Machine Problem 1: Video/Audio Recorder and Player

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Feb 2, 2014 Due: 5:00pm Feb 19, 2012

1. Introduction

Watching movies is a great source of fun for all of us. But have you ever wondered how videos and sounds are captured and transformed into binary bits in your PC? How does your video player provide cool functionalities like fast forward (FF), rewind (R), play (P), and pause (S). In our first MP, we are going to explore and learn some of these techniques. In summary, our learning objectives are follows:

1. Capturing audio and video using microphone and webcam
2. Storing the audio and video file in PC using compression
3. Playing a video from the stored file with fast forward, rewind, pause and play functionalities
4. Playing an audio from the stored file
5. Comparing performance of different compression (encoding) and decompression (decoding)

2. Work Environment

You are recommended to work on Linux for this machine problem. The Linux workstations are available in the EWS lab rooms (SC218, SC220, and SC222.) A group directory with sufficient disk space will be set up for everyone. You can use your own machine as well (linux, mac or windows). The recommended Linux version to install is Ubuntu 10.04 or newer. Java is the recommended language for your program and gstreamer-java is the recommended library for your work. However, if you use C/C++ or other language and windows or mac, you have to deal with installation, driver and library problems on your own. Each group can borrow two Logitech QuickCam Orbit MP Webcams from Engineering IT (Barb Leisner). The Logitech webcam can also be used as the microphone for audio recording. You need to prepare your own headphone/speaker to test the sound playback.

3. Implementation

In this section, we start by giving the overview of the system that you are going to implement and name the list of parameters that you need to consider. Finally, we show several functional components that you can use to design your MP. We also provide some design examples at the end.

Machine Problem 1: Video/Audio Recorder and Player

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Feb 2, 2014 Due: 5:00pm Feb 19, 2012

3. 1 System Functionality

Implement three logical processes named as **recorder**, **player** and **monitor**. Depending on the given parameters, **recorder** should be able to capture the video frames from the webcam and the audio data from the microphone, compress the data and save them in separate files. Similarly, depending on the given parameters, **player** should decompress and then play the media files saved by the **recorder**. In each case, the **monitor** should monitor the compression and decompression time of each video frame, size of the compressed frame, and the compression ratio.

3. 2 System Parameters

The tools should be able to consider the following parameters in recording, playback and monitoring phase. **[Note: There will be no command line. Please use GUI to input those parameters]**

Recorder

Video

The video recording should take the following parameters.

-**source** the source of the recording (*default v4l2src*)
-**height** The height resolution of the recorded video (*default 640*)
-**width** The width resolution of the recorded video (*default 480*)
-**rate** The recording rate (*default 20*)
-**encoder** The video compression technique (*raw* [*default*], mjpeg, mpeg4)

Audio

The audio recording should take the following parameters.

-**source** the source of the recording (*default hw:1*)
-**rate** The audio sampling rate (*default 44,100 Hz*)
-**channel** The number of audio channels (*default 1*)
-**encoder** The audio compression technique (*raw/pcm* [*default*], vorbis)

Player

Video: The video player should take the following parameters.

-**source** The filename or recording source

Audio: The audio should take the following parameters.

-**source** The filename or recording source

Machine Problem 1: Video/Audio Recorder and Player

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Feb 2, 2014 Due: 5:00pm Feb 19, 2012

Monitor

The monitor should monitor the following parameters for each video frame.

-**compression_time** The time to compress a video frame
-**decompression_time** The time to decompress a video frame
-**frame_size** The size of the compressed video frame
-**compression_ratio** size of the raw video frame/ size of the compressed video frame

3. 3 System Components

This section helps you to design your MP. You will need several basic components for this MP. You can use any libraries (gstreamer-java is preferred) to implement those components.

a) **Media Source:** This provides media interface to capture the media data (audio and video). For video, this component should be configured with **v4l2** interface and for audio, the component should be configured with **alsasource** interface with **hw:1** as the audio device (use command **$cat /proc/asound/cards** to confirm).
b) **Media Filter:** This component helps to modify the media frames with given requirements (such as video resolution, video capturing rate, audio sampling rate and so on)
c) **Media Encoder:** This component uses standard encoding library to compress the media data
d) **Media Decoder:** This component uses standard decoding library to decompress the media data
e) **Media Sink:** It gets the final frames. Depending on the user command, it can store the frames in a file or play them. For video playback, the sink displays the frames on the graphical window. However, for audio, you can use "**alsasink**" device to send the frames to the soundcard to play.

