UNIVERSITY OF SÃO PAULO

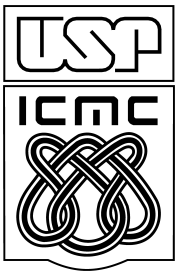
Institute of Mathematical Sciences and Computing

Screencast recording

Marco Aurélio Graciotto Silva
Ellen Francine Barbosa
José Carlos Maldonado

n. XXX

TECHNICAL REPORT



São Carlos - SP

Novembro/2011

UNIVERSITY OF SÃO PAULO

Institute of Mathematical Sciences and Computing

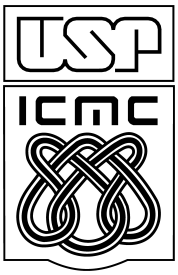
ISSN: 0103-2569

Screencast recording

Marco Aurélio Graciotto Silva
Ellen Francine Barbosa
José Carlos Maldonado

n. XXX

TECHNICAL REPORT



São Carlos - SP

Novembro/2011

Contents

Chapter 1

Introduction

A screencast is a movie-record of a running application (GANNOD; BURGE; HELMICK, 2008). Its main application is to demonstrate a specific functionality of a tool or just to showcase its main features. In the educational setting, screencasts are used to deliver lectures, usually through a subscription-based broadcasting of video (podcasts), meant to be watched before commencing the class (LAGE; PLATT; TREGLIA, 2001) or after it, to reinforce the learning activity.

An approach that uses screencasts before the class is the inverted classroom (LAGE; PLATT; TREGLIA, 2001). It consists of a teaching environment that mixes use of technology with hands-on activities: in-class lecture time is replaced with laboratory and in-class activities; outside-class time is used to watch lectures (GANNOD; BURGE; HELMICK, 2008). Thus, the student, aware of the subject and content the class will be about, can get involved in in-depth learning activities with other students and the teacher. Besides inverted classroom settings, screencasts can also be employed for for traditional ones, as an alternative to before-class assignments, replacing article readings. Actually, some studies characterizes most students as visual learners, and not as verbal learners (FOWLER *et al.*, 2000; THOMAS *et al.*, 2002), which favors the usage of videos.

The development of a screencast (at least a simple one) sounds trivial: one single person can prepare the content to be recorded, do the actual recording, and upload the movie to an podcast site. However, there are several issues in this process. First of all, a (recording) script must be written and rehearsed. Although reasonable videos can be produced without much training, good ones requires several tries and editing time. Actually, there are five phases for the development of a screencast:

1. **Planning** A screencast is not composed of just a plain video. Instructions, either in text or in audio format, should be displayed, describing the decisions done and the inner details of the activity handled by the screencast. Such interventions are necessary not only to clarify the functionalities of the tool, but also as an interaction with the viewers, keeping their attention in the main parts of the screencast.

Thus, the recording of screencast requires the careful definition of the execution script, balancing the content and the interactions with the viewer. This script should also define actions that must be taken before and while recording (such as computer configuration, and actions that are not shown in the video but are required to improve the screencast – such as mouse and window positioning).

2. **Recording** Recording consists on playing the script and recording the screen activity. It hardly is a one shot capture: several takes are usually required for a single screencast. Sometimes, it is required that several audio tracks be recorded (*i.e.*, for a screencast with multiple audio tracks for internationalization purposes). As it is unfeasible to perform a straight recording without committing mistakes, the person responsible for the actions should annotate the errors and cut scenes (moments where there will be a transition to another subject or location, therefore defining a significant change in the video flow). Such annotations will be used later for editing the video.
3. **Editing** The recorded video must be verified and corrected based upon the script and the annotations made while recording. Additional audio tracks must be mixed in. Cut scenes must be blended using suitable transition effects. If a blunt error is detected, it may be the case that a selection of the video needs to be recorded again.
4. **Encoding** The edited video has to be resized and compressed, as required by the target audience platform. For instance, if the screencast is aimed to mobile devices, the screen size should be set to 320x200; if it is aimed to digital television, 1920x1080 is the size of choice. The video also must be compressed: video encoding using MPEG-4 AVC (**??**) can drastically reduce file size (20 fold size reduction are common place), but the encoding process requires considerable computing power (thus taking a long time).
5. **Release** Finally, the screencast should be released in the Internet, either in sites such as YouTube or in podcast format.

This technical report provides a detailed description on how to record screencasts, from planning to delivery. Open source and free (free as in beer) software are used to support most activities in screencast recording. Instruction for both Windows and Linux are provided, as, often, screencasts must be recorded for both platforms.

1.1. Text organization

This technical report is organized as follows. Chapter 2 provides instructions on planning the screencast: how to write a script, evaluate, and test it. Chapter 3 describes the computer and operational system configurations required to record a screencast. Although the script provides several requirements regarding such configuration, technical issues related to screen size, operational system internationalization and localization, as well as minor details (such as installed software) must be handled. Chapter 4 addresses issues in the recording process, guided by script, as well as considering features such as audio recording, and auxiliary text. Chapter 5 suggests encoding programs and options that should be used for the screencast, providing excellent video and audio quality for the smallest possible file size. Finally, Chapter 6 provides some options for the distribution of the screencasts in the Internet.

1.2. Text conventions

The following conventions were adopted in this text: monospaced words represent system commands, short source code and text extracts.

Chapter 2

Planning

The first phase of a screencast is planning. This phase is responsible for the definition of the script to be executed, taking notice of learning requirements and outcomes, the content that will be recorded, and the required resources.

2.1. Script creation

The main elements of a script are the actions to be executed, and the moment (start time and duration). It is defined as a sequence of statements of the kind `<time> : <action>`

An action has a type, an identifier, and a sequence of parameters. There are three predefined types of action (the user can define as much types as required): setup, tool, and screencast actions.

