Project #3

Tracking a Sprite

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

Computer Science Division, Clemson University

Brian Malloy, PhD

September 24, 2017

Due Date:

In order to receive credit for this assignment, your solution must meet the requirements specified in this document and be submitted, using the `handin` facility, by 8 AM, Thursday, September 21st, 2017. The handin close date is set at three days after the due date. If you submit after the due date but before the handin close date there will be a ten point deduction. No submissions will be accepted after the handin close date and no submissions will be accepted by email.

Project Submission:

To submit your solution through handin, copy the README file from the top project directory in the repo to your project directory, fill in the blanks in the README, make clean in your project directory, and compress the project directory using tar or zip.

Project Specification:

The purpose of this assignment is to: help you to become familiar with: (1) C^{++} language constructs including inheritance and polymorphism; (2) design patterns including factory and singleton; (3) data-driven programming, and (4) the tracker framework. You will begin the project by using the tracker framework, shown in Figure 1, provided in the directory for project #3 in the course repo. Starting with this framework, you must complete the following tasks:

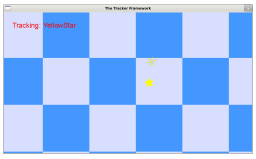
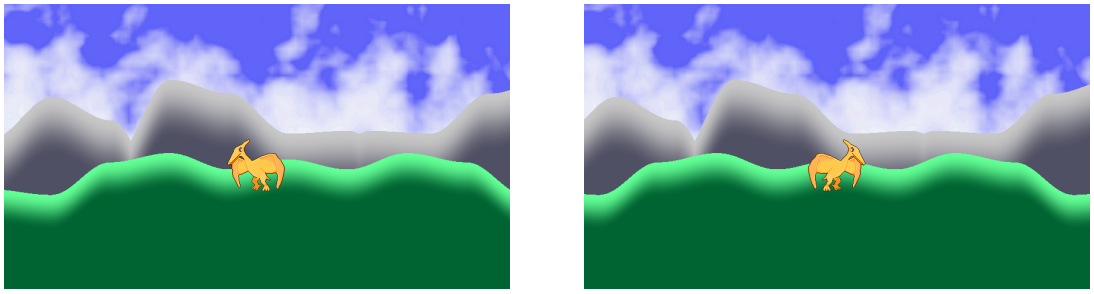


Figure 1: Tracker

1. Convert the while loops, in the following functions, into ranged for loops:
   - ParseXML::display(),
   - Gamedata::displayData()
2. Convert the ranged for loops in the ImageFactory destructor to while loops without using *auto*.
3. Convert ImageFactory from a GoF singleton into a Meyers singleton.
4. Use IOmod and C^{++} stringstreams to write the fps of your game to the screen, and write your name in the lower left corner of the screen.
5. Add generality to IOmod by permitting the user to write text in a different color font.
6. 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*.
7. Use **delete** to "explicitly disallow compiler generated functions" in Engine, and FrameFactory.

8. Convert the current *tracker framework* into a meaningful animation, using as many sprites as you need. Store the sprites in a polymorphic vector of Drawable*, This is explained in the next item and examples will be provided during lecture. Creating a meaningful animation entails replacing the images currently used in the *tracker framework* with 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. Creating your own images is time consuming but always makes for a more authentic animation and will earn more points.

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

9. With some small changes, the Engine class in the *tracker framework* will contain a polymorphic vector of type Drawable* consisting of either Sprite or MultiSprite.

   Extend your animation, and this vector, to contain a third type of spite: 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 2. You must have at least three different types of sprites in your polymorphic vector.

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

11. Incorporate class FrameGenerator into the Tracker framework so that F4 toggles frame generation.

12. A video of your animation. You can do this in one of two ways:

    (a) You do it by using the FrameGenerator class to generate the frames, and then use avconv to make an mp4; or, use a video capture tool to make the video. Submit the mp4 with your project.

    (b) Let us do it by using the constants that you have set in /xmlSpecs/game.xml to capture the video. Fix game.xml so that FrameGenerator captures enough frames to permit us to generate a video of your game with each frame consisting of your name as its file name prefix. The initial version of game.xml generates file names with *malloy* as the file prefix. Fix this so that your *username* is the file name prefix rather than mine.