There are two optional components. You can use them if you want.

f) **Media Muxer:** It helps you to multiplex the media data. It is necessary when you want to put audio and video data in the same file. Even with only the video data, using standard muxer library helps your video file to run using the standard media players.
g) **Media Demuxer:** This component uses the standard demux library to de-multiplex the stored video file to get the video frames. This is necessary to run different movie file available in the Internet.

NOTE: you may need some other components as well to monitor per frame metadata such as compression/ decompression time, frame size and compression ratio in parallel.

Machine Problem 1: Video/Audio Recorder and Player

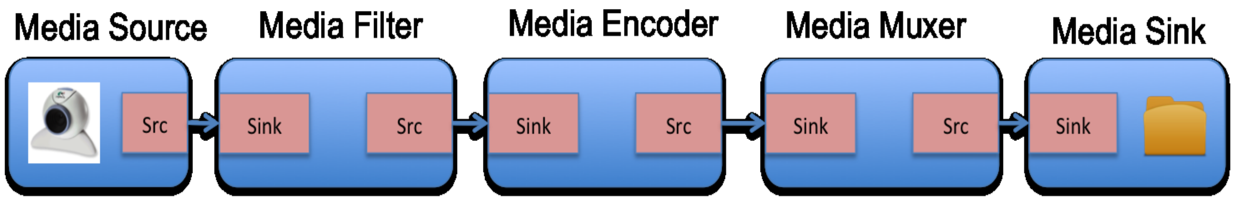
CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

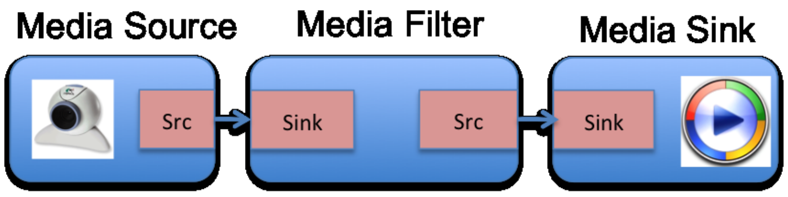
Posted: Feb 2, 2014 Due: 5:00pm Feb 19, 2012

To understand how the system works, we present the interaction among the components using several examples. Remember that, muxers and demuxers are optional components. Use them if you need it. [**Note: these are generic examples, not gstreamer-java specific examples**]

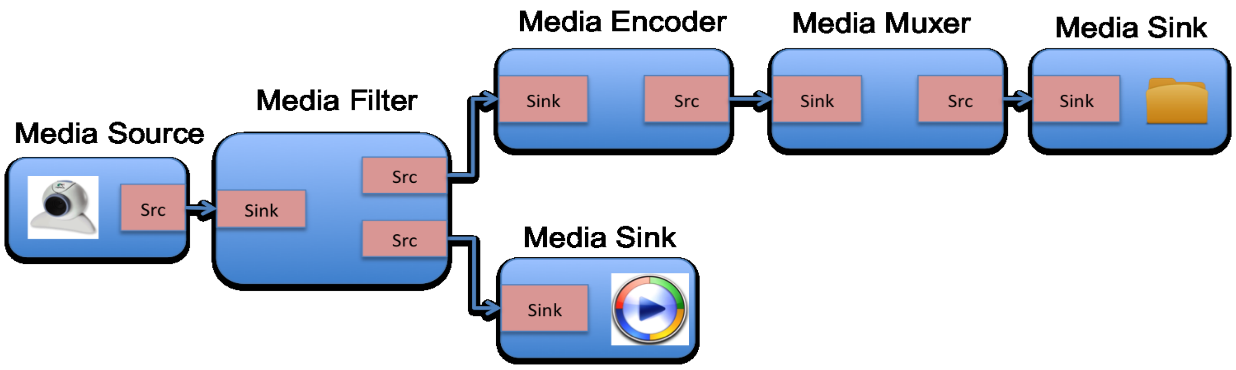
1. *Capturing media (video or audio) from the webcam and storing in a file*
   Here, webcam is the media source and file is the media sink.



2. *Capturing media (video or audio) from the webcam and playing it on the player*
   Here, webcam is the media source and player is the media sink.

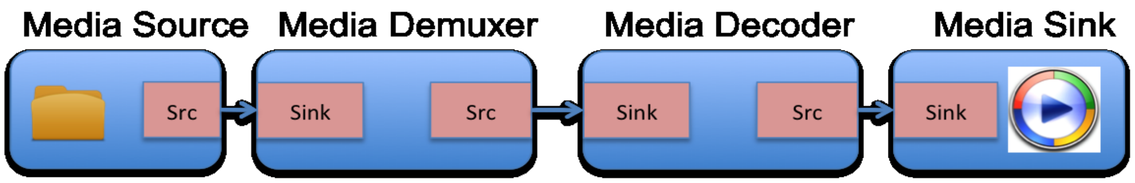


3. *Capturing media (video or audio) from the webcam, storing in a file and playing on the player* ***[You should have parallel threads to do this]***
   Here, webcam is the media source and file and player both are media sinks.