- Setup action: A setup action is one required usually before running the application to be recorded: create a file, erase a directory, run the application once to set some default settings, etc.
- Tool action: A tool action is anything the user must do regarding interaction with the application being recorded. Such actions can be describe as high level statements (such as `save the file`) or low level statements (*e.g.*, `move the mouse to position (x,y), select the option Menu, then Save.`). Most actions with the computer should be defined using high-level statements, specially with videos that will be recorded once and are not subject to changes along time. However, if the video must be updated recurrently, it may be desirable to define every low-level computer interaction in such a way it can be used later to play the script automatically.
- Screencast action: An screencast should also interact with the user, explaining the tool actions, their rationale and outcomes. The commonest actions are:
  - Speak. Some text that must be spoken. Although it can be recorded after the screencast, it is better to record it while running the application that is being recorded (it is easier to speak within

the application context).

- Write. Some text that must be inserted. This text is usually highlighted and within a box. This box also should have some indication (like an arrow) that indicates the part of the application the text is about.
- Wait. Some actions can be recorded really fast. However, the learner probably needs some time to process the information. The wait/pause usually depends upon the amount of text to be read or listened.

For example, Section 2.1 provides a script about the use of a software testing tool (JaBUTi) to perform software measurement. It commences with two setup actions, required to prepare the tool for execution. As the timing is zero, they must be performed before starting the recording. The next three actions are ones that will be actually recorded: the subject will operate the tool as instructed. Finally, the subject must explain something relevant about the current screen of JaBUTi.

```
00:00 [Setup action] Open JaBUTi.

00:00 [Setup action] Prepare recording application.

00:00 [Tool action] Open test project.

00:03 [Tool action] Select Visualization/Complexity Metrics.

00:10 [Tool action] Slide metrics panel to the right.

00:11 [Screencast action] Explain the metrics shown in the dialog.
```

The script defined in Section 2.1 can be further improved. Indeed, when explaining the metrics, it may be wise to provide a text beforehand to be read. Even regarding the setup actions, decisions regarding the project to be open, and even the initial mouse position when recording should be considered.

2.2. Script testing

Script testing is the phase which tailors the script to the proper recording. It consists of a rehearsal of the recording, and it has the sole purpose of detecting errors before starting the recording itself. Possibly there will be adjustments to be done, and timing to be corrected.

As there is no strict requirement regarding quality, it can be performed using plain microphones (such as those embedded in notebooks). There is no need to capture the video, but it is required to observe the time taken to perform the actions and to read the text: such timing information should be compared against that defined into the script and, if necessary, corrected.

It is not possible to forecast all the problems that can arise during testing, but the usual pitfalls are:

- Application window size is greater than the captured area. Although it is easy to set up the capture are to the current window size, sometimes the application being recorded creates new windows or dialog boxes that are positioned out of the capture are.
  For instance, considering JaBUTi, this happens when creating a test project: the dialog shown to configure the project has a fixed size. If the recording started with a 800x600 capture area, this will be insufficient to capture the dialog box (that is wider than 800 pixels).
- The application, after loading, is in an unusual state (*e.g.*, for JaBUTi, the test requirements could

have already been covered, or no test case had been imported, etc). It important to keep a consistent configuration control of the application elements, such as the files required by user's action.
A recommended strategy is to have those files under strict configuration control, using a directory for each set of files required to run the application. If possible, use a version control system (such as Subversion (Apache Software Foundation, 2000) or GIT (TORVALDS; HAMANO, 2005) to store the required files.

Once more, let's consider Section 2.1. After testing the script, several errors were detected: timing, omitted actions, and so on. Section 2.2 provides the corrected script. As you can see, it is much more verbose and precise. The timing is quite different: from 13 seconds to a full minute. Finally, and quite important, it adds a missing and essential action: when to stop recording.

```
00:00 [Setup action] Open JaBUTi.

00:00 [Setup action] Move JaBUTi's window to the top left (position 0, 0).

00:00 [Setup action] Set JaBUTi's window height to 768 and width to 1024.

00:00 [Setup action] Create a test project for a Java application (e.g.,
                     VendingMachine) or open a test project already defined.

00:00 [Setup action] Close the project.

00:00 [Setup action] Prepare the recording application.

00:00 [Screencast action] Explain the purpose of the screencast (software
measurement so that an adequate software testing strategy can be defined).

00:15 [Tool action] Open test project.

00:17 [Screencast action] Provide a brief description of the application that
will be used as example.

00:25 [Tool action] Select Visualization / Complexity Metrics.

00:26 [Tool action] Slide metrics panel to the right.

00:26 [Screencast action] While sliding the panel to the right, explain how
many metrics JaBUTi implements.

00:40 [Screencast action] Explain the results (the metrics) and how they could
be used to define a software testing strategy.

00:60 [Setup action] Stop.
```

It is hard to estimate how much time it is required to test a script, as it depends on the application to be recorded, on the subject experience and so on. Nonetheless, it is safe to state that at least half an hour is required for any recording (regardless of its size).

Chapter 3

Screencast recording

The recording of a screencast – a movie record of a running application – requires the careful execution of the recording script to avoid an excessive editing effort. This chapters describes the main activities and procedures to properly execute a screencast recording.

3.1. Requirements and baseline configuration

Performing a screencast recording is better accomplished using a virtual machine. Although the usage of a physical machine is feasible, sometimes it is not possible to capture screenshots of screens that avoid the usual operational system rendering engine (*e.g.*, hardware-accelerated 3D applications). On the other hand, a virtual machine screen can always be recorded by the host computer, no matter the rendering mechanism employed by the guest operational system.

The use of a virtual machine also has other advantages. Usually, the recording requires several system-wide configurations that are harder do apply to a production system or that could compromise the security of computer applications. For instance, the recording is often done in one unique language and locale, which can be different from the native one. The same requirement applies for installed applications, desktop icons, etc. Finally, a virtual machine can be configured with an operational system different than the host's one. Thus, a Linux user can install Windows as a guest operational system and record the screencasts (and vice-versa).

The virtual machine of choice for recording is VirtualBox (Oracle Corporation, 2007). It is supported by major operational systems (Windows, Linux, MacOS) and is open source, *i.e.*, it is always possible to hack the virtual machine application and fix important issues (but that is not generally the case).

