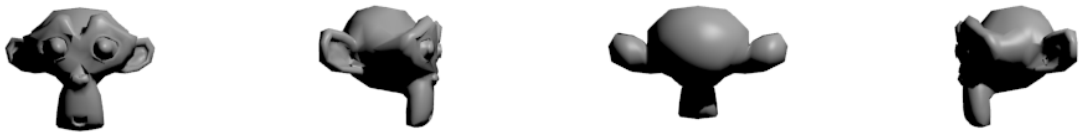
CS 145 Homework 5
Playright

1 Overview

Your objective in this homework is to introduce yourself to with working with arrays, files, and exceptions. You will do this in the context of producing an animation using a single image. Sounds crazy, huh? To get a feel for what inspired this homework, watch Rufus Butler Seder talk about his Scanimation books at `http://scanimationbooks.com/about-scanimation`.

Seder shows that a movie consisting of just a few frames can be compressed into a single static image. When you slide a custom filter over the static image, different frames of the movie become visible and the image appears animated. For example, consider the four-frame movie shown in figure 1.



(a) Frame 0 (b) Frame 1 (c) Frame 2 (d) Frame 3

Figure 1: Suzanne, mascot of the Blender modeling software, spins her head across four frames.

We compress the movie into a single image by copying columns 0, 4, 8, 12, and so on from frame 0. Columns 1, 5, 9, 13, and so on are copied from frame 1. Columns 2, 6, 10, 14, and so on from frame 2. And so on, for as many frames as there are. The result is shown in figure 2.

Figure 2: Frame 3

On its own, the image is hard to parse visually. Let's overly a sliding filter. We set the filter to block 3 columns at a time. As it slides from left to right, only one frame's columns are visible. Using a graphical user interface that we provide, we drag the filter to "reanimate" the movie. Snapshots of the animation are shown in figure 3.

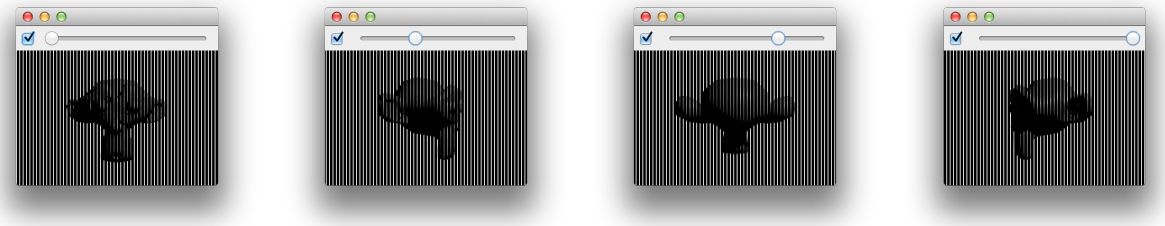


Figure 3: Viewing the static image with a sliding filter restores the animation.

2 Requirements

Complete the two classes described below. Place all classes in package `hw5`. Make all methods `static`.

2.1 Playright

Write a class `Playright`, which contains several methods for producing movies that trade space for time. We assume that the frames of a movie are stored in a directory somewhere on disk, that the directory contains only images and optional meta files like `.DS_Store` on OS X and `Thumbs.db` on Windows, and that the individual frames are named in dictionary order. That is, frame 0 comes alphabetically before frame 1, which comes alphabetically before frame 2, and so on. You are asked to complete the following tasks:

1. Write a method `getSortedContents` that accepts a `File` parameter for a directory. Return a sorted `File` array of the directory's contents. See `Arrays.sort` and `File.listFiles`. This method is used to put the list of frames in the proper order. `File.listFiles` by itself guarantees no particular order.

2. Write a method `filterMeta` that accepts a `File` array parameter. It returns a `File` array containing only those files whose names are neither `.DS_Store` nor `Thumbs.db`. Files in the returned array appear in the same relative order as they do in the parameter array. This method is useful for filtering out non-image files.

3. Write a method `readImages` that accepts an array of `File`s as a parameter. Assume all elements are image files. Return a new `BufferedImage` array, where each image element is read from the file on disk identified by the corresponding element of the `File` array. See `ImageIO.read`. This method does not have enough information to handle any exceptions. It passively throws any `IOException`s that come its way.

4. Write a method `compress` that accepts an array of `BufferedImage`s as a parameter. If the array has no elements, throw an `IllegalArgumentException`. Otherwise, create a new `BufferedImage`, column 0 of which comes from column 0 of image 0, column 1 of which comes from column 1 of image 1, column 2 of which comes from column 2 of image 2, and so on. Once you get into columns that exceed the number of images, start back at image 0 again. Return the single `BufferedImage` that you produce.

5. Write a method `compressAndShow` that accepts a `File` parameter for a movie directory. It sorts the directory's children, filters out the meta files, reads in the images, and compresses them into a single image. It then displays the movie in a viewer that we have provided. The viewer may be downloaded from `http://www.twodee.org/teaching/cs145/2015c/homework/hw5/PlayrightViewer.java`.

   Construct a viewer in the following manner:

   ```
   new PlayrightViewer(image, nFrames);
   ```

   Note that it expects both the single movie image and the number of frames that have been spliced together. In the viewer, drag the slider to advance frames. Uncheck the checkbox to see the spliced image in its entirety.

2.2 Main

Write a class `Main`. It has a `main` method, which prompts a user to select a directory containing frames of a movie. It then compresses the directory into a single image and displays it using the `PlayrightViewer`. If the compression fails for any reason, the method should start over from the beginning, prompting the user for another directory. Use the *loop until fixed pattern* we discussed in lecture:

```
isValid = false
while !isValid
  try
    attempt the dangerous
    isValid = true
  catch
    print "Uh oh! Let's try that again."
```

One can pop open a file picker dialog for directories using the following code:

```
// Show a file picker for directories only.
JFileChooser chooser = new JFileChooser();
chooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
int result = chooser.showOpenDialog(null);

// If the user hit Okay...
if (result == JFileChooser.APPROVE_OPTION) {
  File selectedDir = chooser.getSelectedFile();
}
```

3 Extra

Any student who posts on Piazza a sequence of images and the resulting compressed static image under folder `hw5_extra_credit` will receive one extra credit participation point. Students who post an accompanying movie (captured via phone or recording software) will receive two.

4 Submission

To submit your work for grading:

1. Put the SpecChecker for this homework in your Build Path. Run the SpecChecker as a Java Application and fix problems until all tests pass.

2. Commit and push your work to your repository. Verify that your solution is on Bitbucket.

A passing SpecChecker does not guarantee you credit. Your grade is conditioned on a few things:

- You must meet the requirements described above. The SpecChecker checks some of them, but not all.

- You must not plagiarize. Write your own code. Talk about code with your classmates. Ask questions of your instructor or TA. Do not look at others' code. Do not ask questions specific to your homework anywhere online but Piazza. Your instructor employs a vast repertoire of tools to sniff out academic dishonesty, including: drones, CS 145 moles, and a piece of software called MOSS that rigorously compares your code to every other submission. You don't want to live in a world serviced by those who squeaked by through questionable means. For your future self, career, and family, do your own work.

- Your code must be submitted correctly and on time. Most excuses devolve into, "I started too late." The fix for this problem is not an extension.