TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Silver Valdvee 179390IADB

ADDING ZOOMING PRESENTATION CAPABILITY TO LIBREOFFICE IMPRESS

Bachelor's thesis

Supervisor: Edmund Laugasson
Master’s degree

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Silver Valdvee

SUURENDAVA ESITLUSE FUNKTSIONAALSUSE LISAMINE LIBREOFFICE IMPRESSILE

bakalaureusetöö

Juhendaja: Edmund Laugasson
magistrikraad

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Silver Valdvee

17.05.2021

Abstract

This thesis produced a zooming and panning feature known as zooming presentations for LibreOffice Impress. To a student of software development, the thesis briefly describes methods for extending complicated and aged software. The requirements set for the planned feature were zooming in and out in LibreOffice Impress slideshow mode and then also transforming objects on the slideshow such that the viewport would appear to move. Existing code was used wherever possible, which was also the main difficulty of this work—finding ways to connect code across the codebase to create new features. Tangled class hierarchies and high complexity of development were observed. From setting up a development environment and weighing different options for integrated development environments, this thesis found Visual Studio Code to be the best for specifically LibreOffice development.

This thesis is written in English and is 23 pages long, including 4 chapters, 33 figures and 0 tables.

Annotatsioon
Suurendava esitluse funktsionaalsuse lisamine LibreOffice Impressile

Selle lõputöö tulemusena valmis LibreOffice Impressi tarkvarale lisa, mis lubab kasutada suurendava esitluse võimalusi. Tarkvaraarenduse tudengile on lõputöö abiks, sest kirjeldab lühidalt keerulise ja vana tarkvara arendamise tehnikaid. Tarkvarale kehtestatud nõudeks oli esitlusrežiimis pildi suurendamine ja vähendamine ning pildi liigutamine. LibreOffice'is juba olemasolevat koodi kasutati igal võimalusel ning see oli ka raskeim osa tööst — koodi ühendamine kogu koodibaasi ulatuses. Arendustegevuse käigus taluti segast klassihierarhia ülesehitust ja üldist keerukust memotehnikaid kasutades. Arenduskeskonna üles seadmisel pandi tähele, et konkreetselt LibreOffice Impressi arenduseks on parim integreeritud programmeerimiskeskond Visual Studio Code.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 23 leheküljel, 4 peatükki, 33 joonist, 0 tabelit.

List of abbreviations and terms

IA	Department of Computer Systems
LO	LibreOffice
SID	Slot ID
GUI	Graphical User Interface
URL	https://et.wikipedia.org/wiki/Internetiaadress

Table of Contents

1 Introduction....................11

1.1 General background....................11

1.2 Problem formulation....................11

1.3 Objectives and software requirements....................12

1.4 Overview....................12

2 Methodology....................14

2.1 Overview of LibreOffice Impress codebase....................14

2.2 Overview of tools used....................15

2.3 Overview of the development process....................16

2.4 Edits to existing code....................21

2.4.1 svx/sdi/svx.sdi line 9126....................21

2.4.2 sd/source/ui/view/drviewse.cxx line 728....................21

2.4.3 sd/sdi/_drvwsh.sdi....................22

2.4.4 include/sfx2/sfxsids.hrc....................23

2.4.5 slideshow/source/engine/slideshowimpl.cxx....................23

2.4.6 ZoomingAnimation.cxx....................24

2.4.7 ZoomingAnimation.hxx....................25

2.4.8 Service.hxx....................26

2.4.9 NotificationCenter in Service.hxx....................27

2.5 Takeaways....................28

2.5.1 File slideshow/source/engine/slideshowimpl.cxx....................28

2.5.2 cppcanvas/source/inc/servicefolder/Service.hxx....................30

2.6 Fullfillment of software requirements....................31

3 Findings and discussion....................33

4 Summary....................36

References....................37

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis....................38

Appendix 2 slideshow/source/engine/animationnodes/ZoomingAnimation.cxx...........39
Appendix 3 – slideshow/source/inc/ZoomingAnimation.hxx.......................................42
Appendix 4 – Service.hxx..44
Appendix 5 – SlideshowImpl.cxx constructor..46

This table of contents contains file paths that are necessary in full because some of the files exist under identical names in different paths. There are no URL-s in headings.

List of Figures

Figure 1: IRC chat history me (silver_est) getting help with code................................19
Figure 2. Creating patches with GIT..20
Figure 3. The SID definition with properties...21
Figure 4. The exec method here responds to the SID..22
Figure 5. The method name for this slot is defined...23
Figure 6. Slot assigned an ID in include/sfx2/sfxsids.hrc..23
Figure 7. Initial setup of a slideshow regarding zooming presentations.........................24
Figure 8. Next slide listeners are not fired unless all zooming transitions are done.......24
Figure 9. Calculating timeTaken is essential in the interpolation..................................24
Figure 10. Interpolation for translations..25
Figure 11. Declaration of ZoomingAnimation...25
Figure 12. Singleton class S provides communication and storage between modules....26
Figure 13. A singleton object is initialized lazily once the getInstance2 method is called, and then the method returns the same object every time it is called..............................26
Figure 14. Definition of a macro that stores a visibility attribute..................................26
Figure 15. Events can hold any data. The templated notification system is very flexible. 27
Figure 16. An empty struct for an event type...27
Figure 17. Templates used to create a hub of listeners where all kinds of events can be fired and responded to..27
Figure 18. Lazy initialization of a static templated vector of callback functions...........27
Figure 19. notificationCenter is a member of S...27
Figure 20. Visibility attribute setting syntax...28
Figure 21. A basic setup of a vector that already contains elements..............................28
Figure 22. Template function calls to clear listeners for two events..............................28
Figure 23. A listener is added, which makes reacting to events possible.......................28
Figure 24. The method for accessing the center iterator..29
Figure 25. Accessing and incrementing the center points iterator.................................29

Figure 26. Constructor arguments are taken from make_shared....................................29
Figure 27. A struct for passing constructor arguments...30
Figure 28. Defined as an extern method in the header file, the symbol for getInstance2() is defined in a library...30
Figure 29. getListeners method...30
Figure 30. Reinserting an activity to the queue allows the activity to run again when its turn comes...31
Figure 31. Giving the activity a chance to run again if it wants by requeueing it...........31
Figure 32. Menuitem for adding an item to the zoom transition list...............................32
Figure 33. License agreement that was sent to the LibreOffice community mailing list. 32

1 Introduction

1.1 General background

Using presentation software allows illustrating text with media and is more effective than simple text. LibreOffice Impress (later also LO Impress) is free and open-source presentation software. It enables users to compose, draw, write, decorate and animate their slides and present them while looking at notes. Impress has all the advantages of an open-sourced program. The most known LibreOffice competitors are Microsoft Powerpoint and Prezi. It is also worth noting that LibreOffice has a chain of ancestors, the latest and most well known being Apache OpenOffice [1] .

LibreOffice Impress has a slight advantage over both Microsoft Powerpoint and Prezi [6] because LO code is shared. Being customizable by anyone, Impress is unique and has added potential to fit to users' needs. Powerpoint is customizable only through extensions [5] . This restricts the degree to which Powerpoint can be adjusted.

Zooming presentations are also available in a software called Sozi [14] . Sozi isn't a standalone application like LibreOffice Impress is. Sozi relies on the principle that a software should do one thing and do it well. Inkscape can be used to create presentations with Sozi. Sozi also has the advantage of being open-source.

1.2 Problem formulation

LibreOffice Impress does not yet have zooming presentations. A zooming presentation scales a slide and its contents and moves pans it across the screen. When zooming is performed on an object smaller than a pixel, it is invisible until the zoom increases its size. Impress's nonfree software contestants do have the feature set, and it is due time for LibreOffice Impress to have the same features. To have the best publicity and use for this feature, it makes much sense to include it in LibreOffice Impress.

1.3 Objectives and software requirements

This work proposed to implement zooming presentations in LibreOffice Impress. Only zooming and panning of viewport were attempted. Recognizing that LibreOffice has a colossal codebase and is itself hard to navigate, a complete experience of zooming presentations was not planned. Because of the amount of time needed to test and debug a complex application thoroughly, this would be unsafe to attempt in the relatively short thesis timespan.

