Programming Project #3

Polymorphism in Action

The Tracker Framework

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

Computer Science Division, Clemson University

Brian Malloy, March 1, 2017

Due Date and Submission:

You may receive a grade of 90% if your solution meets the requirements specified in this document, in the course policy, and is submitted by 8 AM on Thursday, March 10*th*, 2017. If you do not complete the project by the due date, you can receive 80% of the grade if you submit the assignment within three days after the due date. The final 10% is explained below.

Project Specification:

To facilitate the development of a video game in one semester, I have designed and implemented a bare bones video game framework, tracker framework, that you should use as a starting point for this project. Study the data-driven *tracker framework* code, which is free of memory leaks, and includes an XML parser, singleton classes, a factory class, flyweight, and an engine class that contains a polymorphic vector of drawables. You may design your own framework, or you may use part of the tracker framework, but your resulting submission must include the positive aspects of the tracker framework.

To succeed in this project, you should complete the following tasks:

1. Print your name and the title of your animation at the top/middle of the window.

2. Class RenderContext has two global constants; get these constants from XML (or JSON).

3. Incorporate class FrameGenerator into the Tracker framework so that F4 toggles frame generation. I've done this for you in 4160assets-2017/projects/3/tracker.withFG

4. Incorporate parallax scrolling into the animation by using at least two backgrounds. Please note that readTexture and readSurface routines in IOmod require that your image has an *alpha channel*. If your sprite sheet doesn't have one, you must add it. I recommend gimp: (1) Make sure the *image* → *mode* is **RGB**. Then (2) choose *layer* → *transparency* → *color to alpha*. Click *ok*.

5. Add generality to IOmod by permitting the user to write text in a different color font.

6. The destructors in classes Engine and FrameFactory use while loops to delete memory. Convert these loops to ranged **for** loops (use **auto**).

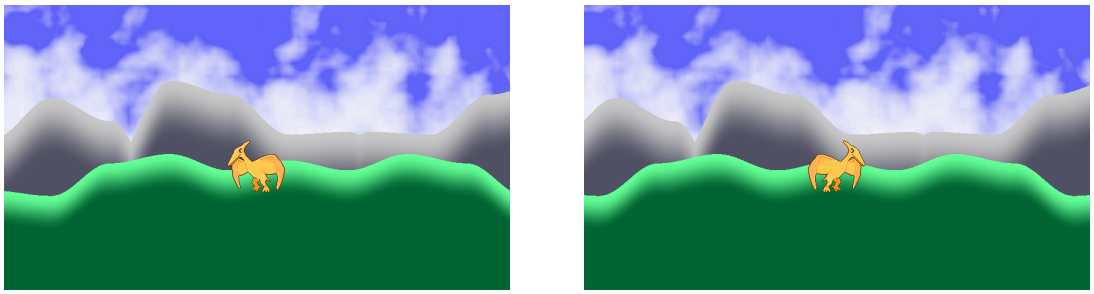
7. Use **delete** to "explicitly disallow compiler generated functions" in Engine, and FrameFactory.

8. Write an assignment operator for class Sprite. You can find an example of an overloaded assignment operator in the Meyer's text, Item # 12, page 59.

9. Modify the Clock class so that it averages the frame rate (fps) over the last frameMax frames, as specified in the XML file game.xml. Print the average fps in the upper left corner of the screen.

10. Convert the current *tracker framework* into a meaningful animation, using as many sprites as you need. Examples will be provided during lecture. This will entail replacing the images currently used in the *tracker framework* to use your images; you may not use my images in your animation. You may use images from the internet but you must cite the source in your README. Making your own images always makes for a more authentic animation.

    Adding new images will require that you modify game.xml, since all constants are read from this file, and modifying the constructors for Sprite and MultiSprite classes; **if you don't modify these constructors your sprites will appear on top of each other**. You may also have to modify the frameFactory class, since the frameFactory should manage all frames and textures in your animation.



(a) Pter flying left.

(b) Pter flying right.

Figure 1: This figure illustrates a two-way multi-frame sprite.

11. The Engine class in the *tracker framework* contains a polymorphic vector of type Drawable* consisting of either Sprite or MultiSprite. Extend your animation, and this vector, to contain a two-way multi-frame sprite. If you have an idea for an additional "sprite type',' you may substitute for the two-way multi-frame sprite, or simply implement both as an added feature, with the instructor's permission. A two-way multi-frame sprite is a sprite that can travel in two directions, such as the pterodactyl in Figure 1. You must have at least three different types of sprites in your polymorphic vector.

12. Your submission must contain no memory leaks in user code.

Making a Video of Your Animation:

There are two alternatives for making the video:

1. You do it by using the FrameGenerator class to generate the frames, and then use avconv to make an mp4; or, use any video capture tool that you like to make the vide.
2. We do it by using the constants that you have set in /xmlSpecs/game.xml to capture the video.

Class Engine contains code that, when F4 is pressed, will generate the number of frames specified in game.xml, currently specified as 300 frames. Please add more frames if needed to capture your animation. The initial viewport size is 854x480 pixels; if you keep close to these dimensions your animation will blend nicely with the others. Finally, modify game.xml so that the generated frame file names use your *username*, rather than mine. Use the same naming convention that we used in the previous project to name your frames.