Besides the virtual machine application, it is required the operational systems themselves. It is thoroughly recommended to have, at least, one Windows and one Linux virtual machines. As for the virtual

machine configuration, the following configurations are recommended:

- Disable shared clipboard (menu item `General / Advanced / Shared Clipboard`). This way the clipboard of host and guest operational systems are kept apart (if required, you can enable it for a specific screencast later).
- Set the base memory to, at least 512 MiB (menu item `System / Motherboard / Base Memory`). Consider setting it to 1 GiB if enough memory is available in the host system.
- Enable PAE/NX (`System / Processor / Extended Features / Enable PAE`). This option enables the usage of larger memory pages and execution protection technologies by the guest operational system.
- Enable VT-x/AMD-V (`System / Acceleration / Hardware Virtualization`). Recent processors (AMD Phenon, Intel Core and so one) provides instructions that supports the execution of virtual machines without emulation of some hardware features (mostly related to memory management and I/O instructions).
- Enable nested paging (`System / Acceleration / Hardware Virtualization`). Besides some basic virtual-machine acceleration instructions, some CPUs supports nested paging, which further improves the performance of guest systems.
- Set video memory to 16 MiB (`Display / Video / Video Memory`).
- Enable 3D acceleration if the application to be recorded requires uses DirectX or OpenGL APIs (`Display / Video / Extended Features`). If enabled, the video memory (previous configuration) must be increased as instructed by VirtualBox.
- Disable display server (`Display / Remote Display / Enable Server`).
- Set the IDE Controller to ICH6 and use host I/O cache (`Storage`).
- Set the harddisk image size to 8 GiB or more (as required by your recording requirements).
- Enable audio support (`Audio / Enable Audio`).
- Enable network adapter (`Network / Adapter 1 / Enable Network Adapter`) and set it as attached to NAT.
- Enable USB controller (`USB / Enable USB Controller`) and USB 2.0 support (`USB / Enable USB Controller / Enable USB 2.0 Controller`). The later (USB 2.0) requires the installation of a proprietary extension pack (available at `https://www.virtualbox.org/wiki/Downloads`). If you are sure you will not need USB 2.0 support in your guest operational system, you can ignore this option.
- Set up a shared folder (usually the home directory). This is useful to store files required by the application to be recorded.

Both virtual machines (Windows, and Linux) have to be carefully configured. The following configurations are strongly recommended:

1. Operational system installation, default language and locale must be set to English (unless the recording is aimed for a foreign language audience).
   Although trivial, keep in mind that this configuration, if different from the current host machine one, can trigger errors in the application to be recorded (that was the case of JaBUTi, that has to convert numbers generated by another application, which depends on the system locale – Brazil uses a comma as decimal separator while English uses a dot).
2. Configure a trivial primary user account, with a generic username and password (*e.g.*, username `user` and password `user`). If possible, disable password at all.
3. Set the default user's permissions to Administrator level. Security is not an issue for a virtual machine

which unique purpose is to record screencasts, and this configuration will avoid errors related to denied permissions.

4. The virtual machine guest drivers must be installed as soon as possible. It is a firm required to have them installed, as the guest operational system performance depends on them.
5. Every possible security patch and service pack must be installed manually. Any automatic update mechanism must be disabled. This will avoid any unwanted window from appearing meanwhile recording.
6. Install the barely minimum applications necessary, *i.e.*, nothing but the operational system, the recording application, and the application to be recorded. There is no need to install antivirus software (the virtual machine will be isolated from the host machine most of the time, and its state can always be rolled back to a stable one, *i.e.*, without viruses).
7. Disable any automatically run or unwanted application. For Windows, the `msconfig` application can be used. It is thoroughly suggested to disable the following start up items:
   - VirtualBox tray icon (`VBoxTray`).
   - Adobe Acrobat Reader accelerator (`Reader_sl`).
   - Adobe Reader Updater (`AdobeARM`).
   - Microsoft text input service support (`ctfmon`).

   For Windows, the user should also disable the system restore feature (the user can use the rollback feature of the virtual machine application later, if necessary).
   For Linux, disable any service that is not required by the system (Apache, MySQL, Bluetooth, etc).
8. Remove any unwanted icon from the shell menu and from the desktop. Keeping the menus and desktop as clean as possible will focus the user attention on the application being recorded (and, for instance, will even improve the compression ratio of screenshots and movies).
9. Configure the operational system and applications to remove any user history and recently used document or application at start up or log out. The virtual machine must have, after every restart, the very same configuration. This will assure consistency throughout the recordings.
10. It is recommended to disable fancy window manager decoration and effects (such as sliding menus, shadowed mouse cursor). Actually, it is recommended to disable everything but text antialiasing settings (that, in fact, you must enable and configure it properly).

For screencast recording, at least two applications must be installed: one for resizing windows, and other for recording itself. Windows resizing, although trivial, is rather important: data compression efficiency depends heavily on picture size: it must usually be a multiple of 8 or 16, and have the smallest possible dimension. Therefore, the recommended applications to resize application's windows are:

- Window resizing:
  - Linux: wmctrl (STYBLO, 2003). It is a command line application. Its usage is rather trick: first you have to discover the window id (using the command `wmctrl -l`) and set the desired geometry (x-offset, y-offset, width, and height) using the command `wmctrl -i -r <window id> -e 0,<x-offset>,<y-offset.,<width>, <height>`. Example:

```
$ wmctrl -l
0x01c0001e -1 labes-desktop-010808 x-nautilus-desktop
0x01600085  0 labes-desktop-010808 404 Not Found - Mozilla Firefox
0x03a0003d  0                  N/A JabRef
0x03000112  0 labes-desktop-010808 Oracle VM VirtualBox

$ wmctrl -i -r 0x03a0003d -e 0,0,0,800,600
```

  - Windows: Sizer (Brian Apps, 1997). This is a graphical application. You should configure it to disable the tray icon, and to run it at startup (or at your discretion, manually running it). It will add an option in the window's menu option (usually the icon at the top left corner of the window): select it to change the window size.
- Screencast recording:
  - Linux: xvidcap (BECKERS *et al.*, 2003). It is a graphical application that records movies of an specific region of the desktop. It can save either to several files (as single images) or to a movie.
  - Linux and Windows: wink (KUMAR, 2009). This is application is just perfect for screencast that do not have to capture highly interactive or rich (fast pacing/movement and color rich) applications. It not only captures screenshots, but can be used to edit screencasts, add subtitles, images, and so on. It is recommended to also install CamStudio (SMITH; MCQUADE *et al.*, 2005).