It is not possible for zooming presentations to reach a LibreOffice release without having a corresponding standard released to support saving the order of zooms into an odp file. At the same time, it is incredibly helpful and welcome for people to implement new features and work on LibreOffice. Without working code for zooming presentations, there would be no motivation for technical writers to add this functionality into a standard.

Further works on this topic would have to develop the software such that:

- zooming transitions can be reordered and removed,
- zooming transitions can be registered comfortably using the already available viewport in sd module,
- hiding of elements in transitions is possible, like in Prezi.

1.4 Overview

This thesis is constructed as follows. Chapter 2 reports the methodology. First, the LibreOffice codebase is described in detail for the components correlated with the thesis problem. Then an overview of all the tools utilized for development follows. Additionally, the complex development process is explained. Edits done to the code are then described. Finally, some clever C++ takeaways from reading and writing code are listed and explained. The final product of this work is described. It is shown how a smooth zooming transition is now possible between presentation objects. Chapter 3 delves into analysis and discussion on the topic of LibreOffice Impress development. In Chapter 4, a summary is provided.

2 Methodology

2.1 Overview of LibreOffice Impress codebase

“LibreOffice is Free Software. LibreOffice is made available subject to the terms of the Mozilla Public License v2.0 which is reproduced below. It is based on code from Apache OpenOffice made available under the Apache License 2.0 but also includes software which differs from version to version under a large variety of other Open Source licenses.” See [3] .

The code of LibreOffice Impress is a module on top of LibreOffice Draw. The latter is a drawing software component of the LibreOffice suite. LibreOffice Draw and Impress both use the **sd** module for editor features. The **slideshow** module is unique to LibreOffice Impress. There, transitions, animations, slide changes, sound playback, and even physics for animations are implemented. Some of the code is quite aged, dating back to a predecessor of LibreOffice Draw named StarDraw. That is from where the module name **sd** originates.

An ***Activity*** is something that gets queued. ***ActivityQueue*** is what keeps a queue of activities to be completed and runs at the request of a ***SlideshowImpl***. Then it loops ***Activities*** and queues an ***Activity*** regularly when requested by the running ***Activity***. An A***ctivity*** consists of an ***Animation*** and struct of options related to it. Struct ***CommonOptions*** holds options to establish the length of the ***Animation***, potential acceleration, deceleration and enables storage of some pointers to shape objects — the latter was not employed. The queue system is clever. It allows for various ***Activities*** to be taking turns to repeat.

ZoomAnimation was added to permit zooming animations. Before, animations were only considered as a way to move an object. Additionally, scaling and transforming objects of the slides was adopted from the existing code. ***ZoomAnimation*** is a derived class of ***NumberAnimation***, which is a derived class of ***Animation***. While an abstract

Animation class should only be concerned with having virtual functions for performing the animation: start, perform and end. Base class ***Animation*** still contained specific parameters in a method for animating special shapes, undesirable base-class design. It is apparent that all animations should inherit from ***Animation***. As an unexpected discovery, a few subclasses, including the aforementioned ***ZoomAnimation***, overlooked the shape specifics in an interesting way: giving only types for the overridden method and dropping parameter names.

```
osl::MutexGuard const guard( m_aMutex );
```

2.2 Overview of tools used

Different operating systems were tried. It was challenging to set up the build process on Microsoft Windows 10. Ubuntu version 20.04 was chosen instead. On Ubuntu, libraries used in the build process are obtained easily by issuing commands to the Apt package manager. The same cannot be said about Windows 10, where build dependencies were supposed to be acquired by a broken script written by a LibreOffice contributor. Ubuntu was chosen for these advantages. Different hosts for running Ubuntu were tried. First, deploying Ubuntu on a virtual machine on the Windows 10 host was tried. This was then cancelled due to loss of performance. The build process is lengthy, and any performance loss adds hours of waiting time. A complete build of LibreOffice takes about 1 hour and 10 minutes on a Ryzen 7 4700U laptop processor with eight cores. Being able to use one fewer core would add an hour to the compilation process. Virtualized cores on a virtual machine are also less performant than real hardware processor cores.

Many code editors and Integrated Development Environments (IDEs) were considered. Jetbrains CLion [8] was a good candidate until the realization that the inheriting Makefiles [7] of LibreOffice cannot easily be made to work with CLion. Apache NetBeans [9] was extensible with support for C++, but it was disregarded again for the difficulty of setup. Finally, Visual Studio Code was used as a light editor. The configuration file for this editor was automatically generated by a script that comes with the LibreOffice codebase. The configuration that was generated by Visual Studio Code was later used with CLion to configure the latter. Then when configured, CLion offered many advanced features and with quick response times. This made CLion the best option. CLion also supported a feature for which an extension was downloaded for Visual Studio Code: numbered bookmarks are a big timesaver when working with huge codebases.

Ccache [10] was used in the compilation process and saved hundreds of hours of time. Each fix would have left me waiting for the compiler if it wasn't for the precompiled headers of Ccache. Reading compiler and linker errors was the only way available to see if the code was correct.

The complete build process of LibreOffice takes an hour, but after that, additional changes take minutes or seconds to compile. A change in a file included in many places would trigger compilations of many more files than a change in a rarely included or self-contained file.

Debugging was essential in fixing runtime errors and segmentation faults. Visual Studio Code integrated debugging tools [11] were used, but they were too wrongly configured. With a correct configuration "jump over" operation of the debugger would not have jumped back and forth while also moving forward. Seeing buggy situations occurring in the integrated debugger makes using GDB more appealing. At the same time, debugging actions in a graphical environment are easier to remember and do not require consulting a manual even for once.

Before patches could be uploaded to https://gerrit.libreoffice.org/ an account had to be made. A tool called logerrit was also used to simplify uploading patches [15] .

2.3 Overview of the development process

Studying C++ and learning to work with the GNU G++ compiler was among the first things accomplished. An application can only be written in C++ with knowledge of syntax and experience for solving both compilation and linker problems.

In the beginning, many unpleasant timesavers were developed to verify if zooming the viewport in a slideshow was reasonably doable. Transitions from slide to slide were the first place that already had code for transforming the slides. Scaling, also known as zooming, was then added to slide push transitions where the current slide moves up, pushed by the next slide from below.

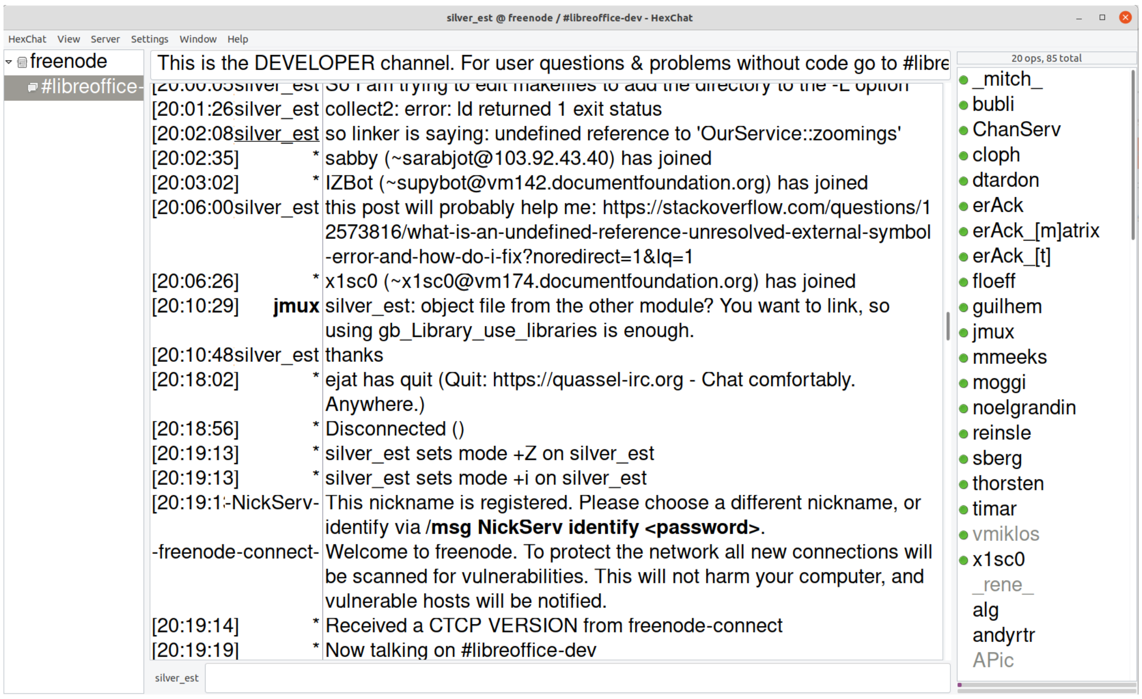
During the later parts of the development process, a digital notebook was kept. Noting the task at hand was crucial because all tasks consist of tens of subtasks. Investigating any of them without a fitting procedure would surely result in losing track of the remainder. Much of the existing code is also very deeply nested object-oriented code with several hierarchies of objects interleaved. It was frequent to have factories for factories. One instance of a misleading documentation comment was noticed — a namespace was labelled a class.

