PS0: Hello World with SFML

The main purpose of this assignment is for you to get up and running with your build environment. You'll write a bit of code.

1. Get your build environment set up (skip this step if you have installed the VM already!)

1) Download and install the VirtualBox platform package for your operating system. The current version is 6.0.2:

   https://www.virtualbox.org/wiki/Downloads

   (*Note*: If you have a 32-bit host computer, you must use version 5.2.x: https://www.virtualbox.org/wiki/Download_Old_Builds_5_2)

   When you run VirtualBox for the first time, it will prompt you to install the VirtualBox Extensions Pack.

2) Download the instructor's Ubuntu virtual disk image (VDI) for VirtualBox:

   https://studentuml-my.sharepoint.com/:u:/g/personal/yelena_rykalova_uml_edu/Ee3djnigomdLgAZ2FJ5shMUBiJcZnxvN43dhH0ohtxEuAQ?e=PSeuRB

   This system image has been prepared with most of the tools that you will need for this course, as well as a course source folder called ~/COMP2040 to help get you started.

   (*Note*: The Zipped disk image file size is about 19 GB. If you have trouble downloading this file, please see the alternate download instructions below.)

   Move this VDI file to an appropriate folder (e.g., "Documents/Virtual Machines").

   Alternate download instructions for Ubuntu VDI file:

   The link below will take you to a directory containing the Ubuntu VDI, compressed by 7-zip and split into chunks:
   https://studentuml-my.sharepoint.com/:f:/r/personal/yelena_rykalova_uml_edu/Documents/Ubuntu_split?csf=1&web=1&e=kXHhxi
   Download each chunk individually into a folder, then use 7-zip to re-combine the chunks.

3) In VirtualBox, create a new Ubuntu guest virtual machine by clicking the "New" button. At the "Hard disk" step, choose "Use an existing virtual hard disk file" and navigate to the Ubuntu VDI that you downloaded in Step 2.

   Note: VirtualBox should default to creation of a 64-bit guest virtual machine. If you are given only the option for creating a 32-bit guest machine, you probably need to change one or more settings in your host computer's BIOS. See the following article for helpful hints:
   https://superuser.com/questions/1241956/virtualbox-only-allowing-32-bit-os

4) Click the "Start" button to "power-on" your shiny new Ubuntu VM (the default username and password are both "osboxes.org"). When you run the VM for the first time, it will prompt you to upgrade the Guest Additions for Linux, which will allow you (among other things) to designate a shared folder on your host computer that can be accessed both by the host and guest machines.

We are using the C++ Simple Fast Media Library, SFML, www.sfml-dev.org, version 2.5.1

Ubuntu VM is required on both platforms: Linux and Windows. If you cannot install VB on your machine for any reasons – you can use our CS Labs' machines (on and off campus). (see https://www.uml.edu/Sciences/computer-science/CS-Infrastructure/Laboratory-Information.aspx for information about Labs)

For access use MobaXterm (free app) or Xquartz instead of putty/x2go. All settings and configurations are all set for you to do their assignments and projects. This is terminal that support both text and gui display. It also has function like winscp within it and has more functions to help end user easier to access remote unix mac.

For Xquartz in Mac, you only need to download and install it. They use Mac terminal and run

ssh -X student_CS_username@cs.uml.edu

at the prompt.

Please send email to help@cs.uml.edu if you have any issues.

2. Build the SFML Hello World code

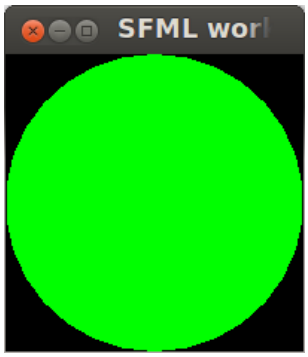
The SFML "did I install everything correctly" code is about 20 lines long and results in your computer popping up a window like this:

Go to the tutorial for your platform, e.g. https://www.sfml-dev.org/tutorials/2.5/start-linux.php , and get the code running on your machine.

