Creativity Through Interaction: 10 projects using open-source software

Justin Yang
2014-09-04

Abstract

10 projects for the digital artist.

Contents

PROJECT 1 - MOUSE THEREMIN

1 PROJECT-ONE

With a few lines of code, we build a pretty useful and infinitely expandable digital instrument. All of the ingredients for digital performance are here, instrument, interface and score. It'll introduce us to basics concepts in coding, synthesis, SuperCollider, Processing, and give us a basic template where the sky's the limit.

1.1 Start At The End

EXPLORING IN THIS SECTION
—> Following Instructions
—> Wrist & Finger Dexterity
—> Mouse Accuracy & Speed
—> Digital Performance

Run the Code: Copy the SC code below into SuperCollider, or you can download it from here: Mouse Theremin

1. Run it as indicated.
2. See if you can figure out what to do without any further guidance.
3. Did you play a tune? Which one? My favourite is the Godfather theme.

```
//PLACE MOUSE CURSOR ON EACH OF THE NUMBERS IN ORDER AND HIT CMD-RETURN

(
// 1):
// Initialize
var currMidiNum, noteNames, octaves, screenRes, yres;
~pitchLo = \a0;
~pitchHi = \c8;

//MIDI NOTE DICTIONARY
////Allows you to input a symbol and get out the midi note number
////Use only these symbols:
//// \c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b

~mnd = ();
currMidiNum = 0;
noteNames = [\c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b];
octaves = [-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
octaves.do{ |it ix|
        noteNames.do{ |it2 ix2|
                ~mnd.put((it2 ++ it).asSymbol, currMidiNum);
                currMidiNum = currMidiNum+1;
        }
};
```

```
//FUNCTION TO LOOK UP Y-LOCATION FOR A PITCH

screenRes = "system_profiler SPDisplaysDataType | grep Resolution".
    unixCmdGetStdOut.findRegexp("(?<!@ )[0-9]{3,}").collect({|item| item
    [1].asInteger}).clump(2);
~yres = screenRes[0][1];

~midiToY = {
        arg midinote;
        ~mnd[midinote].linlin(~mnd[~pitchLo].asFloat, ~mnd[~pitchHi].
            asFloat, ~yres, 0.0).round;
};

//SYNTHDEF
SynthDef(\mouseTheremin, {
        arg pitchLo = 30, pitchHi = 80, pitchres=1.0;
        var osc, freq, env, envgate, pitch, amp;
        amp = MouseX.kr(-40, 10).dbamp;
        pitch = MouseY.kr(minval:pitchHi, maxval:pitchLo, lag:0.0001).round
            (pitchres);
        envgate = MouseButton.kr(lag:0.0001);
        env = EnvGen.ar( envelope:Env.asr(releaseTime:0.05), gate:envgate )
            ;
        freq = pitch.midicps;
        osc = SinOsc.ar( freq );
        Out.ar(0, osc*env*amp);
}).add;

) ////////////////////////////////////////////////////////

(
// 2):
// Boot Server
s.boot;
)

(
// 3):
// Start Synth
~mt = Synth(\mouseTheremin, [\pitchLo, ~mnd[~pitchLo], \pitchHi, ~mnd[
    ~pitchHi]]);
)


// TO STOP SYNTH OR USE CMD-.
~mt.free;



//UTILITIES

//EXAMPLE OF MIDI NOTE TO Y COORDINATE TO LOOK UP VALUES FOR YOUR
    PROCESSING PATCH:
////Use only these symbols:
//// \c, \cs, \d, \eb, \e, \f, \fs, \g, \gs, \a, \bb, \b
```

```
~midiToY.value(\g5);
~midiToY.value(\fs4);
//Look @ Post Window


//USE THESE TO 'PLAY ALONG' AND THEN FIND MIDI PITCH
~mspoll = {MouseY.kr(~mnd[~pitchHi], ~mnd[~pitchLo]).round.poll}.play;
~mnd.findKeyForValue(66);


~mspoll.free;
```

next section run through code resources

Appendix installing sc and processing

`http://www.processing.org`

Wikibooks home

`run:/Users/jyang/Desktop/SADFAY_33.wav` Play My Jam

2 Project 1 - Mouse Theremin

Example1: Summary

```
SynthDef(\money, {

}).add;
```

Example2: Useless code

```
SynthDef(\money, {

}).add;
```

3 wazzup

Is b introducin chapter3

3.1 Author, Layout Designer and Compositor

Every piece of writing – from a short handwritten note to a printed book – consists of two components: **contents** and **design.** In most cases the design is only a means to allow the reader to understand the contents more easily.

What is it? This **is me** trying to learn LaTeX/LaTeX

Author: The **contents** will be written by the author of the document. She usually is supported by the publisher, who issues guidelines about how the document should be written and conducts proofreading.

