Assignment 02

CSE24101

May 19, 2023

1 Description

You are to make a simple TUI (Textual User Interface). Specifically, you are to write a number of TUI elements, described below, which must render graphically to ASCII text. To do this, you will use polymorphism, having all of your elements descend from a common `widget` abstract class.

Your `widget` class needs a (virtual) destructor (as some of your elements will need to manage memory). In addition, it will need the following method:

```
char char_at(unsigned int x, unsigned int y, unsigned int width, unsigned int height) const;
```

The method will return the character to display, at the given x, y coordinates. The *width* and *height* parameters tell you the dimensions of the element being rendered.

In addition to writing the code for all of the widget classes, you must also write `main10.cc`, to produce the correct output for test case 10. It should produce a 60x30 rendering.

1.1 Elements

`label` —renders a simple string, all on one line. For any y coordinate other than 0, or for any x coordinate beyond the length of the string, simply return a space character. Otherwise, return the appropriate character from the string. If the string is too long for the given width, then the last three characters returned should be `...`. See test cases 00, 01, 02, 05, 06, 08, 10.

`checkerboard` —renders two characters, over and over again, in a checkerboard pattern. For example, if the two characters are `A` and `B`, then even-numbered rows would be rendered `ABABABAB...` and so on, and odd-numbered rows would be rendered `BABABABA...`. See test cases 03, 05, 09, 10.

`stretchy` —renders a pattern that looks like something being stretched vertically. All characters on the top-most and bottom-most rows must be `.` (period). All characters on the 2nd-top-most and 2nd-bottom-most rows must be `:` (full colon). All characters in the middle rows (if there are any) must be | (vertical bar). It's called a "stretchy" because it stretches vertically to fit any height. See test cases 04, 06, 07, 09, 10.

`vertical_split` —composed of two pointers to `widget` objects, a "top" and a "bottom". If the height of the vertical split as a whole is h, then the height of the top element is $\lfloor \frac{h}{2} \rfloor$ and the height of the bottom element is $\lceil \frac{h}{2} \rceil$. See test cases 05, 06, 08, 10.

`window` —displays a frame around another `widget` object. The frame is composed of `-` (hyphen), | (vertical bar) and `+` (plus) symbols. Note that if the dimensions of the `window` are w and h, then the dimensions of the inner `widget` are $w-2$ and $h-2$. See test cases 07, 08, 09, 10.

`overlapping` —composed of two pointers to `widget` objects, the "back" and the "front". If the dimensions of the `overlapping` are w and h, then the dimensions of the back and front are $\frac{2}{3}w$ and $\frac{2}{3}h$. You are free to round up or round down, as you find most convenient. The "back" `widget` is in the upper-left and is partially obscured by the "front" `widget`, which is in the bottom-right. See test cases 09, 10.

2 Project structure

Submit your project as a .zip file, consisting of:

Makefile — must build all test cases (which you have finished). You may use the one on Blackboard if you wish, and may modify it if you wish. Running `make` *must* build all test cases.

widget.h — must include a definition of the `widget` class and all subclasses

widget.cc — must include the method bodies of any methods for the classes in `widget.h`. If you want to break this up into multiple `.cc` files, that is fine. Just make sure your `Makefile` includes all the appropriate `.o` files

main10.cc — must produce the output given for test case 10, as indicated below

3 Test case correct output

Here's the output for `main00-bin`:

```
text
```

Here's the output for `main01-bin`:

```
hello world!


```

Here's the output for `main02-bin`:

```
hello w...

```

Here's the output for `main03-bin`:

```
 / / / / / / / / / /
/ / / / / / / / / /
 / / / / / / / / / /
/ / / / / / / / / /
 / / / / / / / / / /
/ / / / / / / / / /
 / / / / / / / / / /
/ / / / / / / / / /
 / / / / / / / / / /
/ / / / / / / / / /
```

Here's the output for `main04-bin`:

```
....................
::::::::::::::::::::
||||||||||||||||||||
||||||||||||||||||||
||||||||||||||||||||
||||||||||||||||||||
||||||||||||||||||||
||||||||||||||||||||
::::::::::::::::::::
....................
```

Here's the output for `main05-bin`:

```
This is some very fi...




.X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.X
.X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.X
.X.X.X.X.X.X.X.X.X.X.
```

Here's the output for `main06-bin`:

```
........................
::::::::::::::::::::::::
||||||||||||||||||||||||
::::::::::::::::::::::::
........................
abcdef

........................
::::::::::::::::::::::::
........................
```

