

CS-224 Object Oriented Programming and Design Methodologies

Assignment 03

Fall 2021

1 Guidelines

You need to submit this assignment on **Oct. 24 at 11:59pm** . Late submissions are not allowed for this homework. Some important guidelines about the assignment are as following:

- 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

2 Package Delivery System

For this assignment you will be creating a package delivery system. You need to think in terms of objects. The first object is the delivery truck (**Truck**) that can store 50 liters of petrol. The cost per liter of petrol is 2.73\$. You will be using the sample file, **Drivers.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  
218  
9  
7
```

Based on this entry, the driver's name is Elton John, his truck has 34 liters already, his total funds are 218\$. His truck covers 9 km per liter if empty and 7 km per liter when loaded.

The trucks can carry 10 packages/boxes (which is the second object) with random dimensions. The length, width and height of every package can range from 5 to 30 inches. You can set each box dimensions randomly for every truck.

The function **loadTrucks()** reads the file **Drivers.txt**, and populates a vector of trucks according to information given in the file. Each Trucks' **load()** is called in this function.

The function **calculateCost()** calculates the total cost it will take the loaded truck to travel 60 km, drop the cargo and return empty based on the fuel consumption when the tank was full. The drivers need to fill the tank first before making the journey, the tank can be filled only once. Based on the amount of money they have, calculate if everyone can do the journey. Those trucks who are unable to make the journey will be removed from the vector.

The function **makeJourney()** updates all the trucks (which made journey) after making all the journey, i.e. it'll update with the remaining fuel and funds.

The function `unloadTrucks()` shows all the trucks information, once the journey is complete. It calls the `unload()` of every truck, which prints the volume of all boxes it carried. A new file `Trip.txt` should be generated that will show the current state of all the Trucks that made the journey. The file format is similar as `Drivers.txt`.

The truck needs to have a `load()` and an `unload()` Function. When the trucks have been generated, the `load()` function should be called that will generate the boxes and put them inside the truck (It should show the dimensions of all the boxes). Once the journey is over, it should call the `unload()` function and unload all the boxes (It should show the dimensions of all the boxes).

Note: this question does not require SDL files.

3 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 object of Pigeon to show how things are drawn in SDL. Refer to `Pigeon.hpp/cpp` and `HUMania.cpp` \Rightarrow `drawObjects()`.

You are required to:

- Create a `Pigeon` class (see the `pigeon.hpp/cpp`), that will contain attributes and functions (`fly`, `draw`) related to a pigeon. The `fly` function flies the pigeon gradually to top-right side, and gets back the pigeon to left most corner as they approach at the right most border of the window. Three different images in assets file will be changed back and forth to make the pigeon fly. `draw` is only drawing the Pigeon object.
- Create an `Egg` class (create `Egg.hpp` and `Egg.cpp`), that will contain attributes and functions (`drop`, `draw`) related to egg. `drop` function makes the egg drop on the floor. Its shape changes to broken egg as it reaches to bottom of screen, and it doesn't move further down. `draw` function is only drawing the Egg object.
- Create a `Nest` class (create `Nest.hpp` and `Nest.cpp`), that will contain attributes and functions (`wiggle`, `draw`) related to Nest. `wiggle` is making a nest wiggle on the screen (without dropping down). Look at the three images in assets file to make a nest wiggle. `draw` function is only drawing the Nest object.

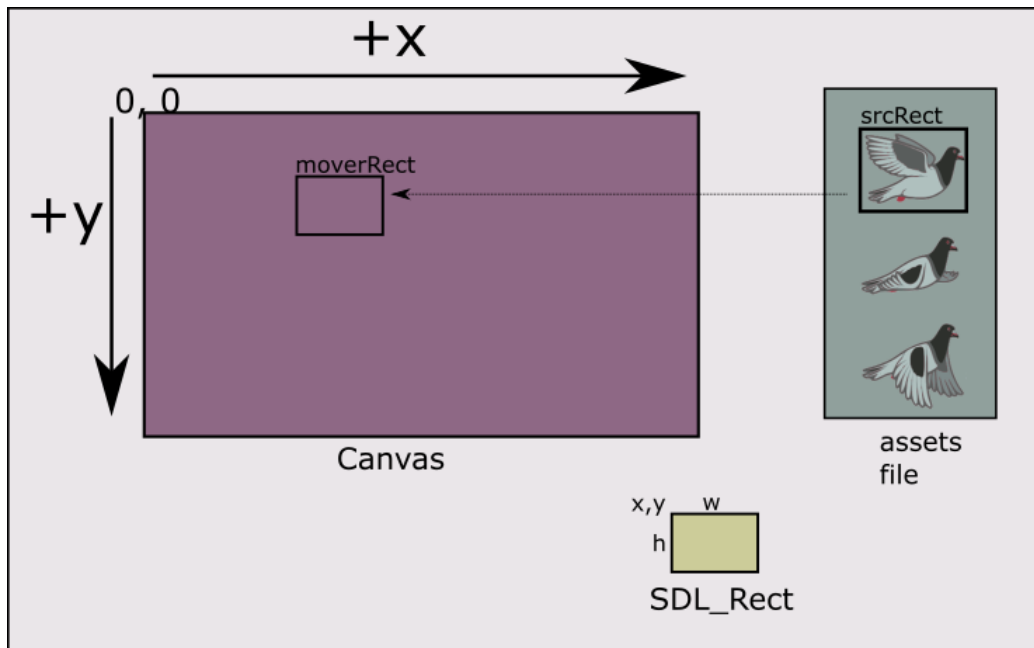


Figure 1: SDL Drawing Basics

- As you click on the screen, one of the above objects is created randomly. You'll maintain three vectors (pigeons, eggs, nests) in `HUMania.hpp/cpp` to store objects of different classes. The object that you create on the click will be pushed into corresponding vector. Refer to `HUMania.cpp` \Rightarrow `creatObject()`, where you get mouse coordinates.
- Finally, you iterate over all the elements of vectors, and call their corresponding functions to make them animate(fly, drop or wiggle) and draw.
- Are you having fun?? You are more than welcome to add more stuff to make this game interesting, e.g. pigeons swoop at random intervals and then fly upwards, some eggs don't break on the floor rather hatch to a new pigeon etc. [*It doesn't carry any marks*]

3.1 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` 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` \Rightarrow `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.

4 `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;

class Distance{
    int feet, inches;
public:
    Distance(int ft, int inch): feet(ft), inches(inch){}
    void show(){
        cout<<feet<<" "<<inches<<"\"<<endl;
    }
};

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));

    for(int i=0;i<dst.size();i++)
```

```

        dst[i].show(); // call show method of dst[i] object
    }

    //////////////// Output: ////////////////
    3'4"
    5'2"
    2'7"
    7'8"
    13'1"

```

5 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.
- Make the class declarations in `hpp` files, and provide function implementations in `cpp` files. Don't use `hpp` files for implementation purposes.
- As a general rule, class's data is private, and functions are public. Don't use getter/setter functions to manipulate data, rather think in object oriented directions and provide all the functionality in the class.
- 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.

6 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

7 Credits

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