When using LaTeX the contents appears in the main part of the document – between `\begin{document}` and `\end{document}` – with some additional LaTeX-commands that describe the meaning of the different parts.

Designer: The **design** of a printed document is usually defined by the publisher who employs designers for this purpose.

The layout is defined in so-called class files. When writing a document you have to specify the class you want to use with the `\documentclass` command.

The document classes distributed with LaTeX have been designed by American layout designers according to the conventions used in the USA. The "Koma-Script" classes have been adapted to European taste and contain many small improvements.

Compositor: The compositor takes the contents from the author and formats it according to the layout the designer specified.

When using LaTeX, this job is done by the TeX program and the printing is done by a device driver.

3.2 Changing the Layout

If you want to use a layout different from the ones distributed with LaTeX, you have to take the following steps:

1. The layout has to be defined. This is usually a job for a professional designer.

→ Chapter 4

   You can find detailed information about how change the design in chapter4.

2. The layout has to be programmed in LaTeX. This can either result in a "package" that changes the behaviour of a standard class, or in the definition of a new "class". `Refman` used to be a package that changed the definition of the `article` class but is now a class of its own. When writing a new class, you will have more work, but as a result more freedom to change things.

→ Chapter 5

   Chapter5 contains more information about the new layout.

→ Chapter 5.6

   Where to put text

4 The Art of Layout-Design

4.1 Common Rules

There are almost no common rules because every kind of document has different requirements and needs a specific layout. This layout should consider *who* will read the document, and *how*.

An important criteria is if the reader will read the document from start to end like a detective novel (linear reading) or if he wants to find certain information as in a telephone directory or a reference manual.

In addition to that, the layout has to consider certain conventions, like the habits of the reader or "Corporate Design" rules that distinguish publications from different publishers. [1]

! → The main purpose of a layout is to make sure the reader finds the information he wants and is able to read and understand it easily. The structure, readability, and consistency of a document is more important than it's "beauty".[2]

The following "rules of thumb" will be valid for most applications:

Line spacing : The spacing between two lines should be larger than the spacing between two words to guide the eyes of the reader.

[1] Compare the layout of different daily papers or magazines like "page" or "invers".

[2] This is not always true for adverts that usually contain no information at all and, picture magazines where the beauty of the picture *is* the contents.

Line length : The length of a line – or when using multicolumn layout of a column – should be about 60 characters. When lines get longer they are more difficult to read and it is easier to go to the wrong line after finishing the current one. Increasing the linespacing may help a little. When lines get to short it is difficult to set them justified, and you will get lots of hyphenated words.

Page layout : Normal text pages should look the same throughout the document. Figures, tables and special pages like the index need not appear in the same layout but should take as much space as needed.

Margin notes : Margin notes are often more suitable than footnotes because they appear right next to the text they refer to. Special margin notes are the "attention sign" or the "dangerous bend" that guide the user to important parts of the text.

Headings and Footings : Headings and footings should make it easier for the reader to orient himself in the document. If you expect readers to copy single pages from the document they should contain information about the paper as a whole, just in case you need more information or want to cite the whole paper.

If you expect the document to change often (like software manuals), each page should contain a version information or at least a date.

4.2 Special note for technical descriptions

Let us compare three different layouts and check if they are usable for technical descriptions, user's guides and reference manuals. [3]

Plain TEX : The standard format use by plain TEX has the great disadvantage that the lines are much to long, which reduces the readability.

Standard-LATEX : The format used by LATEXs standard classes isn't ideal. The line length is correct, but that leaves us with a wide unused margin. The font used for section headings is too large. The Koma-Script classes improve this and offer many ways to configure the fonts. This is a good design for a paper one usually reads from start to end.

Reference Manual Style : A new design that appeared some years ago and is used in recent reference manuals, is much more suited for our purpose. [4]

- The text is printed in rather short lines in the right part of the page. This part is used for continuous reading.
- The wide left margin is used for headings and margin notes. Since you now have a wide margin it is easier to use long margin notes to supply additional information and to lead the reader to important parts of the documents. Please note that the margin is always on the left side thus two-sided printing does not look symmetrical. This is done on purpose, because the reader will always start reading at the left side, and with this layout section headers really "stand out". In a symmetrical layout, half the headers would be buried in the text.
- Figures and tables are either inside the text column, inside the margin or, if necessary, fill the whole page.

→ Section 5.6 Section5.6 describes how to implement such a layout in LATEX.

[3] This hint came from Paul Stiff, who teaches layout design at the University of Reading.
[4] The "PostScript Reference Manual" is one document that uses such a design.

5 How to change a LaTeX layout?

5.1 Advantages and Disadvantages of the text processing system LaTeX

Advantages: The big advantage of LaTeX is, that it implements a "generic" or "logical" design. This means that the author has to specify the *meaning* of special parts of the text like: headings, citations, lists, literature references, and so on. These logical definitions will be processed by the system and printed in the "correct" way. The meaning of *correct* is defined in the document class and additional packages.

