Programming Project #1

Writing Classes for SDL Animation

CpSc 4160/6160: Data-Driven 2D Game Development

Computer Science Division, Clemson University

Brian Malloy, January 25, 2016

Due Date and Submission:

To receive credit for this assignment, your solution must meet the requirements specified in this document, and be submitted, using the department `handin` facility, by 8 AM on Wednesday, February 10*th*, 2016. If you cannot complete the project by the due date, you can receive 90% of the grade if you submit the assignment within one week after the due date.

Your project should be a compressed directory containing your program files, assets, README, and a Makefile. Be sure to 'make clean' before compressing your directory. Your program files must be written in C^{++}, use the Simple DirectMedia Layer (SDL), and contain no memory leaks in code written by you.

Projects compressed using rar will receive a zero and will not be graded. To compress the directory that contains your project, assume that all items are in a directory labeled 'asg1'. Then you can compress the directory in 1 of 2 ways:

```
tar zcvf asg1.tar.gz asg1
zip -r asg1.zip asg1
```

Project Specification:

Your assignment is to build a project that implements a simple SDL animation. To illustrate the idea, several animations will be shown during lecture, and four directories illustrating the progression toward animation will be included in the course repo. Your project should include at least two well-written C^{++} classes designed and implemented by you. These classes should illustrate the concepts described in lectures, the course videos, the slides, and the items in the Meyer's text (use the g++ compiler and supply the Meyer's flag in your Makefile).

To design your classes, you might consider writing a manager class that manages the assets and the sprites. You might also consider writing a sprite class that loads and manages the image(s) and moves the image(s) in the animation. You might consider including methods in the sprite class for drawing and updating the image(s). However, the overall design is completely your choice, with the exception that your classes must be in C^{++}, not C, and conform to the rules and policies outlined above.

Finally, your animation should look the same on all platforms. Also, you should arrange your animation so that it captures 60 frames per second and updates the animation, using the game clock, not the cpu clock, to drive your animation. An example, titled smoothAnimation, is provided in the repo; you should use this code as a model and, possibly, your starting point. In other words, you can use this code in your solution, except that you should have 2 or more classes, not counting GenerateFrames (explained below). There is no upper bound on the number of classes but please keep the scope of the project reasonable.

Generating Frames:

A C^{++} class, GenerateFrames, will be provided for you that will facilitate generation of frames so that we can make a video illustrating your animation for all members of the class to view and appreciate. However, **do not** submit frames. There is a boolean variable, makeVideo, that will control whether or not the frames are captured. We will set this variable to true and that should enable us to generate frames to capture your complete animation.

Each frame should be named as follows:

```
<username>.0000.bmp
...
<username>.0199.bmp
```

For example, a frame might be named "malloy.0001.bmp". However, your project should generate frames with your username, which will require you changing a class constant in GenerateFrames.