Variables that were critical for the task were written down as a tuple of name and location.

Notekeeping was not enough on its own. Browsing through code required time because overlooking any parts of it would cause a misunderstanding of the system. Reading everything in the domain of LibreOffice Impress was a necessity to manifest an understanding that would save time later.

IRC is a messaging protocol that was used to get help with the process. It has a few caveats:

- not being able to see messages from when the IRC client was disconnected from the server
- on some IRC clients, an inconvenient login mechanism
- no support for collapsible text or formatted multiline text


silver_est @ freenode / #libreoffice-dev - HexChat
HexChat View Server Settings Window Help
freenode
#libreoffice-
20 ops, 85 total
silver_est

```
This is the DEVELOPER channel. For user questions & problems without code go to #libre
[20:00:05silver_est So I am trying to edit makefiles to add the directory to the -L option
[20:01:26silver_est collect2: error: ld returned 1 exit status
[20:02:08silver_est so linker is saying: undefined reference to 'OurService::zoomings'
[20:02:35]        * sabby (~sarabjot@103.92.43.40) has joined
[20:03:02]        * IZBot (~supybot@vm142.documentfoundation.org) has joined
[20:06:00silver_est this post will probably help me: https://stackoverflow.com/questions/1
                    2573816/what-is-an-undefined-reference-unresolved-external-symbol
                    -error-and-how-do-i-fix?noredirect=1&lq=1
[20:06:26]        * x1sc0 (~x1sc0@vm174.documentfoundation.org) has joined
[20:10:29]     jmux silver_est: object file from the other module? You want to link, so
                    using gb_Library_use_libraries is enough.
[20:10:48silver_est thanks
[20:18:02]        * ejat has quit (Quit: https://quassel-irc.org - Chat comfortably.
                    Anywhere.)
[20:18:56]        * Disconnected ()
[20:19:13]        * silver_est sets mode +Z on silver_est
[20:19:13]        * silver_est sets mode +i on silver_est
[20:19:1:-NickServ- This nickname is registered. Please choose a different nickname, or
                    identify via /msg NickServ identify <password>.
-freenode-connect- Welcome to freenode. To protect the network all new connections will
                    be scanned for vulnerabilities. This will not harm your computer, and
                    vulnerable hosts will be notified.
[20:19:14]        * Received a CTCP VERSION from freenode-connect
[20:19:19]        * Now talking on #libreoffice-dev
```

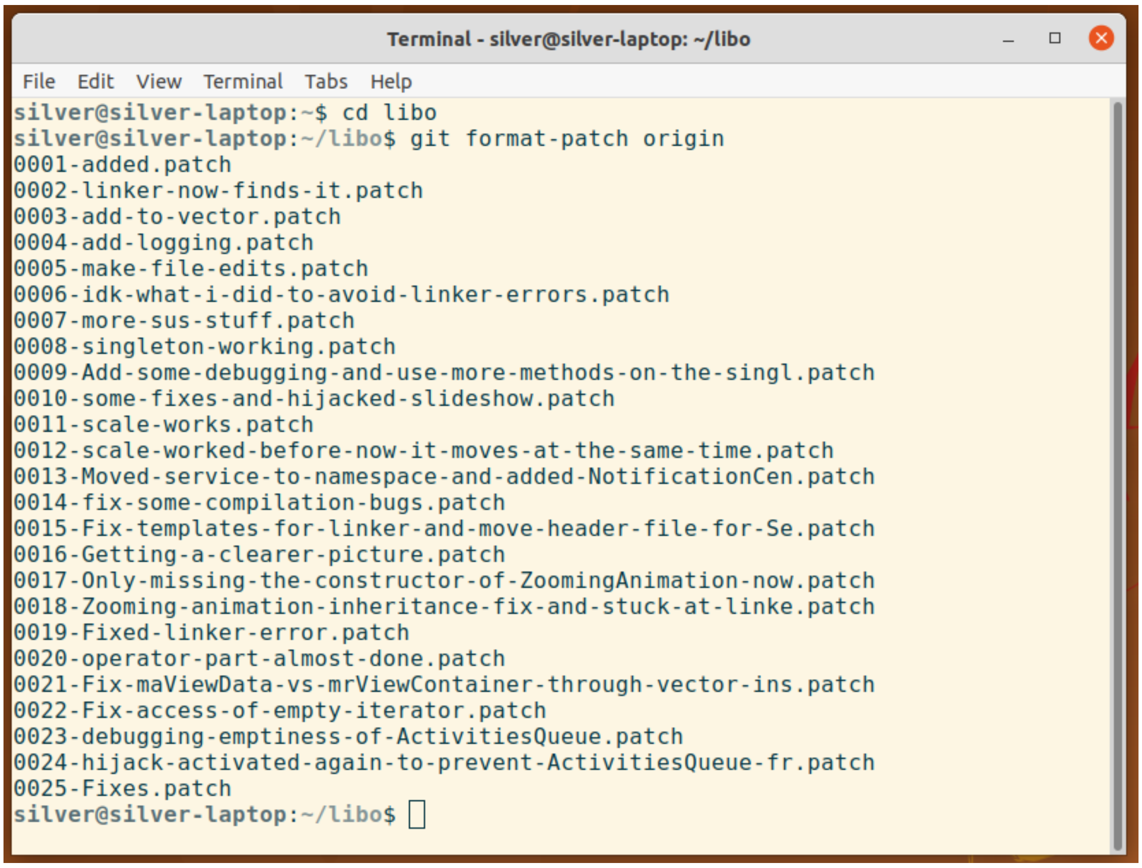
```
_mitch_
bubli
ChanServ
cloph
dtardon
erAck
erAck_[m]atrix
erAck_[t]
floeff
guilhem
jmux
mmeeks
moggi
noelgrandin
reinsle
sberg
thorsten
timar
vmiklos
x1sc0
_rene_
alg
andyrtr
APic
```

Figure 1: IRC chat history me (silver_est) getting help with code

The laptop running the IRC client was constantly powered on to mitigate these deficiencies. HEX IRC client was used because it preserves chat records from when it has been connected. A lot of questions were asked on LibreOffice-dev IRC channel, and multiple people responded to most questions. Getting started with the development process was more straightforward, thanks to the help received. Usually, help was needed finding places in code where a specific functionality could be implemented.

Upon configuration, the free HEX IRC client asked for multiple alternative usernames to use if one of them was unavailable. When it lost connection to the internet, the regular name was still being regarded logged in, and HEX logged in with an alternative name.

GIT was used for backups. For every backup, modified files and untracked files that needed to be committed and were added to the GIT index. Then a commit was created specifying a generic description of changes done. GIT commits were usually formatted using a ghost prefix to establish a format for the title of the commit message. The ghost prefix is a sentence: "When the changes from this commit are applied, it will". Then it is followed, for example, by "move slot id, add missing parameters".


Terminal - silver@silver-laptop: ~/libo
File Edit View Terminal Tabs Help

```
silver@silver-laptop:~$ cd libo
silver@silver-laptop:~/libo$ git format-patch origin
0001-added.patch
0002-linker-now-finds-it.patch
0003-add-to-vector.patch
0004-add-logging.patch
0005-make-file-edits.patch
0006-idk-what-i-did-to-avoid-linker-errors.patch
0007-more-sus-stuff.patch
0008-singleton-working.patch
0009-Add-some-debugging-and-use-more-methods-on-the-singl.patch
0010-some-fixes-and-hijacked-slideshow.patch
0011-scale-works.patch
0012-scale-worked-before-now-it-moves-at-the-same-time.patch
0013-Moved-service-to-namespace-and-added-NotificationCen.patch
0014-fix-some-compilation-bugs.patch
0015-Fix-templates-for-linker-and-move-header-file-for-Se.patch
0016-Getting-a-clearer-picture.patch
0017-Only-missing-the-constructor-of-ZoomingAnimation-now.patch
0018-Zooming-animation-inheritance-fix-and-stuck-at-linke.patch
0019-Fixed-linker-error.patch
0020-operator-part-almost-done.patch
0021-Fix-maViewData-vs-mrViewContainer-through-vector-ins.patch
0022-Fix-access-of-empty-iterator.patch
0023-debugging-emptiness-of-ActivitiesQueue.patch
0024-hijack-activated-again-to-prevent-ActivitiesQueue-fr.patch
0025-Fixes.patch
silver@silver-laptop:~/libo$ 
```

Figure 2. Creating patches with GIT.