Finally, it is highly recommended to remove unneeded files from the system and, if possible, defragment the filesystem from both guest and host computers. You should also use the system for a while and fine tune it. Then, create a snapshot (select the virtual machine, select the `Snapshots` tab, and click on the first button in the toolbar). Name this snapshot `Baseline`. This way, you call always rollback to this snapshot before performing a new recording.

3.2. Recording

After creating and testing the script, and configuring the virtual machine, the actual recording is straightforward. You only need to configure some settings regarding the video capture as follows:

- Capture at no more than 15 frames per second (fps). Actually, it is often the case that it is only required to capture the screen after an event (such as a mouse click). However, if your application change its state automatically, a timed capture can be useful: in this case: 15 fps is a fine choice.
- If saving to a video file (AVI), set the key frames to the same value as the frame rate (for example: if using 15 fps, set a key frame every 15 frames).
- Capture a screen at every mouse and key press. If you are screen capture is driven by interaction events (key and mouse presses), that is done automatically. However, if you are using time-based capture, at a fixed frame rate, keep in mind you need to perform such interactions in a slower speed (preferably stopping any movement for half to one second after each major interaction), so that it can be captured clearly on the video.
- Configure shortcut keys to control the recording software (to capture a screenshot, to start and stop time capture, to start and stop input-driven capture). Otherwise, such control interactions may appear on the video, requiring further editing.
- Set the area to be captured to default screen resolutions: 320x240, 640x480, 800x600, 1024x768, 1280x720, 1366x768, 1920x1080. You must keep in mind that the smaller the required area size, the better). Special care must be taken regarding the screen proportion: standard (4:3) or widescreen (16:9). A setting of 800x600 (4:3) or 1280x720 (16:9) is usually enough.
- Save the video and audio to an uncompressed stream or use a lossless encoding (usually Huffman-based codecs are lossless).

The recommended application to capture a screencast is Wink (KUMAR, 2009). It captures screens on time basis (*e.g.*, every three seconds) or after every mouse or keyboard event, which is convenient to capture plain interaction with applications. However, if the goal is to capture applications that produce high definition graphical output (such as movies or games), it is recommended to use applications such as Jing (TechSmith, 2011b) or CamStudio (SMITH; MCQUADE *et al.*, 2005) (for Windows) or Xvidcap (BECKERS *et al.*, 2003) (for Linux).

3.2.1. Wink

If using Wink (KUMAR, 2009) to record the screencast, some extra properties must be configured. The output file type must be set to `Macromedia Flash` (that is the default value). At the menu item `Project / Settings`, the following options must be unchecked: `Add preloader to flash output` and `Add control bar to flash output`. The frame rate can be kept at a low value if the screencast will capture only plain windows (the recommended value is three frames per second). However, if there are many interactions (such as keypresses and mouse events) that, as configured in the screencast, triggers the capture of screenshots, the frame rate should be set to a higher value (15 or 30 frames per second). As for the cursor movement, it should be set as shown in Figure 3.1.

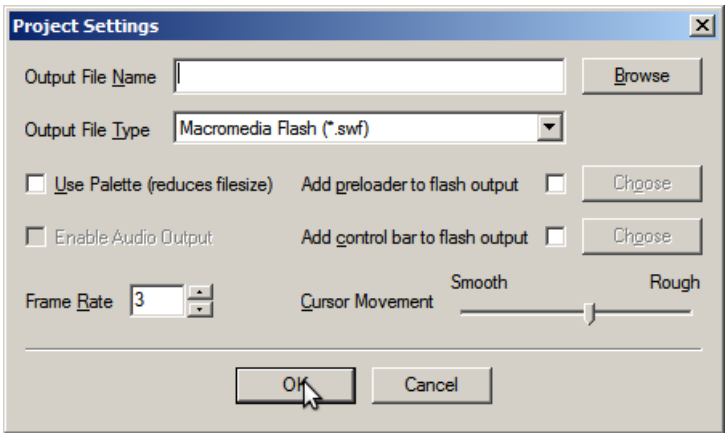


Figure 3.1: Recommended configuration for a Wink project.

It must be remembered that the recording will be later edited. So, instead of stopping a record and start it over again, it is possible (and preferable) to just repeat the wrong scene. The only care it should be taken is to write down the moment the scene is cut (this is specially important if the person that edit the recording is different of the one that recorded it).

After recording the screencast, it is necessary to covert the SWF to an AVI file. The software swf2avi (TEAM, 2004) is recommended to perform such conversion: it produces an uncompressed AVI file (which, after edition, can be compressed). For each file to be converted, it is possible to define conversion configuration/profile. It is strongly recommended to use the same settings as used while capturing the video for frames per second and screen size.

3.2.2. Xvidcap

Xvidcap is available on most Linux distributions, thus not requiring any special installation instruction. However, stock Xvidcap cannot record using lossless codecs. As its development halted some years ago, we

have forked it, rewriting most of the code and adding support for newer codecs. The source code of this newer Xvidcap is available at `https://www.ironiacorp.com/svn/projects/xvidcap/trunk` and, soon, compiled packages will be available at `http://www.ironiacorp.com/projects/xvidcap`.

3.2.3. Jing

Jing is a free and simple screencast recorder from TechSmith (2011b). It is a reduced version of Camtasia, a full fledged commercial recorder (TechSmith, 2011a). The most important limitation is that at most five minutes can be recorded using Jing. Besides that, it does not support editing and can only save the output as Flash (SWF) files.