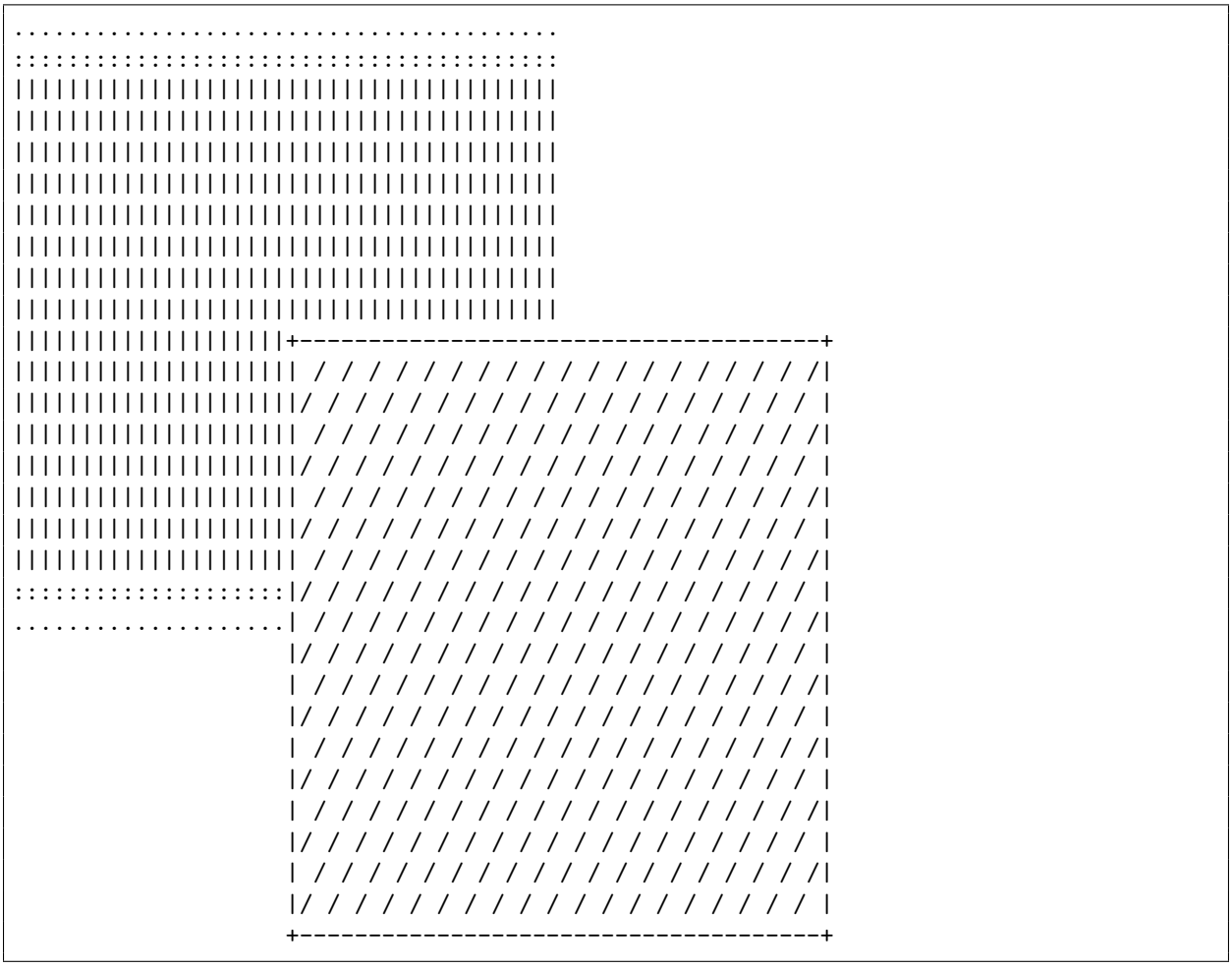
Here's the output for `main07-bin`:

```
+----------------------+
|......................|
|::::::::::::::::::::::|
||||||||||||||||||||||||
||||||||||||||||||||||||
||||||||||||||||||||||||
||||||||||||||||||||||||
|::::::::::::::::::::::|
|......................|
+----------------------+
```

Here's the output for `main08-bin`:

```
+------------------+
|This is some text |
|                  |
|                  |
|                  |
|and some more text|
|                  |
|                  |
|                  |
+------------------+
```

Here's the output for `main09-bin`:



Here's the output for `main10-bin`: