ECS40 Winter 2017 **2017-02-22**

Homework 3 due Fri 2017-03-03 at 20:00

Use the handin directory hw3 to submit your work

Filling a rectangular domain with random shapes

Description

This assignment builds on the **`Shape`** class designed in HW2. In this assignment (HW3), you will modify the implementation of the **`Shape`** class that was developed in HW2, and implement a class **`Domain`** representing a rectangular area in the x-y plane, with functions that draw a visual representation of the domain as a scalable vector graphics (svg) file. The svg file can be visualized using a browser such as Firefox, Safari, Chrome or Internet Explorer.

The domain is a rectangular array of cells that can be occupied by shapes. The goal of the assignment is to implement a program that places randomly selected shapes at random positions on an arbitrary rectangular domain until the domain is full, and creates a graphical representation of the domain.

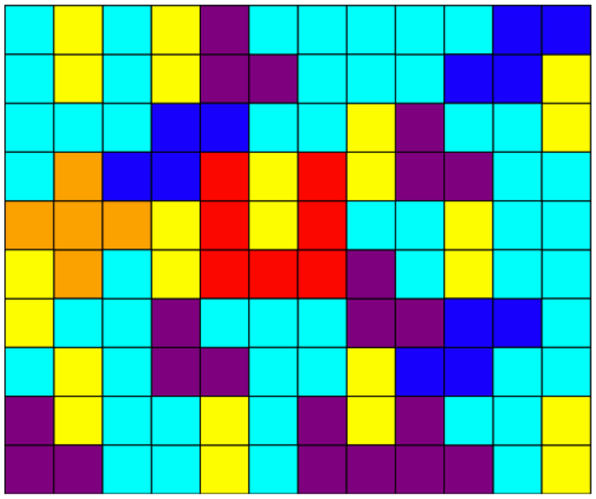

Figure 1: Rendering of an svg file obtained with a 10x12 filled domain

HW3 Assignment

In HW3, you are given the files **`fill.cpp`** (which contains the **`main`** function) and the files **`Shape.h`** and **`Domain.h`**. You should not modify these files. Your task is to implement the class **`Domain`** by writing the file **`Domain.cpp`** and by writing a new file **`Shape.cpp`** that implements the additional specifications given in HW3. Note that the file **`Shape.h`** is different from the one given in HW2.

All source files should compile without warning. You should provide a **`Makefile`** in order to be able to build the executable **`fill`** using the command

```
$ make
```

Input

Input for the **`fill`** program is provided on the command line. It does not read input from any file.
The command line arguments of the **`fill`** program are: **`size_x size_y seed`**
`size_x`: size of the domain (number of cells) in the x direction
`size_y`: size of the domain (number of cells) in the y direction
`seed`: an integer used as a seed for the random number generator **`rand()`** used in **`fill.cpp`**
The program is run, for example, using the following command line:

```
$ ./fill 12 10 1234 > output.svg
```

The **`fill`** program will be tested with various sizes in the range [1,15], and various seed values. It is not necessary to test for incorrect values in the command line arguments.
You can use the **`fill`** executable with specific command line arguments, and compare your svg output with svg output test files provided on the web site to check the functionality of your classes. **Note:** the random numbers generated on different computers **may differ even if the same seed is used**. Make sure to generate and compare your output on the CSIF computers. Make sure that your program reproduces the test output *exactly*. Use the **`diff`** command to compare your files with the example files. Other command line arguments will also be used when grading your implementation. The example files are generated using the following naming convention. For example:

```
$ ./fill 12 10 2 > test_12_10_2.svg
```

Modifications of the Shape class

The **`Shape`** class should have the following features in addition to the ones defined in HW2 (see the file **`Shape.h`**)

Additional public members of the Shape class

`virtual const char* color(void) const = 0`
A pure virtual member function returning a C-style string (such as e.g. **`"red"`** or **`"blue"`**) that defines the color of the shape. The strings returned by the function should be as follows:

O	cyan
I	yellow
L	purple
S	blue
X	orange
U	red

`int getX(int i) const`
Returns the x coordinate of cell **`i`** of the shape. It is assumed that the value of the argument is valid.

