You need to do this assignment in a group of two students.
- You will submit your assignment to LMS (only one member of the group will submit).
- Clearly mention the group composition in submitted file name e.g. AhmadHassan_ah01345_BatoolAiman_ba03451.zip.
- You need to follow the best programming practices
- Submit assignment on time; late submissions will not be accepted.
- Some assignments will require you to submit multiple files. Always Zip and send them.
- It is better to submit incomplete assignment than none at all.
- It is better to submit the work that you have done yourself than what you have plagiarized.

- It is strongly advised that you start working on the assignment the day you get it. Assignments WILL take time.
- DO NOT send your assignment to your instructor, if you do, your assignment will get ZERO for not following clear instructions.
- You can be called in for Viva for any assignment that you submit

3 Package Delivery System

For this assignment you will be implementing a binary search tree(BST) composed of Truck variable. Truck structure is defined as:

```
struct Truck
{
    string driver;
    double petrol;
    string regNo;
    int fullMileage;
    int emptyMileage;
};
```

You will be using the sample file, `Input.txt` for this assignment. Your code should however take into account that if an entry is increased or reduced (5 lines per entry) it reads all the entries in the file (You are going to assume that there are no errors in the file). For example, if there is just one entry:

Elton John
34
AB218
9
7

Based on this entry, the driver's name is Elton John, the truck has 34 liters of petrol in its tank, its registration number is AB218. It covers 9 km per liter if empty and 7 km per liter when loaded.

The function `loadTrucks()` reads the file `Input.txt`, and populates a BST of trucks according to information given in the file. You can compare a truck variable is less than other truck variable on the basis of its registration number, hence based on this comparison you can populate the binary search

tree. As you read information of a truck, a new BSTNode is created and inserted in proper location in the BST.
The function `makeJourney()` traverses all the trucks and updates their remaining fuels after a truck takes cargo and travel 60 km, drop the cargo and return empty based on the fuel consumptions. For example, the truck information given above can make this journey, as it has 34 litres petrol in its tank and it requires $60/9 + 60/7 = 15.23$ litre of petrol. Hence the remaining fuel in this truck would be 18.77 litre. Those trucks who are unable to make the journey will will not travel, and hence their fuel information will not be updated.

Finally, the function `unloadTrucks()` shows all the trucks information in ascending order of registration number. A new file `Trip.txt` should be generated that will show the current state of all the Trucks. The file format is similar as Input.txt.

Note: this question does not require SDL files.

4 HUMania

A sample code is given in HUMania folder, if you run it you can see a pigeon is moving slightly towards right side. This example creates just one variable of Unit type, and draws pigeon taken from assets file.

You are required to create 3 different types of objects drawn on the screen: Pigeon, Butterfly and Bee. As you click on the screen the function createObject is called with mouse coordinates given in x, y variables. You have to create a pigeon, butterfly or a bee randomly on the screen. You'll be maintaining 3 different std::vectors (refer to section 5), one for each of the type of item drawn. Please refer to section 4.1, you only have to change srcRect to draw different type of items.

Animations: As you draw the objects on screen, you also have to animate them. For this purpose, you cycle through the 3 different states of the object as given in assets file. All of the objects should keep moving slightly towards right, and when an object is reached to right most corner of the screen, it should reappear from the left most corner of the screen.

4.1 SDL Drawing Basics

The basic drawing function in SDL is very simple, you need two SDL_Rect variables to draw a portion of image from assets file to the canvas. `SDL_Rect`