The command "git format-patch origin" was applied to create a series of ".patch" files. These patch files include changes relative to the previous patch file. The first patch file is dependent on the commit that was pulled from the remote origin for work on the zooming features. See image below

A separate remote repository for code was not established because of the excessive storage requirements to hold the entire codebase. Instead, all the patch files were backed up to Google Drive. It was imperative to name the folder these patches are stored in after the starting git commit hash. Then if progress was lost, everything could be restored by downloading the codebase from LibreOffice, checking out to the starting commit using the hash in the folder name, and applying patches.

Figuring out which files to edit was difficult because even with a potential keyword to search for, the same keywords like “AffineTransform”, “Animation”, “slideshow”, and even filenames that contain these are in multiple modules. Knowing how to edit these files sometimes required connecting parts of code and copying code into another place.

2.4 Edits to existing code

This subsection presents the most important edits done in the code base of LO.

2.4.1 svx/sdi/svx.sdi line 9126

Slots are required for creating new commands, and commands can be executed from all customizable button locations in LibreOffice Impress. Slots connect C++ code to XML.

Figure below, see Figure 3, shows a slot file to adjust some slot options.

```
SfxVoidItem AddToTransitionList SID_ADD_TRANSITION_TO_LIST
()
[
    AutoUpdate = FALSE,
    FastCall = FALSE,
    ReadOnlyDoc = TRUE,
    Toggle = FALSE,
    Container = FALSE,
    RecordAbsolute = FALSE,
    RecordPerSet;

    AccelConfig = TRUE,
    MenuConfig = TRUE,
    ToolBoxConfig = TRUE,
    GroupId = SfxGroupId::View;
]
```

Figure 3. The SID definition with properties.

2.4.2 sd/source/ui/view/drviewse.cxx line 728

The final destination of a slot is C++ code, see Figure 4. The C++ code first gets a reference to the singleton and then pushes an SdrObject to its vector zoomings. Figure

11 shows the final destination of a command. In this code fragment, the slot ID is checked to be SID_ADD_TRANSITION_TO_LIST. Then it is checked that an object is selected. A reference is acquired to the singleton that I created, and the zoomings vector on it gets a new SdrObject inserted. This command was attached to a method on an existing class that already had slots attached to it. Doing so helps make sure that everything works the first time but also may confuse readers of the code.

```
void DrawViewShell::AddToTransitionList(SfxRequest& rReq){
    sal_uInt16 nSId = rReq.GetSlot();
    if(!mpDrawView->AreObjectsMarked()){return;}
    switch(nSId)
    {
        case SID_ADD_TRANSITION_TO_LIST:
        {
            if (mpDrawView->GetMarkedObjectList().GetMarkCount() > 0)
{
                ::tools::silverdev::S& s;
                s = ::tools::silverdev::getInstance2();
                auto name = mpDrawView->
GetMarkedObjectList().GetMark(0)->GetMarkedSdrObj()->GetName();
                SAL_DEBUG(name << " was added to Transitions list.");
                SdrObject* sdrObj = mpDrawView->
GetMarkedObjectList().GetMark(0)->GetMarkedSdrObj();
                s.zoomings.push_back(sdrObj);
                rReq.Ignore();
            }
        }
        break;
    }
}
```

Figure 4. The exec method here responds to the SID.

There are many more commands in LibreOffice. These commands can be assigned to different buttons in the application itself through GUI configuration. In this work, the command SID_ADD_TRANSITION_TO_LIST was added to the context menu that is displayed when a text box is selected. This simplifies the process of adding a new zooming transition. With one right-click, the text box can be selected, and the context menu also opens.

2.4.3 sd/sdi/_drvwsh.sdi

A connection between a method and SID (Slot ID) has to be established. See Figure 5.

```
SID_ADD_TRANSITION_TO_LIST
[
    ExecMethod = AddToTransitionList;
]
```

Figure 5. The method name for this slot is defined.

2.4.4 include/sfx2/sfxsids.hrc

An index is also assigned to a slot. After a special metaprogramming compilation step, this ends up in a big array at SID_SD_START+147 array index. Integer constant SID_SD_START and similarly named variables used in SID indexing context refer to numbers. See Figure 6.

```
#define SID_ADD_TRANSITION_TO_LIST (SID_SD_START+147)
```

Figure 6. Slot assigned an ID in include/sfx2/sfxsids.hrc.

2.4.5 slideshow/source/engine/slideshowimpl.cxx

SlideShowImpl hosts a slideshow and fires ***NotificationCenter*** events from existing code. Also initialises listeners to the events in its constructor. ***ZoomingAnimation*** objects are created in one of its listeners. Saving the locations of objects at the start of the slideshow is also an important step, see Figure 7.

```
using namespace ::tools::silverdev;
S& s = getInstance2();
s.centersOfObjects.clear();
SaveZoomSdrObjectCenters();
s.zoomIterator = s.zoomings.begin();
s.centerIterator = s.centersOfObjects.begin();
s.isIndexAtLast = false;
s.isIndexAtFirst = true;
s.notificationCenter.clearListeners<PreviousZoomEvent>();
s.notificationCenter.clearListeners<NextZoomEvent>();
s.notificationCenter.addListener<PreviousZoomEvent>(
    [this, &s](PreviousZoomEvent& event) {
        if(!s.isIndexAtFirst){
            auto center = nextCenter(s);
            queueZoomActivity(center);
        }
    });
s.notificationCenter.addListener<ViewUpdateEvent>([this]
(ViewUpdateEvent& event)

{ maEventMultiplexer.notifyViewsChanged(); });

s.notificationCenter.addListener<NextZoomEvent>(
    [this, &s](NextZoomEvent& event)
    {
        SAL_DEBUG("NEXT ZOOM REQUEST RECEIVED");
        if(!s.isIndexAtLast){
            basegfx::B2DPoint center = nextCenter(s);
            queueZoomActivity(center);
        }
    });
```

Figure 7. Initial setup of a slideshow regarding zooming presentations.

```
using namespace ::tools::silverdev;
S& s = getInstance2();
if(s.zoomings.empty() || s.isIndexAtLast){
    maListenerContainer.forEach<presentation::XSlideShowListener>(
        [&bReverse](const
uno::Reference<presentation::XSlideShowListener>& xListener)
        {
          return xListener->slideEnded( bReverse );
        }
    );
}
```

Figure 8. Next slide listeners are not fired unless all zooming transitions are done.

In SlideShowImpl::notifySlideEnded an if statement checks if the next slide should be triggered or not, see Figure 8.

2.4.6 ZoomingAnimation.cxx

```
bool ZoomingAnimation::operator()(double t)
{
    if (zoomTargetObject == nullptr)
    {
        return false;
    }
    double timeTaken = t - lastTimePoint;
```

Figure 9. Calculating timeTaken is essential in the interpolation.

In this work, t ranges from 0 to 1 through the animation. If the animation is 6 seconds long then and has been running for 0 seconds, then t is 0. If the animation has been running for 3 seconds then t is 0.5. At the end of the animation t is 1. Time t which has passed since the last execution of the animation is here called timeTaken and is later multiplied with the planned translation amounts on x and y axis (see Figure 10).

```
B2DPoint getTakenTranslation(double timeTaken, B2DPoint difference) {
    return B2DPoint(difference.getX() * timeTaken * 1000.0,
difference.getY() * timeTaken * 1000.0);
}
```

Figure 10. Interpolation for translations.