The media sink (player) in the second pipeline should be executed in parallel with the first pipeline. Therefore you should consider them running in different threads. [*Hints: you may need tee element to implement this in gstreamer-java*]

4. <u>*Capturing media (video/audio) from a file and playing it on the player*</u>
   Here, file is the media source and player is the media sink



[**Note:** you will need a monitoring component in different thread for each example shown above]

4. Required Features (100 points)

Video recording (15 points): ***recorder*** can access the webcam through ***V4L2*** interface, and save the captured raw image frames in a file with given resolution and frame rate. *The video should be played during recording in a different thread.*

Video compression (10 points): *recorders* should be able to record video in different compression formats mentioned above (mpeg4, mjpeg).

Audio recording (10 points): ***recorder*** can access the webcam microphone and save the captured raw sound in a file with given sampling rate, sample size and number of channels.

Audio compression (10 points): ***recorder*** should be able to record audio in different compression formats (mu-law, a-law).

Video playback (15 points): *player* can play the video from the stored file. The user should be able to pause, play, rewind or fast-forward the running video. Identify the decoder using filename extension.

Audio playback (10 points): *player* can play the audio from the stored file. Identify the decoder using filename extension.

User-friendly GUI (10 points): provide a nice GUI (Graphical User Interface) that allows all the functionalities in a user-friendly way.

System monitor (10 points): You should have ***monitor*** to compute the video compression and decompression time per frame as well as compression ratio and compressed frame size in different compression techniques.

Report writing (10 points): Submit a complete report explaining your design and approaches. It should also include how to compile and run your code. Your report should be written well so that we can run your code using instruction in the submitted document. The report should include two parts: user manual and development manual (see the submission section for details).

5. Optional Features (20 points)

Performance Visualizer (10): Add an additional window in the GUI that shows the **System Monitor (**explained before) using graphical plots such as (time vs. average frame size), (time vs. average compression ratio), (time vs. average compression time) and (time vs. average decompression time). Increment time with 5 seconds interval and compute the average value (of compression ratio, compression time, decompression time, and frame size) for each 5 seconds and plot them in run time.
Camera pan and tilt (10): Add the functionality to pan and tilt the webcam any time (before, after or while recording).

6. Submission

Pack all source codes and documents into a zip file or tar ball and submit them through Compass2g. You will find the submission link after you login the system. Each group only needs to submit one copy and we will only grade your latest submission. Do not submit your solutions through email unless there are technical problems with the Compass system. The submission deadline is Feb 19th at 5:00 pm. You can get up to two days bonus for each MP, but please remember you can use only 3 bonus days throughout the semester. Further late submissions are not accepted and will get 0 point.

6.1 Source Code

You should only submit your source codes (.java, .c or .cpp files), Makefile, and any open source libraries (or jar files) you use in your solution into **compass2g**. Do not include any pre-compiled obj files (.o), binary execution, or pre-recorded media files in the directory.

6.2 Documentation

Your documentation should include two parts: user manual and development manual. The user manual should include all instructions on how to compile and install your source code, and how to run your program and test all features. If you have designed any GUI, it is better to attach some screen shots to explain. The development manual should have the implementation details of all features. The implementation details include program flow, key data structures; media file formats, important algorithms, and so on.

7. Evaluation

Machine Problem 1: Video/Audio Recorder and Player

CS414 Spring 2014: Multimedia Systems

Instructor: Klara Nahrstedt

Posted: Feb 2, 2014 Due: 5:00pm Feb 19, 2012

The evaluation will be done by face-to-face interview with each group. You will run your program. We may ask you to show the source code implementing different functionalities. Your solution is evaluated based on how many features you have implemented and demonstrated. In order to get full score (100 points) of this MP, you need to implement all 100-point required features. 20-point optional features will be used to offset your lower marks at MPs, homework's or exams. Evaluation will happen on Friday Feb 21th at lab SC218 at 5:00~7:00 pm. A sign-up sheet will be provided (by email or using piazza) to schedule your demonstration slot.

8. Links on gstreamer and gstreamer-java

http://wiki.oz9aec.net/index.php/Gstreamer_cheat_sheet

http://wiki.laptop.org/go/GStreamer

http://code.google.com/p/gstreamer-java/

http://lac.linuxaudio.org/2010/download/GStreamerAudioApps.pdf

http://code.google.com/p/gstreamer-java/wiki/Tutorials

Please attend the lecture on February 7st to get more helpful tips on this MP