The opposite of this is the "visual" design that most text processors use. Here the author has to know the correct way to set certain parts of the text and take care of the correct printing.

The logical design makes it easier for the author to write consistent documents (i. e., same font and fontsize for section headings of the same level, same layout for lists and enumerations, ...).

Disadvantages: The main disadvantage of LaTeX is that the author has only limited means to change the layout and that she has only four classes to choose from. This has changed a great deal with the appearance of the Script classes for LaTeX 2.09 in 1992, which in turn have been replaced by the even more improved Koma-Script classes for LaTeX 2_ε.

Another disadvantage is that LaTeX seems to be tied to the "Computer modern" font family. This is simply not true, at least not when you have a PostScript printer. Setting up different fonts isn't an easy task, but once they are installed they are as easy to use as the standard fonts.

But LaTeX is much more powerful and flexible: you can define an arbitrary design by changing the definitions in the class files or overwrite them by packages. This is easier than you may think and happens more often than you think. Many universities and publishers have their own LaTeX class but do not distribute it to the world.[5]

5.2 Input files and class files

According to the principle of separation of content and design, there are two kind of files:

- The content and the logical structure of a document are defined in the LaTeX input file.
- The design (layout) is defined in the class files and packages.

Which class and packages files a document will use is defined at the beginning of the input file. The `\documentclass` command selects the class and the `\usepackage` command specifies additional packages.

To generate a paper you need at least two files, the input file and a class file.

These two files represent the work of the author and designer as explained in the introduction. Even if the author and the designer are the same person, this has some advantages:

- Similar documents (that appear in a series) have the same layout because the layout is defined in a file of its own and not part of the document.
- You can print the same contents without much work in different layouts, e. g.. as an article for a magazine and as a chapter for a dissertation.

[5] You can find a generic `elsevier.cls` on CTAN which you can use to prepare article before submission. This will be replaced by the magazine-specific class at the publisher.

5.3 Class files and Packages

LaTeX supports a hierarchy of layout definitions.

- The first file processed is the class file that is specified inside the curly braces of the `\documentclass` command. This defines the kind of document you want to write.
- The optional argument of the `\documentclass` command inside the square brackets defines class options which select variants of the basic layout,such as different font sizes.
- The last step is reading the packages specified by the `\usepackage` command. This command again takes options to select the layout.
- You can change layout parameters in the input file, but this is discouraged because it violates the principle of separation of content and design.

There are some important differences between class and package files and "normal" input files:

- Class and package files should only contain definitions. They must not output text.
- The "at"-sign `@` is treated as a letter and therefore may appear in command names. Most internal commands of LaTeX contain an `@` to prevent the author from using them accidentally.
- The extension of the file is `.cls`, `.clo` or `.sty` instead of `.tex`.

5.4 Changing the layout, step by step

It is usually easier to change existing class files instead of writing a new one from scratch. In many cases it is even sufficient so replace some definitions and put them into a package instead of creating a new class.

Please note that you are *not* allowed to change the standard classes distributed with LaTeX. You *have* to change the name when you want to make changes. That is another reason to put small changes in packages.

5.4.1 Defining the differences between the desired and the available layout

The first step is to define the difference between the layout you have and the layout you want.

5.4.2 Finding the original definition

The next step is to find out where the original layout is defined. It is best to search the files in the following order:[6]

1. the LaTeX manual by Leslie Lamport,
2. the LaTeX documentations files `*.dtx` for the classes or packages
3. the LaTeX documentations files `*.dtx` for the kernel,
4. the TeXbook by Donald E.Knuth.

The files are usually documented quite well so you should be able to change things even if you don't understand everything.

[6] This hint came from Sue Brooks, who held a workshop for "LaTeX-Hacker" at the 1988 TeX-conference in Exeter.

5.4.3 Writing a new package file

The third step is to create a new package. You choose an appropriate name for the package (like `mysty`) and create a filename by adding the extension `.sty`.

This file will only contain the definitions you want to change or the new commands you want to define.

If you want to change definitions or certain parameters, the best way is to copy them from the original file and modify them according to your liking.

Defining new commands is easier when you find similar commands in the original files which you can change.

It is always a good idea to include the reason you wrote the package, the changes it makes and the new commands it defines in the file. You should include the date of the last change and the LaTeX version it works with, just in case some internal LaTeX commands you use will change.

When writing larger packages, it is an even better idea to use the `docstrip` program which is used to document the LaTeX 2_ε files. Thus you have your code and documentation in one file and it's easier to keep them from going out of sync.

5.4.4 Using the new package

To use the new package, you call it with the `\usepackage` command. This command executes the code of your package and changes the layout as desired.

