Group 4 - MP 1

Our approach to this MP is to have 1 complete audio/video recorder and 1 complete audio/video player. At first we want the program to run in an linux environment, but due to some of the GST-plugin that only work for MAC, we choose to make this a MAC-only project.

User Manual

Compile and Installation:
Open the project with Eclipse. There are two different programs: Recorder and Player. To run the Recorder, use Eclipse to run RecorderGUI.java as main(). To run Player, choose VideoPlayer.java

How to run and test features:
For Recorder:

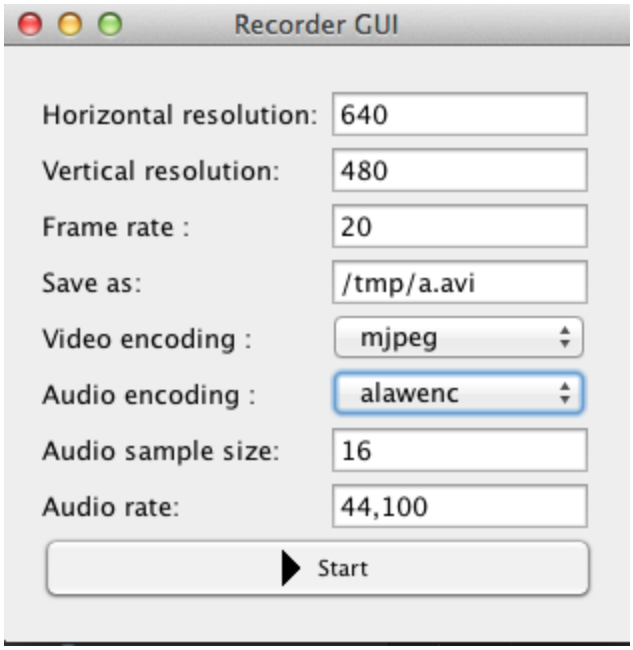
- Acceptable resolution (due to plugins' limitations): 640x480, 176x144, 160x120, 320x240, 352x288
- Acceptable audio sample size (due to alaw and mulaw encodings): 16
- Other attributes can be anything
- Click the button to start recording
- Live video will be displayed
- Both video and audio are saved to the same file
- Monitoring information is displayed via standard out

For Player:

- Run VideoPlayer.java in src/mp1
- Type the path of an exist local media file
- Click play button to start playing
- Use pause button to pause the video/audio
- Use fast forward and rewind button to adjust time

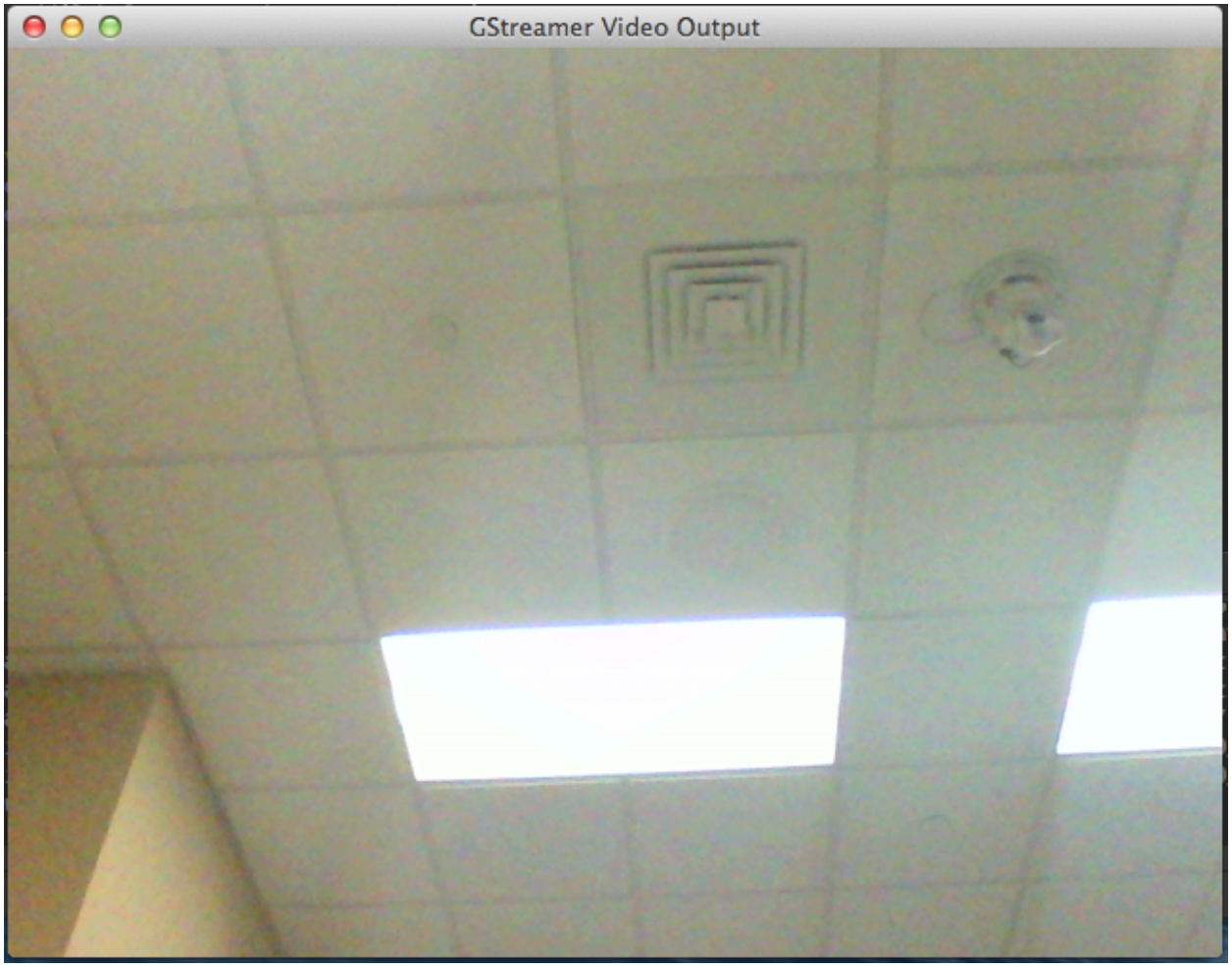
GUI:

Recorder GUI:



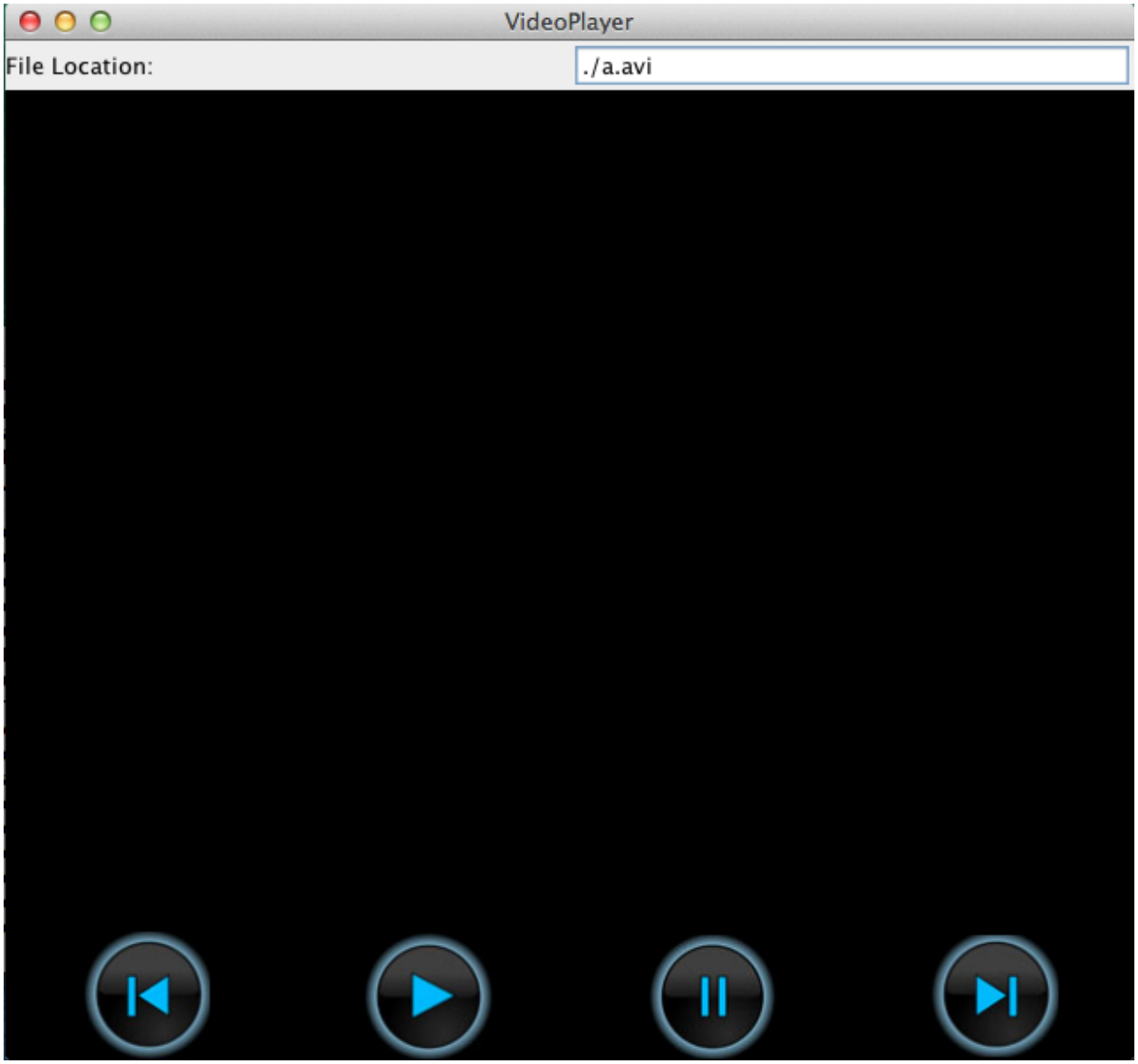
As shown here, user can input multiple properties of the webcam recorder:
Horizontal Resolution: Width resolution of the recorded video (default 640)
Vertical Resolution: Height resolution of the recorded video (default 480)
Frame rate: The recording rate (default 20)
Save as: The location where the file is saved (default /tmp/a.avi)
Video Encoding: The video compression technique (default mpjeg)
Audio Encoding: The audio compression technique (default alawenc)
Audio Sample Size: 16
Audio Rate: [8000-90000] (default 44100)