`int getY(int i) const`
Returns the y coordinate of cell **`i`** of the shape. It is assumed that the value of the argument is valid.

`void draw(void) const`
A function that prints on **`stdout`** a sequence of statements that represent the shape in the scalable vector graphics (svg) format. The output should consist of multiple lines, each line representing one of the cells of the shape. Each cell should be drawn as a square of side 40 pixels, filled with the shape's color. For example the **`draw()`** function called by a 'S' shape located at position (4,2) should output the following four lines:

```
<rect fill="blue" stroke="black" x="160" y="80" width="40" height="40"/>
<rect fill="blue" stroke="black" x="200" y="80" width="40" height="40"/>
<rect fill="blue" stroke="black" x="200" y="120" width="40" height="40"/>
<rect fill="blue" stroke="black" x="240" y="120" width="40" height="40"/>
```

Specification of the Domain class

The **`Domain`** class is responsible for managing the shapes added to the domain. It has a member function **`addShape`** which adds a shape to the domain after checking that the new shape fits inside the domain area and does not overlap with any of the previously added shapes.

public members of the Domain class

`Domain(int size_x, int size_y)`
Constructor taking as arguments the horizontal and vertical size of the domain, expressed as the number of cells in each direction.

`void addShape(char type, int x, int y)`
Creates a shape of appropriate type according to the **`type`** argument, and adds it to the domain. The function should first test if the new shape fits within the domain, i.e. that all cells of the shape are within the domain. If it doesn't, the function should simply return and do nothing else. Care must be taken to free any resources allocated until then.
The function should then test if the shape overlaps with any shape already present in the domain. If there is overlap, the function should return and do nothing else. Care must be taken to free any resources allocated until then.

`bool fits(const Shape &s) const`
A function that returns **`true`** if the shape **`s`** fits inside the the boundaries of the domain. (i.e. the coordinates of all its cells satisfy x >= 0, x<size_x, y>=0, y<size_y) and **`false`** otherwise.

`bool full(void) const`
A function that returns **`true`** if the domain is full, (i.e. the sum of the sizes of all shapes present in the domain is equal to the total number of cells), and **`false`** otherwise.

`void draw(void) const`

This function draws the current state of the domain by printing a complete svg document on **`stdout`**. It should first print the following svg header:

```
<?xml version="1.0" encoding="utf-8"  standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="670" height="670" viewBox="0 0 650 650"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" >
<g transform="matrix(1,0,0,-1,50,650)">
```

Note that this header is fixed, i.e. it is independent of the domain size.

It should then draw a frame with white background defining the domain area. For example, for a domain of 8x5 cells, the frame is drawn by printing

```
<rect fill="white" stroke="black" x="0" y="0"  width="320" height="200"/>
```

Note that the width and height attributes depend on the size of the domain.

It should then draw each shape on the domain by calling the shapes' **`draw`** functions.
Finally it should print the following svg trailer:

```
</g>
</svg>
```

The resulting svg file can be visualized by opening it with a browser such as Firefox, Safari, Chrome or Internet Explorer.

private members of the Domain class

`const int size_x, size_y`

The dimensions of the domain measured in number of cells in each direction.

`vector<Shape*> sList`

A vector of base pointers to store pointers to the shapes added to the domain.

Submission

Create a tar file named **`hw3.tar`** containing the files **`fill.cpp Shape.h Shape.cpp Domain.h Domain.cpp`** and **`Makefile`**. Do not compress the tar file. The Makefile should include the necessary definitions to compile C++ files with the **`-Wall`** option. Include your name and Student ID in a comment at the top of each file that you wrote (but not in the files provided). Submit your project using: **`$ handin cs40 hw3 hw3.tar`**