Multimedia Programming 2004/2005

Assignments No. 4

January 12th 2005

Assignment 1 due: 11.00h, January 19th 2005

Goals of the assignments:

- Learn how to use classes in your program
- Learn how to overcome compiling problems
- Learn how to load, and play video files

Preparations:

1. Download the code from the MMP2004 web-site and unzip it to a local directory
2. All further directories mentioned in the assignments can be found in this local directory

Posting Your Work: See the last page of Assignment Set 1 for the procedure for posting your work. NB Only assignment 1 has to be posted.

Assignment 1: Classes and Pointers

Write a command line C++ program *list.exe* that can process the four commands *add, delete, find*, and *list* in the following way:

```
C:\list.exe
Welcome to the List Program!
Enter your command:
>add lisa
>add ashwin
>add peter
>add jane
>find gail
Sorry, gail is not in the list.
>list
People in the list are:
ashwin
lisa
jane
peter
>find ashwin
ashwin is in the list
>delete ashwin
>find ashwin
Sorry, ashwin is not in the list
>stop
Goodbye
C:\
```

You can assume that there will never be more than 100 names in the list. Furthermore the names will never be longer than 20 characters and will only contain characters from the set {a,...,z, A,..,Z}. The commands should do the following:

- The *add name* command adds name to the list.
- The *delete name* command deletes name from the list if it was in the list or does nothing if the name was not in the list.
- The *find name* command reports if the name is in the list or not
- The *list* command gives an alphabetically ordered of list all the names in the list.

Important: use classes for your implementation of the list data structure. Also, use a structured programming style:

- use clear identifiers for variable-, function-, and class-names
- use an easy readable consequent layout
- use the right control structures
- determine the right tasks for your (member-)functions

Assignment 2: Playing Video Files

1. Browse to the *PlayWnd* directory and double-click the Visual C++ workspace file *playwnd.dsw*
2. Browse through the code and read the comments to understand what the main functions do. Identify the main message-loop, the call back function that handles the main window messages.
3. Select <Build><Set Active Configuration…> and left click in the dialog on <PlayWnd-Win32 Release> and <OK>. This sets the configuration and hence the compiler settings that are used when building the executable.
4. Select <Build><Rebuild All> to build *PlayWnd.exe*. Execute by selecting <Build><Execute PlayWnd.exe>
5. Your program starts by creating the player window and opening an *Open Media File* dialog. Browse to the subdirectory *Media* that contains the video files for these assignments and open the file *butterfly.mpg*. Now this mpg should play.
6. By pressing the *right arrow* the playback ratio increases. Describe in detail what the program does (that is, which statements are executed) if the user presses the *right arrow* on the keyboard. For your description start at the right point in the *WinMain* procedure.
7. Close the player. Select <Project><Settings> and go to the <Debug> sheet of the dialog. Put *'..\Media\butterfly.mpg'* in the edit box <Program arguments> and run the program again. Now the program starts immediately playing the *butterfly.mpg* video as it is passed as argument of the *WinMain* procedure. Describe what is happening with the program argument after you selected <Build><Execute PlayWnd.exe>. (That is, only trace the video file.)

Assignment 3: Grab an Image from a Video File

1. Browse to the *Windowless* directory and double-click the workspace *windowless.dsw*
2. Select as active configuration *Windowless - Win32 Release*. There will be several errors while trying to build the executable. Change your project settings for this configuration such that it compiles and builds correctly. Describe what you had to do to solve these problems.
3. Execute the program and open the *chicken.wmv* video file. ***Important:*** if the content is offending to you, you may also use the *butterfly.mpg* video which has a less violent nature.
4. Change the code such that on a left mouse click the current frame is captured to a bitmap file (without overwriting earlier captured frames), and on a right mouse click the last captured frame is shown in the Windows Image and Fax Viewer (Note that this functionality is partly available through the menu of the program.)

Assignment 4: Playing Three Movies in a Cube

1. Browse to the *Cube* directory and double-click the workspace *Cube.dsw*
2. Build the executable Cube.exe and run the program. Note: if you encounter compilation errors try to solve them by adjusting the settings for your project.
3. Adjust the program such that on pressing the up arrow the rotating speed increases, while on pressing the down arrow the rotating speed decreases.
4. Furthermore, adjust the program such that by pressing the left arrow the direction in which the cube rotates is opposite to the default direction. If the right arrow is pressed the cube should rotate in the default direction.
5. Finally, the if one of the videos end playing the cube moves kind of strange. Adjust the program such that if one and the same movie is played on every face of the cube, the rotation continues smoothly, without returning to the initial position.

Assignment 5: Grab a Sequence of Bitmaps from a Video File

1. Browse to the *GrabBitmaps* directory, double-click the workspace *GrabBitmaps.dsw* and build the executable. Run the program. (Again: solve any building errors by changing the settings.)
2. The current settings of the program are such that every second an image is grabbed from a video file and saved as a bitmap image file. Change the program such that you can set the interval at which a frame is grabbed from the video file, and indicate the maximum number of frames that should be grabbed. That is, you should call the program from the command line as follows:

   GrabBitmaps.exe ..\Media\ruby.avi 25 10

   Where the first argument is the video file, the second argument is the time interval (in milliseconds) at which a frame is grabbed, and the last argument is the number of images that can maximally be grabbed from the video file.