3.2.4. CamStudio

CamStudio is an open source software for screencast recording (SMITH; MCQUADE *et al.*, 2005). Originally it was developed by RenderSoft, which was later acquired by Adobe. Afterwards, it license changed to a non-open one and its development ceased. Fortunately, a group of developers continued the project using the latest open source code available.

The installation is not as straightforward as Wink or Jing. The latest version of CamStudio (2.6) is released as 7Z archive, requiring the user to manually unpack it using a software like 7-Zip (PAVLOV, 1999). It also depends upon Microsoft Visual C++ redistributable libraries (available at `http://www.microsoft.com/download/en/details.aspx?id=5555`).

As explained before, screencasts should be recorded, whenever is possible, using lossless compression. CamStudio supports any codec installed in Windows. You can choose one for recording at `Options / Video options`. The recommended options are:

- Lagarith: Fastest codec and good compression rate. URL: `http://lags.leetcode.net/codec.html`
- Huffyuv: Fast codec. URL: `http://neuron2.net/www.math.berkley.edu/benrg/huffyuv.html`
- Huffyuv_mt: Fast codec, based upon Huffyuv, but with multithreading support (thus faster than Huffyuv on multi-core systems). URL: `www29.atwiki.jp/lossless/pages/11.html`
- CamStudio codec: CamStudio's own codec. URL: `http://camstudio.org`

Lagarith and Huffyuv (and its counterpart, Huffyuv_mt) are supported by most video editing applications. If using a multi-core machine, Huffyuv_mt is probably the best choice. Otherwise, the wisest choice is Lagarith.

Regarding frame rate and key frames, you should slide the time lapse setting on the bottom of the video option configuration window, until the desirable values are set for key frames and playback rate (as shown in Figure 3.2. If it is not possible to configure this, uncheck `Auto Adjust` box and set the values directly.

3.3. Concluding remarks

Although recording is supported on any platform, performing recordings on Windows is easier: better tools, rightly available compression libraries. On Linux, Xvidcap requires the use of unofficial version of the software to recording using lossless codecs; Wink for Linux is at version 1.5, while for Windows it is at

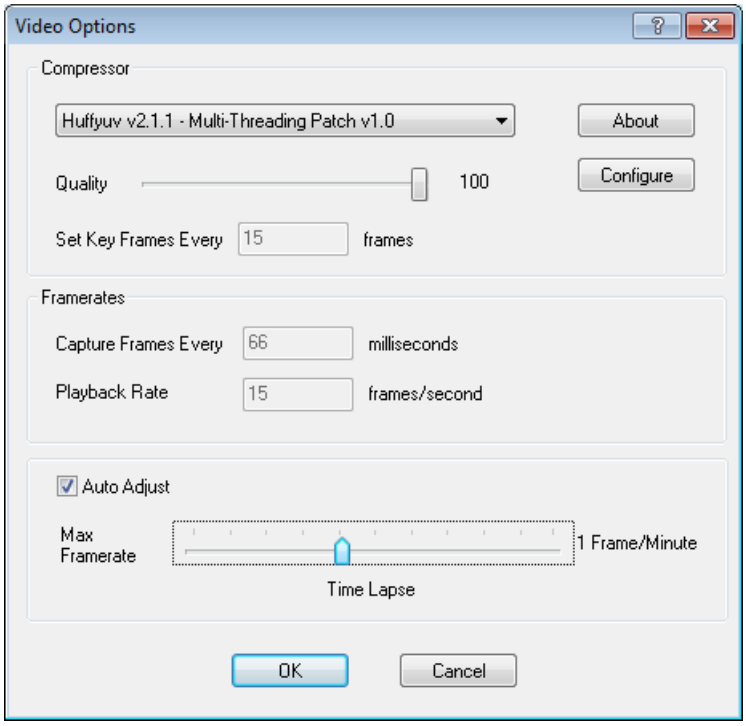


Figure 3.2: Video options for CamStudio.

2.0 (and actively developed). Thus, the use of a virtual machine with Windows to record applications is, sometimes, the best choice.

Chapter 4

Screencast edition

The screencast edition requires the script used for recording and notes done while recording. It is the script that will specify the elements to be added to the video (subtitles, boxes). The notes are useful for cutting the right scenes.

When adding elements to the screencast, it is important to consider the time to be spent per screenshot and for each element added to it. For instance, if a box, explaining the current screenshot, is added, it must be added an extra time, so that the viewer can read all the data and understand it. As a rule of thumb, consider the following guidelines:

- For text in a box, one second per line is a good amount of time. Keep the time always at a minimum of 3 seconds per screen with text.
- Consider adding a 0.3 to 0.5 seconds delay per screenshot. This is specially recommended if the final recording will be rendered at a high frame rate (*e.g.*, 30 frames/s).

4.1. Filters

The application of filters to a video is often required, either to remove an undesired frame or to optimize the video quality. As screencasts have, by definition, no noise, just filters related to cropping are discussed here.

Cropping is required when the video captures more area than the application of interest (although, at recording, such area should haven been correctly defined). Crop can be applied using several software: MPlayer (GEREöFFY *et al.*, 2000), FFmpeg (BELLARD; NIEDERMAYER *et al.*, 2000), etc. For instance, using MPlayer to crop a video to 1024x814, ignoring the top of the window (116 pixels of height along all the windows's width), the command to be used is as indicated in Section 4.1: the filter name is `crop`; the arguments are the horizontal size, the vertical size, the horizontal offset and vertical offset (thus defining the

position of the top-left pixel of the video).

```
mplayer -vo x11 -vf crop=1024:814:0:116 test.avi
```

MPlayer can show the result of the crop operation, but it does not record it. That is addressed my MEncoder (which is provided by MPlayer). The command line is similar Section 4.1. Besides the cropping, you may have to tune the compression options (parameters `lavc` and `lavcops`) as described in Chapter 5.

```
mencoder -vf crop=1024:814:0:116 test.avi -oac copy -ovc lavc --lavcopts ffv1 -o test2.avi
```

Chapter 5

Screencast encoding

A screencast must be encoded using a proper codec, supported by the target platform. It is often useful to generate screenshots (*e.g.*, one for every ten seconds of video) for previewing purpose. This chapter describe how to encode the video and audio of a screencast, as well as how to generate screenshots of the video.