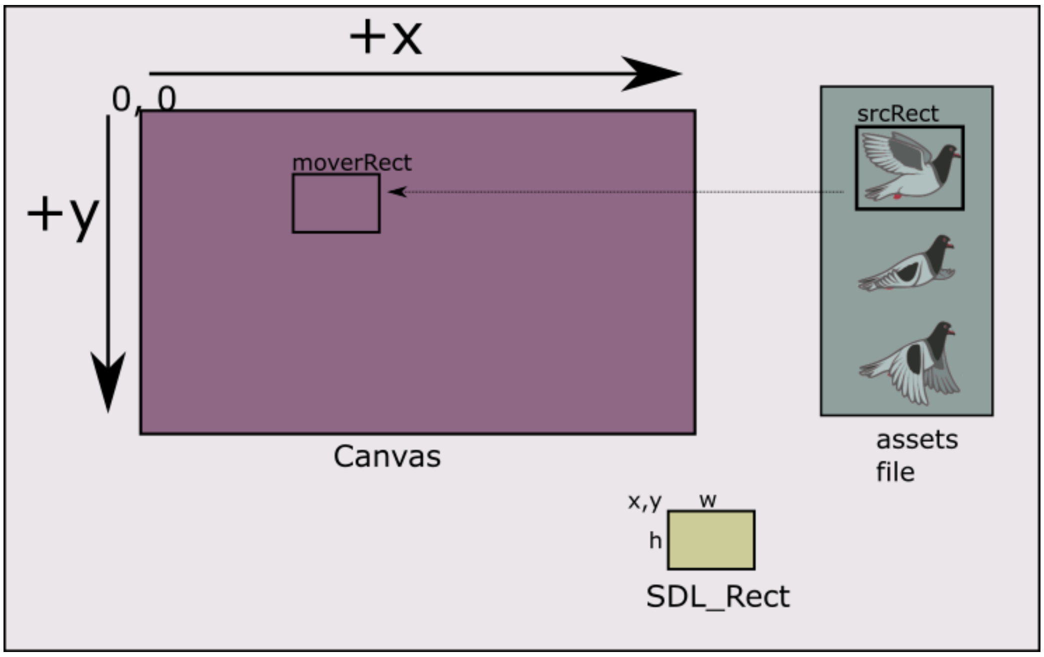

Figure 1: SDL Drawing Basics

is a simple structure containing {`x, y, w, h`} attributes. `(x, y)` is the top-left corner, and `w, h` are width and height of rectangle. You define a `srcRect` for desired object in assets file, and define a `moverRect` for this image to be drawn on desired location on canvas. Refer to Figure 1 for all this process. Finally you call
`SDL_RenderCopy(gRenderer, assets, &pigeonSrc, &pigeonMover);`
that displays this image to the canvas, voila!!!. Refer to `assets.png` file for all the required image assets.

You can draw as many objects in the `HUMania.cpp` ⇒ `drawObjects()`, as you want. Since this function is called infinitely, you can change the `x, y` attributes of `moverRect` to move the objects on screen, and you can change the `srcRect` values to get a flying animation.

5 `std::vector` Tutorial

Following is a basic example to work with vector. Complete reference for C++ vector is given here `https://en.cppreference.com/w/cpp/container/vector`

```
#include<iostream>
```

```
#include<vector>

using namespace std;

struct Distance{
   int feet, inches;
};

int main(){
   vector<Distance> dst; // It’s a vector that can store Distance
       type objects
   dst.push_back(Distance{3, 4}); // create an object, and push it
       in vector
   dst.push_back(Distance{5, 2});
   dst.push_back(Distance{2, 7});
   dst.push_back(Distance{7, 8});
   dst.push_back(Distance{13, 1});

    Distance d1 = {5, 12};
    dst.push_back(d1);

   for(int i=0;i<dst.size();i++)
      cout<<dst[i].feet<<"’"<<dst[i].inches<<’"’<<endl; // call
         show method of dst[i] object
}

//////////////// Output: ///////////////////
3’4"
5’2"
2’7"
7’8"
13’1"
5’12"
```

6 Some important points:

- Sample code is there for your benefit. If you are going to use it, understand how it works.
- You do not need to follow the code given exactly. You can make changes where you see fit provided that it makes sense.

- Complete reference for C++ vector is given here `https://en.cppreference.com/w/cpp/container/vector`
- You need to define separate `*.hpp` and `*.cpp` files for all the classes.
- Exact x,y,w,h values for images in assets file can be found by `http://www.spritecow.com/`.
- A tutorial for file I/O is given `http://www.cplusplus.com/doc/tutorial/files/`.
- You should take `www.cplusplus.com` and `www.cppreference.com` as primary web source to search about C++
- You have to follow best OOP practices as discussed in lectures.

7 Rubric

Warnings/Errors	The code had no warnings/errors	1
Comments	The code was properly commented	2
Coding	The code followed best practices guideline	2
OOP Concepts	The code followed best OOP practices	5
Functionality	All the functionality is implemented as described above	10
Total		20

Table 1: Grading Rubric

8 Credits

Some questions in this assignment are derived from the work of Dr. Umair Azfar Khan.