Pressing start will launch a rectangular screen that uses a webcam and a microphone. To stop the recording, simply close the outputted screen.



The recorded file will save to the location that is specified previously.

Video Player GUI:



File Location: location of the file to play
Button from left to right:
1) Rewind (send a few frame back)
2) Play
3) Pause
4) Forward

Development Manual

Implementation of Recorder:

Files relevant: **RecorderGUI.java, Recorder.java**

RecorderGUI.java: The GUI aspect for the audio/video recorder.
Recorder.java: The implementation of the recorder.

The GUI has multiple fields where the user can input values that correspond to its respective properties. Once “start” is clicked, RecorderGUI.java will create a new recorder and run it.

For the video to start recording, we need to use a pipeline. We also need to create multiple different elements that will use this pipeline:

Elements	**GST-Plugin Elements**	**Description**
Camera	“autovideosrc”	Wrapper video source for automatically detected video source
VideoFilter	“capsfilter”	Pass data without modification, limiting formats
Microphone	“autoaudiosrc”	Wrapper audio source for automatically detected audio source
AudioConvert	“audioconvert”	Convert audio to different formats
AudioEncoder	*“alawenc” or “mulawenc”*	Convert 16bit PCM to 8bit A law / Convert 16bit PCM to 8bit mu law
VideoRate	“videorate”	Drops/duplicates/adjusts timestamps on video frames to make a perfect stream
ColorConverter	“ffmpegcolorspace”	Converts video from one colorspace to another
VideoEncoder	“ffenc_mpeg4” or “jpegnenc” (inputted values from GUI)	FFMPEG mpeg4 encoder / Encode images in JPEG format
LiveSink	“autovideosink”	Wrapper video sink for automatically detected video

		sink
mediaMuxer	“avimux”	Muxes audio and video into an avi stream

Because our recorder record and plays the video in parallel, we have one thread that plays the video and another thread that records both the audio and video. We also used the tee element so that is possible.

Implementation of Player:

File related: VideoPlayer.java

The player first create a GUI with buttons, text field for file path and a video component and use ActionListeners of buttons to trigger events. It uses PlayBin to process the pipeline and handle different file formats.

When play button is clicked, check whether the text field contains a path that’s different from the one that’s playing at the moment. If a new path is in the text field, a new PlayBin will be created with new file path and its status will be set to playing; if it’s still the same path, set the current PlayBin to playing.

.

When pause button is clicked, set the PlayBin’s status to pause.

When rewind or forward button is clicked, get the current playing time by query the PlayBin, then construct a seekEvent which changes the start time of PlayBin’s playing segment.

The player only plays one file at a time. If a new path is added and play button is clicked while a video/audio is still playing, it will stop the current showing and start playing new content in the same window.

Implementation of Monitor:

- For recording, the monitor includes 2 AppSink elements: AppSink0 is placed before the encoder and AppSink1 is placed after it. There are also 2 sub threads, one of which keeps pulling raw frames from AppSink0 and save into a buffer. The other thread takes care of calculating monitoring information. It does so by pulling

encoded frames from AppSink1 and comparing each of which to it corresponding raw frame in the buffer.

- For decoding, the monitoring architecture is the same except that AppSink0 and AppSink1 are placed at the 2 ends of the decoder instead of the encoder.