5.1. Video encoding

Video encoding is the most computing intensive activity of the entire process: uncompressed movies have high demand storage requirements (disk space and I/O performance) that surpass audio and container by a large scale. For instance, a 1920x1080 video (Full HD resolution) of 16 bits per pixel (a conservative YUV encoding) at 25 frames per second requires 103,680,000 bytes per second (roughly 99 MiB/s). A typical movie (with a length of two hours) would require 696 GiB. The audio for this movie, at 48000 Hz, 16 bits per sample, 5 channels, would require 480.000 bytes per second (0.45 MiB/s), which accounts for 3,22 GiB. Today, all this data (almost 700 GiB) can be squeezed to a single Bluray disk (*i.e.*, 50 GiB). Video is responsible for the greatest reduction: a compression ratio of 10:1 is usually achievable using MPEG-4 Part 10.

Video compression tools are readily available. For compression, MPEG-4 Part 10 (h.264) will be employed, which is supported by x264 (AIMAR *et al.*, 2004) (an open source h.264 video encoder). Another option for compression is FFmpeg (BELLARD; NIEDERMAYER *et al.*, 2000), which can create videos compliant to h.264 as well other formats such as FFV1, a lossless video encoder) and apply filters to the video content (which is quite common when encoding a video).

A screencast is the result of the capture of the computer screen. It is mostly used to demonstrate the functionalities of a program. Most compression techniques exploit the limitations on human vision to improve the compression ratio (chroma sub-sampling, transformations, quantizations, entropy coding,

motion compensation, and so forth). However, for screencasts, some techniques renders a low quality output (*i.e.*, quantization, which discard frame information). Thus, for screencasts, it is recommended that some of these techniques be disabled. Fortunately, as screencasts are mostly sequences of figures which unique differences are only the mouse pointer or the cursor location, disabling lossy techniques still provides a significant reduction into the video size. Currently, a complete lossless compression cannot be achievable using h.264 (at least for x.264). Although some lossy compression techniques can be disabled (such as quantization), chroma sub-sampling and other color conversions must take places. A screencast is recorded using 4:4:4 (RBGA) sub-sampling while x264 supports only 4:2:0 (YV12). Fortunately, this should not be an issue.

Several parameters must be configured to encode a video. Next, we describe the main options and how they can be configured using x264 to produce good screencasts, settings which were based on documentation distributed with the software itself and by video compression specialists (Computer Graphics and Multimedia Laboratory, 2008). Latter, we define how such options can be configured for FFmpeg[1].

- **Disable quantization.** Quantization is responsible for discarding information of a image, preserving information which the human eye is more sensitive to. This must be disabled, using the options `--qp 0 --qpmin 0 --qpmax 0`. It is also worth to disable trellis quantization (`--trellis 0`).
  Besides disabling quantization, a profile (`--profile`) must **not** be set as it will disable the lossless compression option.
- **Key frame interval.** For a screencast, it is desirable that the video can be positioned (forwarded and reversed) at discrete and meaningful moments. As keyframes are encoded as I-frames (frames which rendering does not depend on any other frame), the recommended setting is to set the key frame interval to the minimum seeking interval desired for the screencast. As we do not know, in advance, such moments, the smaller the interval between key frames, the better. One second is a good compromise between indexing capability and compressibility. The recommended value is **`--keyint <frame rate>`**. (*e.g.*, `--keyint 15`). It also useful to define the option **`--no-scenecut or --scenecut 0`**, as the key frame interval already is sufficiently low (and extra key frames would decrease the compression ratio).
  If the screencast is not aimed to streaming, the compression ratio can be improved using the option `--intra-refresh`.
- **Reference frames.** Motion estimation is done using frames either before or after the current frame. The more reference frames, the greater the opportunities to define good motion vectors (and improve the compression rate). Set this to 16 (**`-refs 16`**).
- **Streamming optimization.** If the screencast will be streamed, B-frames should be avoided (as they depend on latter frames to be rendered, which is an issue for streams).
  The usage of B-frames is controlled by the options `bframes`, `b-adapt` and `b-bias`. Setting them to 0 disables B-frames usage (**`--bframes 0 --b-adapt 0 --b-bias 0`**).
  If using screencast will not be streamed, B-frames should be enabled. Then, the recommended settings are **`--bframes <n> --b-adapt 2 --b-pyramid=normal --weightp 2`**, where `<n>` is the same value defined as reference frames (`-refs` option).
- **Group of pictures.** Open group of pictures allow a higher compression, thus always leave then open (`--open-gop normal`).
- **Psychovisual optimizations.** Psychovisual optimization should be disabled (`--no-psy`). As no quantization is applied to the video, psychovisual optimizations have no effect on the output quality or compression ratio.

[1] We chose to describe x264 options first because FFmpeg ones are just mappings to x264 options (and mapping often incomplete and, sometimes, even incorrect).

- **Prediction mode for motion vector.** Set it to `auto` (`--direct auto`).
- **Motion vector search range.** Maximum range for motion vector search. The minimum is 16, but the greater, the better is the compression ratio. The recommended value is `--merange 511.75`.
- **Motion estimation method.** Method used to estimate the motion (`--me tesa`). Use either `umh` (uneven multi-hexagon search – faster, but non-optimal) or `tesa` (hadamard exhaustive search – slower, but optimal).
- **Subpixel motion estimation method.** Method used for subpixel motion estimation and mode decision (`--subme`). Recommended value range is between 6 and 9.
- **Partitioning.** Partitions to consider (`--partitions`). Recommended value is `all`. However, if a higher encoding speed is required, this option can be set to `none` without significant compression ratio losses.
- **Miscellaneous options**.
  - `--no-fast-pskip`: Disables early SKIP detection on P-frames.
  - `--no-dct-decimate`: Disables coefficient thresholding on P-frames.
  - `--no-deblock`: Disables loop filter deblocker.
  - `--fullrange=on`: Control the full range of luma and chroma levels.
  - `--seek=<n>`: Specifies the first frame to encode.

