**Group 4 Video Wall**

**Background**

The purpose of this project was to create a video wall program that could display a wall of video profiles and display the corresponding from a movie file when a profile is clicked. When the user zooms in or out using their mouse, the profiles should be resized to be either bigger or smaller. Finally, if the stacked profile images are bigger than the window of the interface, then the interface should provide a scroll bar so that you can view all of the frames.

**Goal**

Originally, our group had ten members assigned to it, but only four students worked on this project. One of the most difficult parts of this assignment was to split the work into parts that each person could work on. We handled this problem by coming up with a list of classes that needed to be built in order to complete the project. By having a list of classes that were required, we were able to assign work to each person so that everyone would be responsible for a part of the system.

**Method And Tool**

The group worked by meeting on class days after class was over. The majority of the work that was done by the group was done individually out of these meeting times. When we meet after classes, those meetings were simply to plan the development of the project and discuss any problems that you may have run into.

The group used Github (with git source control) in order to manage the code itself during the development of this project. Some of the members of the team already had experience with version control, which made using git easy, and a couple members had to become familiar with git in order to contribute to the project. With git, we were able to read each other’s code and add to it in a flexible and easy way. Github allowed us to work together, but in an independent fashion.

**Software Code**

This project was build using the Java programming language. For one aspect of the project, displaying frames from an AVI video file, we had to use the OpenCV library with Java bindings. All other code relies on the standard Java libraries.

This project was divided into 5 major components (classes). Each class has it’s own responsibilities and was developed independently. After the development of each individual class, we had to work together in order to integrate all of the parts into the final system. The major components of the project were: VideoWall class, FrameViewer class, AVILoader class, VideoClipCollection class, and VideoClip class.

The VideoWall class is initialized by the main function of the program. It’s mainly responsible for displaying video profiles in the GUI, and configuring the GUI. The VideoWall class also serves several other purposes. It must be able to resize the profiles when the user zooms in or out, it must be vertically scrollable, and it must be able to calculate where the user may click. The VideoWall class initialized many of the classes that are used in the project, and a lot of the integration work that had to be done was within this class.

The FrameViewer class is responsible for displaying a frame after the user clicks on the GUI. Given an image buffer, the FrameViewer must calculate the best position for displaying the frame (so that it is easily viewable and inbounds), and it must also draw a border around the frame to increase the usability of the program.

The AVILoader class is a wrapper for the OpenCV library. Given the time of the frame and a path to a video file, the AVILoader class returns an image of the frame from the video that can then be used by the FrameViewer class to display the frame to the user.

The VideoClip class provides an easy way to access either the profile of a video clip or the clip itself. Given a directory path, the VideoClip object will store the location of profile images and the clip itself. The VideoClip class does some work in order to validate that the contents of the given directory are valid, and additionally any garbage files are filtered out.

The VideoClipCollection class is simply a container of VideoClip objects. The purpose of this class is so that you can organize your video/video profile data in an organized way in the project file directory, and also that there is some organization of the data within the project. You initialize the VideoClipCollection class with a directory path that contains many subdirectories. Each of these subdirectories contains the contents needed to create a VideoClip object. So when the VideoClipCollection object is initialized, it looks through each subdirectory and initializes a VideoClip object for each one it finds. The VideoWall class is initialized by providing a VideoClipCollection with the required data.

**Implementation Evaluation**

We believe that our class hierarchy and design provides a simple and flexible approach to the video wall program problem. The GUI is snappy and highly reactive to the user input, and each component class works together to create a complete program.