2.4.7 ZoomingAnimation.hxx

This is a class for animating. This contains lots of code borrowed from other files that also change slides, make transitions. Also inherits from ***NumberAnimation***. ***Animation*** is run by ***ActivityQueue*** through an overloaded operator(). All definitions and a few extra helper methods are in ZoomingAnimation.cxx.

```
class ZoomingAnimation : public NumberAnimation
{
public:
    ZoomingAnimation(ZoomingParameters zp);
    ~ZoomingAnimation();
    // Animation interface
    void prefetch();
    void start(const AnimatableShapeSharedPtr&, const
ShapeAttributeLayerSharedPtr&);
    void end();
    void end_();
    // NumberAnimation interface
    bool operator()(double nValue);
    double getUnderlyingValue() const;
    UnoViewContainer* mrViewContainer;
private:
    // difference sums are only used for debugging purposes
    double differenceSumX;
    double differenceSumY;
    bool isFirstTransformation;
    double differenceScreenSides;
    basegfx::B2DPoint targetPosition;
    basegfx::B2DVector currentScreenSize;
    double lastTimePoint;
    std::vector<basegfx::B2DPoint> screenSizesPerView;
    std::vector<basegfx::B2DPoint> startingDiffs;
    std::vector<basegfx::B2DPoint> differences;
    ScreenUpdater* maScreenUpdater;
    FrameSynchronization* maFrameSynchronization;
    EventMultiplexer* maEventMultiplexer;
    basegfx::B2DSize slideSizeAtAnimationStart;
    const SdrObject* zoomTargetObject;
    bool mbAnimationStarted;
    double mfDuration;
    double mfPreviousElapsedTime;
    ::std::shared_ptr<Slide> mpCurrentSlide;
};
```

Figure 11. Declaration of ZoomingAnimation.

2.4.8 Service.hxx

This has events and a singleton. It stores a vector of ***SdrObject*** pointers in zoomings. An iterator is also stored for the same vector. Then the vector is iterated as nextSlide events happen, and new ***Activities*** are created each time for ***ZoomAnimations*** to run off.

```
class  S
{
    public:
        bool isIndexAtFirst;
        bool isIndexAtLast;
        std::vector<SdrObject*>::iterator zoomIterator;
        std::vector<basegfx::B2DPoint>::iterator centerIterator;
        bool hasZooms();
        std::vector<basegfx::B2DPoint> centersOfObjects;
        std::vector<SdrObject*>  zoomings;
        NotificationCenter notificationCenter;
        S();
        S(S const&)               = delete;
        void operator=(S const&)  = delete;
};
```

Figure 12. Singleton class S provides communication and storage between modules.

A fine singleton pattern was adopted. This enabled all methods in connected modules to communicate. Through that communication, an event system was established.

Only the same object is always returned when getInstance2 is called. This object then contains an iterator for zoomings, the zoomings vector itself, a notification center and a helper method for checking if vector zoomings is empty or not.

```
SAL_DLLPUBLIC_EXPORT S& getInstance2()
{
    static S instance;
    return instance;
}
```

Figure 13. A singleton object is initialized lazily once the getInstance2 method is called, and then the method returns the same object every time it is called.

See Figure 13 for lazy initialization of a singleton. It should also be mentioned that SAL_DLLPUBLIC_EXPORT is a macro. See Figure 14.

```
define SAL_DLLPUBLIC_EXPORT  __attribute__ ((visibility("default")))
```

Figure 14. Definition of a macro that stores a visibility attribute.

An event can be any struct. See Figure 15:

```
struct PreviousZoomEvent {
    SdrObject* zooming;
};
struct NextZoomEvent {
    SdrObject* zooming;
};
```

Figure 15. Events can hold any data. The templated notification system is very flexible.

Events are not required to hold any data, but the type has to be declared and defined.

```
struct ViewUpdateEvent {};
```

Figure 16. An empty struct for an event type.

2.4.9 NotificationCenter in Service.hxx

```
class NotificationCenter {
    public:
        template<typename TEvent> void
addListener(std::function<void(TEvent&)> callback);
        template<typename TEvent> void clearListeners();
        template<typename TEvent> void fireEvent(TEvent& event);
    private:
        template<typename TEvent>
std::vector<std::function<void(TEvent&)>>& getListeners();
};
```

Figure 17. Templates used to create a hub of listeners where all kinds of events can be fired and responded to.

See Figure 17 for template code adding listeners to any events and firing them. The idea was found on StackOverflow in an answer by user aliased super [4] . This class uses templates and therefore needs to be included as a header with Service.hxx. These listeners are actually stored in a static vector, see the next figure, Figure 18.

```
template<typename TEvent>
std::vector<std::function<void(TEvent&)>>&
NotificationCenter::getListeners(){
    static std::vector<std::function<void(TEvent&)>> listeners;
    return listeners;
}
```

Figure 18. Lazy initialization of a static templated vector of callback functions.

Because the notification center is defined and initialized with singleton S, only one instance of NotificationCenter exists at all times (see Figure 18).

```
class  S {
    public:
        ...
        NotificationCenter notificationCenter;
```

Figure 19. notificationCenter is a member of S.

2.5 Takeaways

This subsection lists and explains important discoveries from development experience gained from this project.

2.5.1 File slideshow/source/engine/slideshowimpl.cxx

```
__attribute__ ((visibility("default")))
```

Figure 20. Visibility attribute setting syntax.

See Figure 20, it allows linker to see the symbol that this prepends from another object file.

```
s.zoomIterator = s.zoomings.begin();
```

Figure 21. A basic setup of a vector that already contains elements.

See Figure 21, zoomIterator needs to be initialized to the beginning of the zoomings vector.

```
s.notificationCenter
.clearListeners<::tools::silverdev::PreviousZoomEvent>();
s.notificationCenter
.clearListeners<::tools::silverdev::NextZoomEvent>();
```

Figure 22. Template function calls to clear listeners for two events.

See Figure 22. Any previous event listeners are cleared. A crash occurred otherwise on the next slideshow. As s is a reference to the singleton, these listeners do remain after a slideshow is closed. They are available as long as the main application runs.

```
s.notificationCenter.addListener<::tools::silverdev::NextZoomEvent>(
[this, &s](::tools::silverdev::NextZoomEvent& event){
```

Figure 23. A listener is added, which makes reacting to events possible.

See Figure 23. A lambda is used to react to any NextZoomEvents. The lambda grabs some of the context in which it was created: this and &s, both are in square brackets before the lambda argument list.

Some check are then performed to make sure that zoomings is not empty

```
basegfx::B2DPoint SlideShowImpl::nextCenter(tools::silverdev::S& s) {
    if (!s.centersOfObjects.empty()){
        basegfx::B2DPoint center(*s.centerIterator);
        SAL_DEBUG("center is this: " << center);
        if(s.centerIterator + 1 != s.centersOfObjects.end()){
            s.centerIterator++;
            s.isIndexAtFirst = false;
        } else {
            s.isIndexAtLast = true;
        }
        return center;
    }
}
```

Figure 24. The method for accessing the center iterator.

```
basegfx::B2DPoint SlideShowImpl::previousCenter(tools::silverdev::S&
s) {
    if (!s.centersOfObjects.empty()){
        basegfx::B2DPoint rect(*s.centerIterator);
        if(s.centerIterator + 1 != s.centersOfObjects.end()){
            s.centerIterator++;
            s.isIndexAtFirst = false;
        } else {
            s.isIndexAtLast = true;
        }
        return rect;
    }
}
```

Figure 25. Accessing and incrementing the center points iterator.

before a pointer dereference occurs.

```
NumberAnimationSharedPtr
pAnimation(std::make_shared<ZoomingAnimation>(pObj, 1.0,
maViewContainer, mpCurrentSlide));
```

Figure 26. Constructor arguments are taken from make_shared.

here a pointer type is created for ***ZoomingAnimation***, ***ZoomingAnimation*** receives constructor parameters through the standard library make_shared function, Figure 26.

The constructor parameters were later moved into a struct of parameters, this saves some screen space, see Figure 27.

```
struct ZoomingParameters {
    // This is saved from void
DrawViewShell::AddToTransitionList(SfxRequest& rReq){
    const basegfx::B2DPoint _targetPosition;
    // Duration of the zooming animation
    const double _fDuration;
    // A slide is a view, they are contained in this
    UnoViewContainer* _mrViewContainer;
    // A pointer to the current slide
    ::std::shared_ptr<Slide> _mpCurrentSlide;
    // An event needs to be fired when a view is updated to display
in on the screen
    EventMultiplexer* _maEventMultiplexer;
    // This is no longer used, wasn't necessary
    ScreenUpdater* _maScreenUpdater;
    // Frame synchronisation
    FrameSynchronization* _maFrameSynchronization;
    // Current screen size is saved to aim properly at screen objects
    basegfx::B2DVector _currentScreenSize;
};
```

Figure 27. A struct for passing constructor arguments.