3. Extend the demo code.

Now that you've got things running, do something fun with SFML. In addition to the green circle, look at the documentation and get your window to show:

1. drawing an image sprite
2. make the image sprite move
3. make the image sprite respond to keystrokes
4. make it do something else



Use the documentation at https://www.sfml-dev.org/documentation/2.5.1/ .

You probably want to make your window bigger than the 200x200 specified in the Hello World code. Also, no need for audio now (we will be using it later).

For #2 and #3 above, you can make x, y variables that represent the position of the sprite, and update them when a keystroke is detected. You're welcome to do something more complicated if you like.

The **window.setFramerateLimit()** method might be helpful if things move too fast (see "Controlling the framerate" https://www.sfml-dev.org/tutorials/2.5/window-window.php).

Notes:

- It's correct to clear everything in the window and redraw stuff each time through the event loop. That's how SFML works.
- You reposition things (e.g. a sprite) by setting its position before you draw it. Check the sprite API docs.
- Use only relative paths to load your sprite image — e.g. "**./sprite.png**", *not* "**/home/…/ps0/sprite.png**".
- If you draw stuff outside the bounds of the window, you won't see it.

Compiling program:

- When you compile your code, add the flags "**-Wall –Werror -pedantic**". We will be using these to test your code, and your code must compile when these flags are used.

Make sure your code file is named **main.cpp**. You will submit this. Make sure your name is at the top of the file.

Make sure your sprite image file is named sprite.png. You will submit this.

When you are done, take a screenshot of just your SFML window, and save this image in a file named screenshot.png. You will submit this.

For making the screenshot on Ubuntu you can use directions provided here:

https://help.ubuntu.com/stable/ubuntu-help/screen-shot-record.html

If you are new to Ubuntu, I highly recommend Shutter for doing screen grabs of windows or a selected area of the screen. Install with:

```
sudo add-apt-repository ppa:shutter/ppa
sudo apt-get update
sudo apt-get install shutter
```

The Print Screen key also works, but you'll have to crop the grab afterward to just show your SFML window.

4. Assignment readme File

Before submitting, fill out info in the ps0-readme.txt file. Make sure you re-save it as a text only file. It must be named ps0-readme.txt.

5. Submit!

You will be submitting at least four files.

1. Your SFML demo program, named main.cpp.

2. Your sprite image file, named sprite.png. (If you made more than one sprite, you can name the others whatever you like.)
3. Your screen grab showing your code running, named screenshot.png.
4. Your completed ps0-readme.txt file.

The four files must be in a subdirectory named ps0. Then, cd to the directory containing the three files, and type the following:

```
$ ls
```

You should see the files:

```
main.cpp  ps0-readme.txt  screenshot.png sprite.png
```

Then, type the following commands to make a gzipped tar archive of the directory:

```
$ cd ..
$ tar czvf my-submit.tar.gz ps0
```

You should see the following displayed:

```
ps0/
ps0/sprite.png
ps0/screenshot.png
ps0/ps0-readme.txt
ps0/main.cpp
```

Finally, submit your file on Blackboard. (Note: The actual name of your submit file should include your real name and the project name "PS0" – e.g., **PS0_JoeSmith.tar.gz** . But it must be a gzipped tar of the correct directory.)

Grading rubric

Feature	Value	Comment
core implementation	**6**	**full & correct implementation**
		1 pt file name correct 1 pt for SFML window being displayed 1 pt for image 1 pt for image being in moving sprite 1 pt for sprite changing direction or somehow responding to keystrokes 1 pt for additional feature (explain in README)
screenshot	**2**	**screenshot of just SFML window included**
		-1 pt if it's a screen-grab of your entire monitor

tar.gz archive	**2**	**all files packaged in .tar.gz file with correct directory structure**
readme.txt	**2**	**complete**
Total	**12**	