Term Project

Summary report

Implementation details

Opening the video file

The video file is opened using the VideoFileReader class from the Aforge library.

After opening the video the total number of frames is saved in a global variable.

Default percentage of frames to include in the final gif file is set to 20%. The 20% percentage value is translated to an approximate number of frames to skip.

->The test video is a short video file of 474kb with a length of 6 seconds.

->the number of frames is 189. This translates to approx. 5 frames skipped with the percentage set to 20%.

-> This is a rough setting to have some control over the final gif file size.

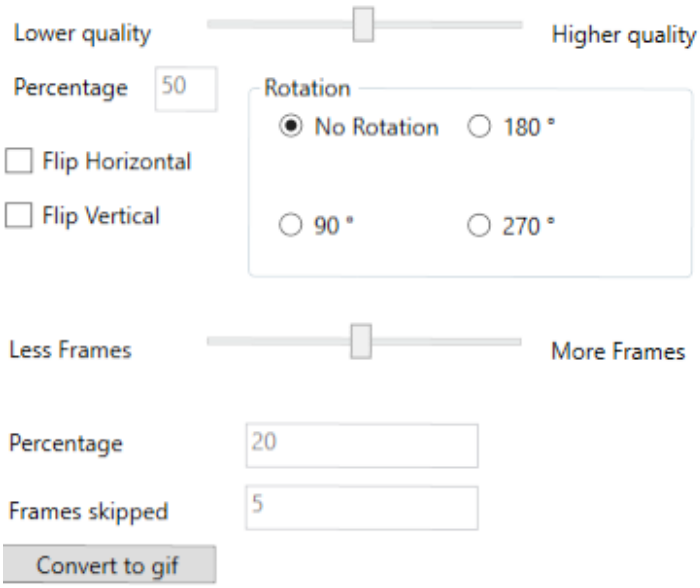
->The “Preview Video Frames “uses an event handler mechanism to seek in the video file. This is done using the Mediatimeline class from the Windows.Media library.

->Every time the slider value is changed the media element seeks in the file and updates the display.

Videoframes sampling and gif file creation

Opening the file enables the button “Convert to gif”

Before clicking this button the user can customize the output.



->The maximum frameskip value is 50%. Higher than that would create very large gif files.

The first slider at the top sets the quality level of the frame. Each frame will be encoded by the JpegEncoder and the quality level will be set using this slider.

The second slider at the bottom sets the amount of frames to cover in the gif file in percentages. This value is converted to number of frames to skip using the formula.

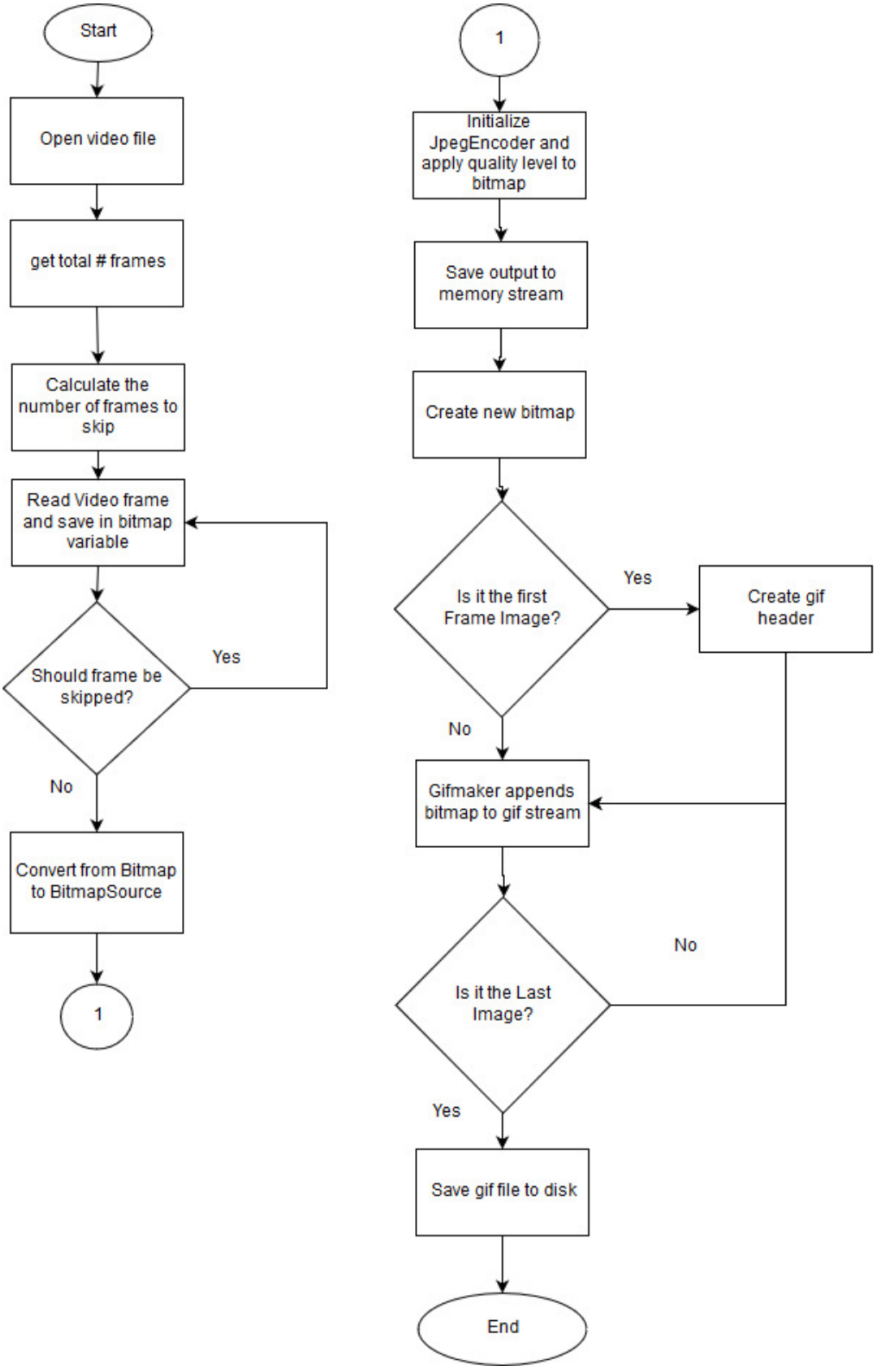
```
gifcoverage = (decimal)(framecount / (framecount * ((double)percentage / 100)));
                gifcoverageroof = Convert.ToInt32(Math.Round(gifcoverage));
```