2.5.2 cppcanvas/source/inc/servicefolder/Service.hxx

```
extern S& getInstance2();a
```

Figure 28. Defined as an extern method in the header file, the symbol for getInstance2() is defined in a library.

indicates to the linker that this symbol is defined externally

```
template<typename TEvent>
std::vector<std::function<void(TEvent&)>>&
NotificationCenter::getListeners(){
    static std::vector<std::function<void(TEvent&)>> listeners;
    return listeners;
}
```

Figure 29. getListeners method.

This code snippet demonstrates a clever initialization of a static vector in a template function. Also worth mentioning is the storage of function references in std::function, which can be held in a vector. A vector of functions is useful for storing listeners for a specific event and then accessing these callback methods to fire an event. See figures:

Figure 18, Figure 29.

LO Impress runs mainly on just one thread. This code file is from a class in ActivityQueue in LO by LO contributor.

```
if( bReinsert )
    maCurrentActivitiesReinsert.push_back( pActivity );
else
    maDequeuedActivities.push_back( pActivity );
```

Figure 30. Reinserting an activity to the queue allows the activity to run again when its turn comes.

```
while( !maCurrentActivitiesWaiting.empty() )
{
    // process topmost activity
    ActivitySharedPtr
pActivity( maCurrentActivitiesWaiting.front() );
    maCurrentActivitiesWaiting.pop_front();

    bool bReinsert( false );

    try
    {
        // fire up activity
        bReinsert = pActivity->perform();
    }
```

Figure 31. Giving the activity a chance to run again if it wants by requeueing it.

2.6 Fulfilment of software requirements

All software requirements were met. Zooming and panning simultaneously from one text-box to another was implemented. This means that the target object, to which a move is made, ends up in the center of the screen while the screen is zooming. This movement is interpolated linearly.

Features that were implemented are not yet useful to users. To be useful and applicable in real applications, more development needs to happen on further related requirements.

A letter was sent to the LibreOffice mailing list stating:

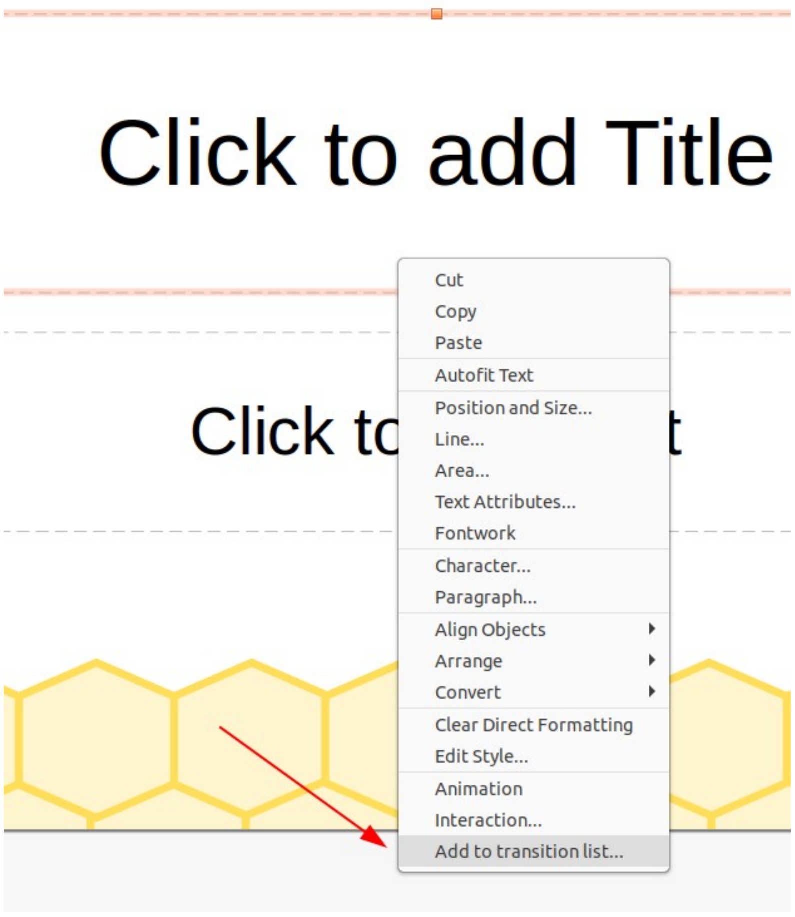


Figure 32. Menuitem for adding an item to the zoom transition list.

```
All of my past & future contributions to LibreOffice may
be licensed under the MPLv2/LGPLv3+ dual license.
```

Figure 33. License agreement that was sent to the LibreOffice community mailing list.

Code was uploaded to https://gerrit.libreoffice.org/c/core/+/115011 where the patch can also be updated, reviewed.

3 Findings and discussion

Impress code is properly structured, and its Makefile system works splendidly. The Makefiles are sectioned and provide easily understandable places to list files to be compiled, header locations and other libraries that are needed to compile. File names make sense but there are matching file names across directories and different modules, this lead to a confusing developer experience. Some features were first developed in sd/source/ui/slideshow/slideshowimpl.cxx but then to be used in a slideshow, ported to slideshow/source/engine/slideshowimpl.cxx. Matching method names were also a worry when automatic refactoring functions were used to rename a method to a more appropriate name. The refactoring system looks through its index of the entire codebase and finds methods with the same name both in comments and code itself. That is worrisome because unintended changes could then possibly be made by the refactoring software.

At the same time, some of the method names used were confusing for a “find in all files” search. For example, "start" for a method name in various classes is sluggish when "find in all files" search is considered. That leaves no alternative but to exclude other methods with the same name manually. While proper IDE support could mitigate this problem, these general method names become a problem that inhibits development speed. Studying from this, writing short universal method names should be avoided in the future. It can be said that typing a longer method name takes more time each time, but having software complete long names makes it a benefit after all. Tabnine [13] is software that uses AI to reduce the amount of typing dramatically. It predicts both long and short texts and works even when no IDE support for the language is available. Focus can be shifted from writing compact to writing long for smooth code navigation.

When the development process started, memory was a big problem, but that motivated developing a stack-based method using a text editor always to know what is being done and what is stopping it. It would be very wasteful not to keep track of subtasks and class hierarchies in the process. Here I found that having a concrete experience with software diagramming would have helped me. Not knowing any methods for representing class hierarchies as a drawing, I would not even attempt it unless my stack-based text processor method stops being productive.

The IDE was showing errors that did not make sense. It took tens of minutes to perform reference searches, did not help with includes and had broken refactoring features. Correctly setting up an IDE was not an easy task and thus was not accomplished. It would have been very beneficial for the IDE to find references to symbols in code correctly. At the same time, multiple extensions for Visual Studio Code were utilized. Extension "Bookmarks" was a great timesaver. A mapping of CTRL + number was set to go to the bookmark of that number. Then the actual bookmark was set using another mapping of SHIFT + CTRL + number. This easily saved hours of navigating code. Bookmarks were accessible in all files, and each number could be used many times — pressing CTRL + number many times for one number would cycle through the bookmark entries.

Hotkeys were used extensively in IDEs. Switching between the header and corresponding cxx files was done using ALT+O. Key F12 was used to go to definition or declaration, then to references. Rename refactoring was used to fix code that was less readable. Extract to function refactoring was not used extensively due to it being slow, taking 14 seconds.

Reliable methods were developed because of strong motivation for understanding a complex codebase. The code was: something unable to be held in a human head, required hundreds of text searches to navigate code, often a criss-cross of class hierarchies navigating through factories. A good workflow on a huge project helps keep programmers on track.

Developing the feature for Impress prompted finding places in code that could be improved. Having a call stack of three functions, each with the same name in a different file, makes for a confusing developer experience. Reading from LibreOffice developer guides and guide videos, people needlessly reformatting old code is a common problem they face. This makes it hard for patch reviewers to read the code because they need to distinguish code changes from formatting changes. They want to spend as little time as possible reviewing a patch. They are capable people themselves and could use that time to fix more problems in code or develop new features.

