Programming Project #1

Writing Classes for SDL Animation

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

Computer Science Division, Clemson University

Brian Malloy, January 28, 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 an example, smoothCapture, will be included in the course repo. Your project should use smoothCapture as a basis, and much of the code in smoothCapture should be incorporated into two or more 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).

One approach that you might adopt is to 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. But you are free to design the project any way that you like. Your design should likely start by considering the data, placing the data in the private section of an appropriate class, and then providing methods to access or manipulate the data.

Your animation should use single frames, rather than multiple-frames, and should run the same on all platforms. Your animation should capture 60 frames per second and should update, using the game clock, not the cpu clock. Study the example in the repo, smoothCapture, and use this code as a model and starting point. We have provided a class, GenerateFrames, which you should use to generate frames for your animation (explained below).

Generating Frames:

A C^{++} class, GenerateFrames, is included in the smoothCapture example, and 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 controls whether or not the frames are captured. You should set makeVideo to false during development and testing of your project. We will set this variable to true when we grade your project and when we generate frames to capture your animation. Arrange your project so that when makeVideo is true, your complete animation is captured.

Your frames must 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.