The encoding options can be set either in the command line or using a preset. The later option will be used. A preset is file, usually hosted at `~/.ffmpeg`. The file's name must have the prefix `libx264-` and `.ffpreset` as prefix. The option `vpre` (for video preset) sets the preset o be used. For instance, Section 5.1 compress a file using a lossless preset (`~/.ffmpeg/libx264-screencast.ffpreset`), specially tuned for screencasts.

```
# ffmpeg \
        -i <input file> \
        -vcodec libx264 \
        -vpre screencast \
        <output file>
```

For screencasts that consists mainly of usual desktop activity, lossless compression is recommended. The MPEG-4 Standard, defines a profile for lossless compression: High 4:4:4 Predictive Profile (Hi444PP). It builds on top of the High 4:2:2 Profile, supporting up to 4:4:4 chroma sampling, up to 14 bits per sample, and additionally supporting efficient lossless region coding and the coding of each picture as three separate color planes (**??**).

It has been reported, for x264, that B-frames should be disabled, as (1) they allow nonuniform allocation of quality, but not if all frames have to be lossless; and (2) they allow nonuniform residual[2]That sounds odd and, actually, "it costs fewer bits to code a large DCT coefficient in a P-frame and a zero in a B-frame, than half the magnitude in 2 P-frames. But prediction is rarely good enough to leave a whole non-quantized block with no residual, so lossless still has to code coefficients in the B-frames". Thus, B-frames are disabled when using lossless compression. Other options that are disabled or have no effect for lossless compression are: deblocking, quantization (quantization matrices), psychovisual optimization (dead zone, trellis), and rate control. Hence, the recommended options to encode movies losslessly are defined in Section 5.1. If such settings do no work, uncomment the lines starting with the symbol `#` (they configure `qmin` and `qmax`, which have no meaning for lossless compression, but are required to some odd FFmpeg behavior).

[2] This information about x264, lossless compression and B-frames was obtained from a forum that discusses video encoding using H.264 (`http://forum.doom9.org/archive/index.php/t-129977.html`).

```
coder=1
partitions=+parti8x8+parti4x4+partp8x8+partp4x4+partb8x8
me_method=full
me_range=512
subq=10
directpred=3
g=250
keyint_min=3
refs=16
wpredp=2
# qmin=0
# qmax=69
cqp=0
flags=-loop+mv4+obmc+qpel+gmc+mv0+part+aic+umv+cbp+qprd
flags2=+mixed_refs+dct8x8-fastpskip+bpyramid+wpred+mbtree-psy
```

For lossy compression, the following presets provided as default by FFmpeg are recommended: `main`, `medium`. Additionally, the option `-crf 20` is recommended.

Although x264 (and the implemented h.264 codec) is highly efficient, it is not properly lossless if it is required a colour conversion (*e.g.*, from RGBA to YUV). If there is a requirement to save a completely lossless version of the video, use the `ffv1` codec. This codec supports all available colour spaces and provides one of the best compression ratios among others lossless codecs (Computer Graphics and Multimedia Laboratory, 2007). It does not provide many options to tune (in contrast to h.264 related codecs). Actually, just two are useful for archival purpose: `coder` and `context`. Choose either arithmetic coding (`ac`) or huffman (`vlc`): the former produces smaller files, the latter is faster. As for `context`, if set to `1`, a large context (thus higher compression ratio) is defined; if set to `0`, a small context will be set (providing lower compression ratio, but higher performance).

```
# ffmpeg \
        -i <input file>
        -vcodec ffv1 \
        -coder ac \
        -context 1 \
        <output file.avi>
```

The `context` option, for current FFmpeg (as of December 2010), is broken. So, if the command fails, try to remove this option and run it again.

5.2. Audio encoding

If the screencast has audio, the best choice for audio encoding is MP3 at constant rate. A bit rate of 64 kbps is reasonable for voice recording. If music or another complex sound is required, 128 or greater bitrates should be used.

5.3. Snapshots creation

Snapshots can be create using Mplayer (GEREöFFY *et al.*, 2000). The rationale is rather simple: instead of playing the screencast to the screen, it is written to files (as PNG images). The option that outputs PNG files is `-vo png`. It supports the following options:

- `z`: Compression level (0 is no compression, 9 is maximum compression). As we will recompress the PNG later, a setting of 6 is a good compromise between speed and compression ratio.
- `outdir`: Output directory. The default value is the current directory.
- `outfile_prefix`: Output filename prefix. The final filename will be in the format `prefix00000000` (eight digits with leading zeros). The default value for `outfile_prefix` is an empty string.
- `alpha`: Create PNG files with alpha channel. This is not required.

Considering a typical screencast, recorded at 3 frames/second, the command line would be `mplayer -vo png:z=6:outfile_prefix=test`. The files will be created in the current directory and they will be named as `test` suffixed by a sequential number (starting at 1) and the extension `.png` (*e.g.*, `test00000001.png`, `test00000002.png`, etc).

The default MPlayer configuration will create one file for every frame of the video. To define the sampling frequency, the option `-sstep` should be used. It has just one parameter: the amount of seconds to skip after each output frame. The recommended value for this is a multiple of the keyframe rate of the screencast (MPlayer can only seek to keyframes, so, if the frame after N seconds is not a keyframe, the next keyframe will be used instead). Considering a screencast recorded at 3 frames/second, to generate screenshots for every 30 seconds, the command line would be `mplayer -vo png:z=6:outfile_prefix=test -sstep 30`. A better option would be to create an screenshot if the different between the frames is significant, but this is not currently supported by Mplayer.