Developing free and open source projects is a nice pastime, but it cannot proceed on a large scale unless properly financed. Full-time developers for these projects are usually specialists funded by the users of the software. Nevertheless, there can be other good motivators. Even with no assumed income, earning a degree, bettering oneself and becoming well known are all good motivators for developers in open source projects. There are, however, very good examples of open source projects that are developed by people for their own needs.

It will take a user from a different product to find this feature of zooming presentations in LibreOffice Impress, and it might well be what keeps the user with this software. It is common for users to come and chat with developers asking why LibreOffice does not have all the features of its competitors and leave after asking developers if it could be implemented today or tomorrow. Now, if they find this feature in the product, they could start using it instead and stop waiting for it to be implemented.

4 Summary

This thesis produced a zooming and panning feature known as zooming presentations for LibreOffice Impress. To a student of software development, the thesis briefly describes methods for extending complicated and aged software. The requirements set for the planned feature were zooming in and out in LibreOffice Impress slideshow mode and then also transforming objects on the slideshow such that the viewport would appear to move. Existing code was used wherever possible, which was also the main difficulty of this work—finding ways to connect code across the codebase to create new features. Tangled class hierarchies and high complexity of development were observed. This thesis found Visual Studio Code to be the best for specifically LibreOffice development, from setting up a development environment and weighing different options for integrated development environments. From a statistical perspective, 257 lines of code were written, three new C++ classes implemented, 146 files of code were read. LibreOffice source contains approximately 120 000 source files.

References

[1] LibreOffice home page, https://www.LibreOffice.org/, May 17th 2021

[2] Vajna, Miklos. LibreOffice: Code Structure, Collabora Productivity, 2017, https://speakerd.s3.amazonaws.com/presentations/69a6d9f8e5a14a948b27796b4c73ae11/beginners-structure-locon-rome-2k17.pdf, May 17th 2021

[3] LibreOffice webpage, About us, https://www.LibreOffice.org/about-us/licenses, May 17th 2021

[4] pseudonym super, “AppEventManager”, https://stackoverflow.com/users/7703024/super, https://stackoverflow.com/a/59707355, May 17th 2021

[5] Powerpoint addins, https://docs.microsoft.com/en-us/office/dev/add-ins/powerpoint/powerpoint-add-ins#powerpoint-add-in-scenarios, May 17th 2021

[6] Prezi web application, https://prezi.com/, May 17th 2021

[7] GNU make https://www.gnu.org/software/make/, May 17th 2021

[8] Jetbrains CLion, https://www.jetbrains.com/clion/, May 17th 2021

[9] Apache NetBeans, https://netbeans.apache.org/, May 17th 2021

[10] Ccache — a fast C/C++ compiler cache, https://ccache.dev/, May 17th 2021

[11] C/C++ for Visual Studio Code, https://code.visualstudio.com/docs/languages/cpp, May 17th 2021

[12] GDB: The GNU Project Debugger, https://www.gnu.org/software/gdb/, May 17th 2021

[13] Tabnine, https://www.tabnine.com/v, May 17th 2021

[14] Sozi, https://sozi.baierouge.fr/, May 17th 2021

[15] Gerrit usage documentation, https://wiki.documentfoundation.org/Development/gerrit, May 17th 2021

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2

slideshow/source/engine/animationnodes/ZoomingAnimation.cxx

```
#include <tools/diagnose_ex.h>
#include <sal/log.hxx>

#include <basegfx/matrix/b2dhommatrix.hxx>

#include <cppcanvas/customsprite.hxx>

#include <transitionfactory.hxx>
#include <tools.hxx>
#include <memory>
#include <math.h>
#include <svx/svdobj.hxx>

#include <animatableshape.hxx>
#include <shapeattributelayer.hxx>
#include <ZoomingAnimation.hxx>
#include <basegfx/utils/canvastools.hxx>
#include <slide.hxx>
namespace slideshow::internal {
    ZoomingAnimation::~ZoomingAnimation()
    {
        end_();
    }

    // Animation interface

    void ZoomingAnimation::prefetch()
    {}


    void ZoomingAnimation::start( const AnimatableShapeSharedPtr&
/* rShape */,
                                  const ShapeAttributeLayerSharedPtr& /*
rAttrLayer */ )
    {
        if( !mbAnimationStarted )
        {
            mbAnimationStarted = true;
        }
    }


    // NumberAnimation interface
    void ZoomingAnimation::end() { end_(); }
    void ZoomingAnimation::end_()
    {
        if( !mbAnimationStarted )
            return;
        mbAnimationStarted = false;
    }
    cppcanvas::CustomSpriteSharedPtr createSprite(UnoViewSharedPtr
const & pView, basegfx::B2DSize const & rSpriteSize, double nPrio ) {
        // TODO(P2): change to bitmapsprite once that's working
        const cppcanvas::CustomSpriteSharedPtr pSprite(pView-
>createSprite( rSpriteSize,nPrio ));
        // alpha default is 0.0, which seems to be
        // a bad idea when viewing content...
        pSprite->setAlpha( 1.0 );
        // if (mbSpritesVisible)
        pSprite->show();
        return pSprite;
    }
    ::basegfx::B2ISize getEnteringSlideSizePixel(const
UnoViewSharedPtr& pView, ::std::shared_ptr<Slide> mpCurrentSlide)
    {
        return getSlideSizePixel(basegfx::B2DSize(mpCurrentSlide-
>getSlideSize()), pView);
    }
    ::cppcanvas::CustomSpriteSharedPtr createSprite(ViewEntry&
rEntry, ::std::shared_ptr<Slide> mpCurrentSlide)
    {
            // create entering sprite:
            const basegfx::B2ISize
enteringSlideSizePixel(getSlideSizePixel( basegfx::B2DSize( mpCurrent
Slide->getSlideSize() ), rEntry.mpView));
            return
createSprite(rEntry.mpView,basegfx::B2DSize( enteringSlideSizePixel )
,101);
    }

    bool ZoomingAnimation::operator()( double t )
    {
        SAL_DEBUG("Y");
        ViewsVec maViewData;
        for( const auto& pView : mrViewContainer ) {
            maViewData.emplace_back(pView);
        }
        const std::size_t nEntries = maViewData.size();
        for(::std::size_t i=0; i<nEntries; ++i)
        {
            ViewEntry& rViewEntry(maViewData[i]);
            const ::cppcanvas::CanvasSharedPtr
rDestinationCanvas(rViewEntry.mpView->getCanvas());
            ::cppcanvas::CustomSpriteSharedPtr
rSprite(createSprite(rViewEntry, mpCurrentSlide));
            //::cppcanvas::CustomSpriteSharedPtr&
rOutSprite( rViewEntry.mpOutSprite );
            const double sizeConst = 10;
            ::basegfx::B2DHomMatrix
aViewTransform(rDestinationCanvas->getTransformation());
            aViewTransform.scale(10 + t * sizeConst, 10 + t *
sizeConst);
            const ::basegfx::B2DPoint aPageOrigin(aViewTransform *
::basegfx::B2DPoint());
```

```
            rSprite->movePixel(aPageOrigin + ((t - 1.0) *
::basegfx::B2DSize(getEnteringSlideSizePixel(rViewEntry.mpView,
mpCurrentSlide))));
            rSprite->transform(aViewTransform);
        }
        return true;
    }

    double ZoomingAnimation::getUnderlyingValue() const
    {
        return 0.0;
    }
    ZoomingAnimation::ZoomingAnimation(const SdrObject*
zoomTargetObject, const double fDuration, UnoViewContainer&
mrViewContainer, ::std::shared_ptr<Slide> mpCurrentSlide) :
            zoomTargetObject(zoomTargetObject),
            mbAnimationStarted( false ),
            mfDuration(fDuration),
            mfPreviousElapsedTime(0.00f),
            mrViewContainer(mrViewContainer),
            mpCurrentSlide(mpCurrentSlide)
            {}
}
```