$$Framesskipped = \frac{Total\ frames}{Total\ frames * \left({Percentage}/{100}\right)}$$

The result is rounded to whole numbers.

The modulus % operator is used to skip the frame in the for loop using the if statement: `if (i % framesskipped == 0)`

Program Flowchart



Each frame of the video file that is not skipped is read using the function ReadVideoframe() from the Videofilereader class.

This videoframe is then saved in a Bitmap variable.

A JpegBitmapEncoder class from the Windows.Media library is used to manipulate the quality of the videoframe.

The quality slider is used to choose the quantization quality.

The JpegBitmapEncoder does not Accept the Bitmap variable, so it must be converted to the BitmapSource variable.

A separate function “Converttobmpscr()” has been created to do this conversion.

The JpegBitmapEncoder saves the result jpeg file to a file. For this application, this is not needed. Instead the resulted jpeg should be saved to memory so it can be used for the creation of the gif file.

A MemoryStream variable is used to save this jpeg data. The memory stream is then placed in an Image variable so it can be converted back to the Bitmap format. For the creation of the gif file itself the class Gifmaker is used. This class implements the 89a specification of the gif standard.

The following header data is used:

```
private const string FileType = "GIF";
private const string FileVersion = "89a";
private const byte FileTrailer = 0x3b;

private const int Extension_Introducer = 0xff21;
private const byte Size_of_Extension_Block = 0x0b;
private const string ApplicationIdentification = "NETSCAPE2.0";

private const int Graphic_Control_Label = 0xf921;
private const byte Size_of_remaining_fields = 0x04;

private const long SourceGlobalColorInfoPosition = 10;
private const long SourceGraphicControlExtensionPosition = 781;
private const long SourceGraphicControlExtensionLength = 8;
private const long SourceImageBlockPosition = 789;
private const long SourceImageBlockHeaderLength = 11;
private const long SourceColorBlockPosition = 13;
private const long SourceColorBlockLength = 768;

private const byte Block_Terminator = 0;
```

The AddFrame(Image img) function of the Gifmaker class takes the global color table info from the given img variable. If it is the first file then the GIF header is initialized. If not then the Graphics Controlblock is written including the framedelay value to specify the delay between frames.

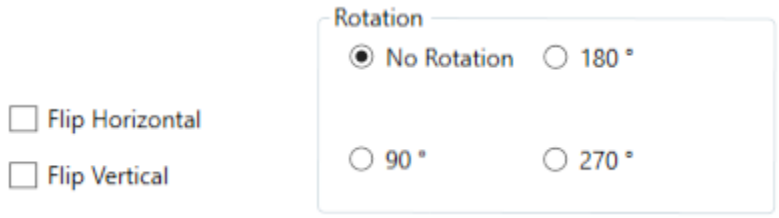
Each Bitmap frame is added to the gif encoder variable so that it can be appended to the gif file.

After doing these operations the Dispose() function of the videoframe should be called so that the memory is freed. The file pointer automatically proceeds to the next frame for the next iteration after calling Dispose().

After the gif is complete it is saved to disk.

Additional functionalities

The user can manipulate the position of the gif file by flipping and rotating the frames of the video file.



This functionality is implemented using the JpegEncoder Class before adding the frame to the GifEncoder.

Results

Frameskip results

For the default settings of 20% coverage of the video file with a frameskip value of 5 frames the resulting file was 2.73 Mb.

By setting the value to 50 % the result is 6.83 MB.

The lowest value (0%) of the frameskip results in just 2 frames. For Gif this is not so meaningful. The result is 149 Kb.

Quantization quality result

For this I used the frameskip value of 20% and compared with the lowest and highest quantization values.

The lowest quality value of 1% resulted in a file size of 2.25 MB.

The highest value of 100% resulted in a file size of 2.73 MB.

Additional results

Flipping the gif file: