Overview of FFMPEG Modifications to Compositing an Animation

Ryan Saunders and Wesley Gormley

Overview

This project is about creating a product that uses the ffmpeg libraries to composite a sequence of animated images into a movie format. We will be modifying the ffmpeg library to include a new image-animation format, .xkcd. We will also create a stand-alone application, bouncer, that will create .xkcd image files, in sequences. The sequence will be created by the user choosing a background image, and the bouncer application adding an animated bouncing ball to that image.

So far, we have set up everything needed to start working on the changes to the ffmpeg library, and for creating the bouncer application. We have reviewed the overall project requirements. Then we installed ffmpeg on both of our machines. Afterwards, we analyzed the structure of ffmpeg, and started reviewing the code of important libraries. Finally, code was added to an ffmpeg library to indicate where we may need to start incorporating our modifications.

There are a few obstacles that will need to be resolved. The first is compressing pixels into a byte of data. This will require some research on different image compression algorithms. The second obstacle is confirming that we can convert from a .png file format to a .xkcd. To do this, we will need to determine how a .png file encodes to a byte stream. The third obstacle will be drawing a ball with a gradient. Without outside libraries, this drawing process will need to be done one pixel at a time. We expect to have to work through the mathematical implications of rendering a circle. There are doubtlessly other problems that we will run across, and we will make sure to give ourselves enough time to compensate for other obstacles.

When the project is completed, a user should be able to run the bouncer application and give it

a background image. The bouncer application will then make the sequence images. Then the user can run ffmpeg to create a movie from the .xkcd files.

Work Completed

There were several steps that were taken overall to complete checkpoint #1. The project specifications were reviewed together, ffmpeg was installed, and the necessary code was added to one of ffmpeg's libraries.

We have begun to research the ffmpeg libraries. The online documentation for ffmpeg was reviewed, and the documentation needed specifically for this project was thoroughly read. We worked through the call stack of ffmpeg and ffplay to determine what libraries were being used and how they were connected. We also used grep to search for specific terms and function/struct definitions.

Through our research we have come to better understand the ffmpeg structure and what library and files we will to replicate and analyze. Ffmpeg's executables are stored within the bin folder, which will be where we store our bounce executable. The main folder contains all of the source code for the executables and will be where we save our bouncer and ball .c and .h files.

Further, ffmpeg is divided into different folders. The two main folders that we will be using are libavcodec and libavformat. Inside of libavcodec are the source files for all supported ffmpeg files. We will be adding our .xkcd files to this directory. The libavformat folder contains information crucial to opening and modifying the different file types in ffmpeg. We will need to include data for our .xkcd file type to ensure that executables like ffplay and ffmpeg will know how to open and modify our file type.

After we better understood the layout of ffmpeg, we were easily able to add the "breakpoint-like" code into the ffmpeg library as specified in checkpoint #1.

Details

We have divided the project into six steps that we believe will help us accomplish this project efficiently and correctly. Our main goal of the first step is to create a .xkcd encoder. The encoder will be able to read in the data from an .xkcd file and save it to a bitstream. This main part of this step will be compressing the pixel data to an appropriate size. This will most likely require the most time because an algorithm must be written to take the three bytes, which an RGB pixel is usually stored as, and compressing that to one byte. We anticipate this step will take five hours and will be finished and tested before Saturday, March 1st.

Next we will work on the decoder for the .xkcd file. This step will require us to take a bitstream and create a .xkcd file from its contents. This step should be pretty straight forward since the bitstream will contain the pixel data exactly as we need it so no compression or advanced algorithm will be needed. We anticipate this step will take no more than 2.5 hours and be done by Saturday, March 1st.

For the third step we will work on converting a .png file to an .xkcd file. This should be a pretty trivial step since at this time we will already have our .xkcd decoder working. We will rely upon the .png encoder to convert the .png file to a bitstream. We will then send this bitstream to our .xkcd file decoder and since the bitstream will contain universal pixel data, retrieving the image from the bitstream should be easy. The most difficult part of this step will probably be deciphering any nuance that ffmpeg uses with the .png files. We anticipate 2.5 hours for this step and to be done by Tuesday, March 4th.

For the rest of the project we will be working primarily within the bouncer application. We will start by taking in a single .xkcd file and using our encoder to get out all of the pixel data. Next we will manipulate the pixels to produce a circle with a gradient. Changing these pixels should be fairly easy, the greatest challenge will probably come from writing an algorithm that can deal with different picture sizes

and ratios. We have already discussed this some and feel we have a decent idea to eliminate any complications that can arise from varying picture sizes. We believe this step will take five hours and will be finished on Saturday, March 15th.

Step five will be building upon what we have done in step four. To complete the bouncer application we will need to write code that will create a sequence of images and draw the ball in these images in varying locations. Creating the sequence of images should not be difficult since ffmpeg can do this and can be used as a reference. The more difficult part will be determining how we want the ball to bounce, and how we wish to pass this information to the function that will draw the ball. This will take some trial and error to create a bounce that appears fluid. We anticipate this step will take two hours and will be done on March 15th as well.

Our last step will consist of testing the project as a whole. When we finish each of the previous five steps we will have done some localized testing to ensure that each part is producing the correct results, but at this point we will do more exhaustive testing. We will search out the edge and extreme cases to ensure that our program is performing as expected and without any unnecessary time or computation bottlenecks. We anticipate that all testing and debugging will take fours hours and will be done by Tuesday, March 18th. Planning on finishing all of these steps a few days before the due date will allow us some flexibility in our schedule in case some steps take longer than we expect.

Step	Estimated Time to Complete	Completed by
Create .xkcd encoder	10 hours	Ryan Saunders
Create .xkcd decoder	5 hours	Wesley Gormley
Convert .png to .xkcd file	7 hours	Both
Draw ball on single frame	2 hours	Ryan Saunders
Create an animation represented as a file sequence	2 hours	Wesley Gormley
Finish testing project	6 hours	Both

Summary

Once the project is completed, it will be a compilation of the ffmpeg library, added libraries to the ffmpeg library, and our bouncer application with its libraries. The added libraries to ffmpeg will be xkcd.c, xkcd.h, xkcdenc.c, and xkcddec.c. The bouncer libraries will include bouncer.c, bouncer.h, ball.c, ball.h, and bouncer. In addition, .xkcd image files will be created by the user running the bouncer application.

With the additional libraries and application, the user could run bouncer with a given .png image as a background. Bouncer will use ffmpeg to convert the .png file into a bytestream. The bytestream will then be modified to render a drawing of a ball, using the ball library. Finally, the bytestream we be converted into a sequence of .xkcd image files. Each file will represent a frame in the animation, where it will appear as if the ball is bouncing on top of the given background image.

The user can then convert these .xkcd image files into a movie. This process will be done in ffmpeg, and will use the .xkcd codex to render the movie. We will know that the project works when the user can see a proper rendering of the animation. This rendering should be able to be done on any .png image, regardless of size or complexity of the image.