Finally, it is recommended to recompress the generated files. The recommended software to optimize PNG files are Pngrewrite (SUMMERS, 2003) and AdvanceCOMP (MAZZOLENI, 2002). The former tries to reduce the color space and palette of the image, and the later recompress the file, using an improved deflate algorithm (TRUTA, 2008). The command at 5.3 will recompress all PNG files in the directory defined as the first argument:

```
#!/bin/bash
TEMP_FILE=`mktemp -u`
for i in `find $1 -iname "*.png" -print0`;
do
    pngrewrite $i $TEMP_FILE;
    if [ $? -eq 0]; then
        mv $TEMP_FILE $i;
    fi;
    advpng -z -4 $i;
done
```

Another option is to create JPEG-based snapshots. Just replace `png` option for `jpg` in Mplayer command line. In order to optimize the generate images (as we did with PNG ones), use jpegtran (a part of the libjpeg software) with the options `-optimize -progressive` to losslessly optimize JPEG images. The option `-optimize` enables the use of dynamic Huffman tables (using arithmetic coding for the entropy coder would make images even more compact, but it is patent ridden and isn't widely used or supported); `-progressive` surprisingly makes many JPEG images smaller, and also is good for use on the web with large images as it lets them display progressively in the browser.

Chapter 6

Release

Nowadays, the best option to release a screencast is to upload it to YouTube (`http://www.youtube.com`). It supports several resolutions and is indexed by most search engines, which increases the visibility of the screencasts. Always upload the MPEG-4 AVC (h.264) version of the video, as it is often smaller.

An alternative to YouTube is Vimeo (`http://vimeo.com/`). It is aimed at higher quality video.

References

AIMAR, Laurent; MERRITT, Loren; PETIT, Eric; CHEN, Min; CLAY, Justin; RULLGåRD, Måns; CZYZ, Radek; HEINE, Christian; IZVORSKI, Alex; WRIGHT, Alex; GARRETT-GLASER, Jason; OTHERS. *x264*. 2004. Computer application. Disponível em: `http://www.videolan.org/developers/x264.html`.

Apache Software Foundation. *Subversion*. jun. 2000. Computer application. Disponível em: `http://subversion.apache.org/`.

BECKERS, Karl H.; OTHERS. *xvidcap*. set. 2003. Computer application. Disponível em: `http://sourceforge.net/projects/xvidcap/`.

BELLARD, Fabrice; NIEDERMAYER, Michael; OTHERS. *FFmpeg*. 2000. Computer application. Disponível em: `http://www.ffmpeg.org/`.

Brian Apps. *Sizer*. fev. 1997. Computer application. Disponível em: `http://www.brianapps.net/sizer/`.

Computer Graphics and Multimedia Laboratory. *Lossless Video Codecs Comparison '2007*. Russia, mar. 2007. Disponível em: `http://compression.ru/video/codec_comparison/lossless_codecs_2007_en.html`.

Computer Graphics and Multimedia Laboratory. *Options Analysis of MPEG-4 AVC/H.264 Codec x264*. Russia, dez. 2008. Disponível em: `http://compression.ru/video/codec_comparison/x264_options_analysis_08_en.html`.

FOWLER, Lynne; ALLEN, Maurice; ARMAREGO, Jocelyn; MACKENZIE, Judith. Learning styles and CASE tools in Software Engineering. In: HERMANN, A.; KULSKI, M. M. (Ed.). *Annual Teaching Learning Forum*. [s.n.], 2000. Disponível em: `http://otl.curtin.edu.au/tlf/tlf2000/fowler.html`.

GANNOD, Gerald C.; BURGE, Janet E.; HELMICK, Michael T. Using the inverted classroom to teach software engineering. In: *International Conference on Software Engineering*. New York, NY, USA: ACM, 2008. p. 777–786. ISBN 978-1-60558-079-1.

GEREöFFY Árpád; OTHERS. *MPlayer*. nov. 2000. Computer application. Disponível em: `http://www.mplayerhq.hu`.

KUMAR, Satish. *Wink*. 2009. Computer application. Disponível em: `http://www.debugmode.com/wink/`.

LAGE, Maureen J.; PLATT, Glenn J.; TREGLIA, Michael. Inverting the classroom: A gateway to creating an inclusive learning environment. *Journal of Economic Education*, v. 31, n. 1, p. 30–43, 2001. Disponível em: `http://econpapers.repec.org/RePEc:jee:journl:v:31:y:2000:i:1:p:30-43`.

MAZZOLENI, Andrea. *AdvanceCOMP*. maio 2002. Computer application. Disponível em: `http://advancemame.sourceforge.net/comp-readme.html`.

Oracle Corporation. *VirtualBox*. 2007. Computer application. Initially developed by Innotek, which was acquired by Sun Microsystem, which was acquired by Oracle Corporation. Disponível em: http://www.virtualbox.org/.

PAVLOV, Igor. *7-Zip*. jul. 1999. Computer application. Disponível em: http://7-zip.org.

SMITH, Nick; MCQUADE, Paul; OTHERS. *CamStudio*. 2005. Computer application. Disponível em: http://camstudio.org/, http://sourceforge.net/projects/camstudio/.

STYBLO, Tomas. *wmctrl*. set. 2003. Computer application. Disponível em: http://tomas.styblo.name/wmctrl/.

SUMMERS, Jason. *Pngrewrite*. fev. 2003. Computer application. Disponível em: http://www.pobox.com/~jason1/pngrewrite/.

TEAM swf2avi. *swf2avi*. 2004. Computer application. Disponível em: http://www.avi-swf-convert.com.

TechSmith. *Camtasia*. 2011. Computer application. Disponível em: http://www.techsmith.com/camtasia.html.

TechSmith. *Jing*. 2011. Computer application. Disponível em: http://www.techsmith.com/jing.html.

THOMAS, Lynda; RATCLIFFE, Mark; WOODBURY, John; JARMAN, Emma. Learning styles and performance in the introductory programming sequence. In: *SIGCSE Technical Symposium on Computer Science Education*. [S.l.]: ACM, 2002. p. 33–42.

TORVALDS, Linus; HAMANO, Junio C. *GIT*. abr. 2005. Computer application. Disponível em: http://git.or.cz/.

TRUTA, Cosmin. *A guide to PNG optimization*. maio 2008. White paper. Disponível em: http://optipng.sourceforge.net/pngtech/optipng.html.