Appendix 3 – slideshow/source/inc/ZoomingAnimation.hxx

```
#include <vector>
#include <unoview.hxx>

namespace slideshow::internal {
    struct ViewEntry
    {
        explicit ViewEntry( const UnoViewSharedPtr& rView ) : mpView(
rView ){}
        /// The view this entry is for
        UnoViewSharedPtr                                      mpView;
        /// outgoing slide sprite
        std::shared_ptr<cppcanvas::CustomSprite>    mpOutSprite;
        /// incoming slide sprite
        std::shared_ptr<cppcanvas::CustomSprite>    mpInSprite;
        /// outgoing slide bitmap
        mutable SlideBitmapSharedPtr
mpLeavingBitmap;
        /// incoming slide bitmap
        mutable SlideBitmapSharedPtr
mpEnteringBitmap;

        // for algo access
        const UnoViewSharedPtr& getView() const { return mpView; }
    };
    typedef ::std::vector<ViewEntry> ViewsVec;
    class ZoomingAnimation : public NumberAnimation
    {
    public:
        ZoomingAnimation( const SdrObject* zoomTargetObject, const
double fDuration, UnoViewContainer& mrViewContainer,
::std::shared_ptr<Slide> mpCurrentSlide);
        ~ZoomingAnimation();
        // Animation interface
        void prefetch();
        void start(const AnimatableShapeSharedPtr& /* rShape */,
const ShapeAttributeLayerSharedPtr& /* rAttrLayer */);
        void end();
        void end_();
        // NumberAnimation interface
        bool operator()(double nValue);
        double getUnderlyingValue() const;
        UnoViewContainer& mrViewContainer;
    private:
        const SdrObject* zoomTargetObject;
        bool mbAnimationStarted;
        double mfDuration;
        double mfPreviousElapsedTime;
        ::std::shared_ptr<Slide> mpCurrentSlide;
    };
}
```

Appendix 4 – Service.hxx

```
#pragma once
#include <svx/svdobj.hxx>
#include <vector>
#include <functional>
namespace tools::silverdev {
    struct PreviousZoomEvent {
        SdrObject* zooming;
    };
    struct NextZoomEvent {
        SdrObject* zooming;
    };
    // inspired by https://stackoverflow.com/a/59707355
    class NotificationCenter {
        public:
            template<typename TEvent> void
addListener(std::function<void(TEvent&)> callback);
            template<typename TEvent> void clearListeners();
            template<typename TEvent> void fireEvent(TEvent& event);
        private:
            template<typename TEvent>
std::vector<std::function<void(TEvent&)>>& getListeners();
    };
    class  S
    {
        public:
            std::vector<SdrObject*>::iterator zoomIterator;
            bool hasZooms();
            SdrObject* takeNextZooming();
            std::vector<SdrObject*>  zoomings;
            NotificationCenter notificationCenter;
            S();
            S(S const&)                = delete;
            void operator=(S const&)   = delete;
    };
    extern S& getInstance2();
    template<typename TEvent>
    void NotificationCenter::addListener(std::function<void(TEvent&)>
callback){
        getListeners<TEvent>().push_back(std::move(callback));
    }
    template<typename TEvent>
    void NotificationCenter::fireEvent(TEvent& event){
        for(auto& listener : getListeners<TEvent>()){
            listener(event);
        }
    }
    template<typename TEvent>
    std::vector<std::function<void(TEvent&)>>&
NotificationCenter::getListeners(){
        static std::vector<std::function<void(TEvent&)>> listeners;
        return listeners;
    }
    template<typename TEvent>
    void NotificationCenter::clearListeners(){
        getListeners<TEvent>().clear();
    }
}
```

Appendix 5 – SlideshowImpl.cxx constructor

```
SlideShowImpl::SlideShowImpl(
    uno::Reference<uno::XComponentContext> const& xContext )
    : SlideShowImplBase(m_aMutex),
      maViewContainer(),
      maListenerContainer( m_aMutex ),
      maShapeEventListeners(),
      maShapeCursors(),
      maUserPaintColor(),
      maUserPaintStrokeWidth(4.0),
      mpPresTimer( std::make_shared<canvas::tools::ElapsedTime>() ),
      maScreenUpdater(maViewContainer),
      maEventQueue( mpPresTimer ),
      maEventMultiplexer( maEventQueue,
                          maViewContainer ),
      maActivitiesQueue( mpPresTimer ),
      maUserEventQueue( maEventMultiplexer,
                        maEventQueue,
                        *this ),
      mpDummyPtr(),
      mpBox2DDummyPtr(),
      mpListener(),
      mpRehearseTimingsActivity(),
      mpWaitSymbol(),
      mpPointerSymbol(),
      mpCurrentSlideTransitionSound(),
      mxComponentContext( xContext ),
      mxOptionalTransitionFactory(),
      mpCurrentSlide(),
      mpPrefetchSlide(),
      mxPrefetchSlide(),
      mxDrawPagesSupplier(),
      mxSBD(),
      mxPrefetchAnimationNode(),
      mnCurrentCursor(awt::SystemPointer::ARROW),
      mnWaitSymbolRequestCount(0),
      mbAutomaticAdvancementMode(false),
      mbImageAnimationsAllowed( true ),
      mbNoSlideTransitions( false ),
      mbMouseVisible( true ),
      mbForceManualAdvance( false ),
      mbShowPaused( false ),
      mbSlideShowIdle( true ),
      mbDisableAnimationZOrder( false ),
      maEffectRewinder(maEventMultiplexer, maEventQueue,
maUserEventQueue),
      maFrameSynchronization(1.0 /
FrameRate::PreferredFramesPerSecond)

{
    // keep care not constructing any UNO references to this inside
```

```
ctor,
    // shift that code to create()!

    uno::Reference<lang::XMultiComponentFactory> xFactory(
        mxComponentContext->getServiceManager() );

    if( xFactory.is() )
    {
        try
    {
            // #i82460# try to retrieve special transition factory
            mxOptionalTransitionFactory.set(
                xFactory->createInstanceWithContext(
                    "com.sun.star.presentation.TransitionFactory",
                    mxComponentContext ),
                uno::UNO_QUERY );
        }
        catch (loader::CannotActivateFactoryException const&)
    {
    }
    }

    mpListener = std::make_shared<SeparateListenerImpl>(
                              *this,
                              maScreenUpdater,
                              maEventQueue );
    maEventMultiplexer.addSlideAnimationsEndHandler( mpListener );
    maEventMultiplexer.addViewRepaintHandler( mpListener );
    maEventMultiplexer.addHyperlinkHandler( mpListener, 0.0 );
    maEventMultiplexer.addAnimationStartHandler( mpListener );
    maEventMultiplexer.addAnimationEndHandler( mpListener );
    // This is the end of slideshowimpl constructor and here's my
part of it -- silver_est
  ::tools::silverdev::S& s = ::tools::silverdev::getInstance2();
    s.zoomIterator = s.zoomings.begin();

s.notificationCenter.clearListeners<::tools::silverdev::PreviousZoomE
vent>();

s.notificationCenter.clearListeners<::tools::silverdev::NextZoomEvent
>();

s.notificationCenter.addListener<::tools::silverdev::PreviousZoomEven
t>([this, &s](::tools::silverdev::PreviousZoomEvent& event){

    });

s.notificationCenter.addListener<::tools::silverdev::NextZoomEvent>([
this, &s](::tools::silverdev::NextZoomEvent& event){
        if(s.zoomings.size()==0){return;}
         const SdrObject* pObj (*s.zoomIterator);
                //s.zoomIterator++;
        if(s.zoomIterator == s.zoomings.end()){return;}
              EventSharedPtr ZoomEndEvent(makeEvent([this](){},
"ZoomEndEvent"));
            double nTransitionDuration(5.0);
            sal_Int32 nMinFrames(5);
                     NumberAnimationSharedPtr
pAnimation(std::make_shared<ZoomingAnimation>(pObj, 1.0,
maViewContainer, mpCurrentSlide));
            auto d1 = mpCurrentSlide->getSlideSize();
            auto d = basegfx::B2DSize( d1 );
                   auto parameters =
ActivitiesFactory::CommonParameters(
                          ZoomEndEvent,
                           maEventQueue,
                           maActivitiesQueue,
                              nTransitionDuration,
                            nMinFrames,
                             false, // autoreverse
                           std::optional<double>(1.0), // repeats
                          0.0, // acceleration
                            0.0, // deceleration
                            ShapeSharedPtr(),
                               d // slide bounds
                       );
            ActivitySharedPtr
pZoomActivity(ActivitiesFactory::createSimpleActivity(parameters,
pAnimation, true /* is direction forward? */ ));
                      maActivitiesQueue.addActivity(pZoomActivity);
    });
}
```