Example:

```
\documentclass[11pt,twoside,a4paper]{article}
\usepackage{mysty} %<- This calls the package "mysty"
```

You shouldn't need to change anything else in your input file, unless you defined new commands or environments that are not available in standard LaTeX.

! → When you copy your input file to a different computer you have to include your new packages as well. Otherwise the document can't be processed.

5.5 A simple example (Equation numbers)

Let's assume that you want to write an article where the equations are numbered separately in every section. In the LaTeX manual you find a notice that the report class does something similar for every chapter.

Looking into the file `report.cls` you will find the following commands that deal with equation numbers:

```
\@addtoreset{equation}{chapter}
\def\theequation{\thechapter .\arabic{equation}}
% or in LaTeX2e since 1995/06/01:
\renewcommand\theequation{
            \thechapter.\@arabic\c@equation
                                }
```

You don't necessarily need to understand these two commands in detail.

Now you create an new file with the name `eqpersec.sty`[7] and copy the commands above into that file. After that you replace every occurrence of `chapter` with `section` and add some comments.

[7] Depending on the computer you are using the name may be different like `EQPERSEC_STY` on a CYBER running NOS/VE. But note that you must not use spaces in the filename.

```
% This is equation_per_section.sty
% Short name: eqpersec.sty
% Original file by Hubert Partl 1988
% Modified by Axel Kielhorn 1996/01/01
% to support LaTeX 1995/06/01 and later
%
% reset the equation counter at the start
% of a new section
%
\@addtoreset{equation}{section}
% Equationnumber = sectionnummer.equationnummer
% Use only one of the below
% depending on you LaTeX version
%
%\def\theequation{\thesection .\arabic{equation}}
% or in more recent versions of LaTeX
\renewcommand\theequation{
              \thesection.\@arabic\c@equation
                                }
```

Whenever you use a `\usepacakge{eqpersec}` command as in

```
\documentclass[11pt]{article}
\usepackage{eqpersec}
```

you will get equations numbered according to your conventions.

5.6 A more complex example (Reference Manual)

We want to create a layout similar to the one used in the *PostScript Reference Manual*, with a wide left margin for headings and margin notes and a small margin at the right and bottom.

5.6.1 Page layout

To define the new layout we use the commands described in the LaTeX manual. For full details see the file `refman.dtx`.

Horizontal: First we define two new names for length that we will use often:

`\fullwidth` is the width of the whole page minus a margin of 1inch on every side.

$$\texttt{fullwidth} = \texttt{paperwidth} - 2 \text{ inch}$$

From this the width of the text is calculated.

$$\texttt{textwidth} = \texttt{fullwidth} \times \texttt{textfraction}$$

`\leftmarginwidth` is the width of the left margin that will be used for headings and margin notes.

$$\texttt{leftmarginwidth} = \texttt{fullwidth} - \texttt{textwidth}$$

This is a little more difficult in reality because the lengths have to be rounded to full points and a possible two column layout – as used in the index – must be taken into consideration.

Vertical: The vertical layout is a little more difficult because you have to deal with the page header and footer.

$$\texttt{textheight} = \texttt{paperheight} - 2.5 \text{ inch}$$

The result of this calculation is rounded to full lines. Depending on the page style – headings or footings – it is shifted up or down by one line.

5.6.2 Section headings

The headings have to be modified to make them extend into the left margin.

In file `classes.dtx` we find the `\@startsection` command that defines the layout of the headings. Only parameters 4 to 6 are relevant for us: parameter 4 is the space above and parameter 5 the space below the section. The 6th parameter does the actual formatting.

This is the original definition:

```
\newcommand\section{\@startsection
             {section}{1}{\z@}%
             {-3.5ex plus -1ex minus -.2ex}%
             {2.3ex plus .2ex}%
             {\normalfont\Large\bfseries}}
```

The commands for sub- and subsubsections are similar. Note that the measures are all in `ex`, thus depending on the font size used.

We define a new command `\secshape` to format the headings. This command uses the whole width of the page for the heading. To discourage hyphenation of the heading we give it a high penalty. This still allows hyphenation when absolutely necessary.

```
\newcommand\secshape{%
               \leftskip=-\leftmarginwidth%
               \rightskip=\@flushglue%
               \hyphenpenalty=2000}
```

This command is inserted into the 6th parameter of `\@startsection`.

Since the headings now extend into the left margin, we can use a smaller font and reduce the space between the text and the heading. The new definition looks like the following:

```
\newcommand\section{\@startsection
             {section}{1}{\z@}%
             {-2ex plus -1ex minus -.2ex}%
             {0.5ex plus .2ex}%
             {\secshape\normalfont\large\bfseries}}
```

5.6.3 Setting the margin notes

The margin notes should always appear on the left side of the text. The normal layout puts them into the outer margin in twoside layout.

The file `latex.dtx` contains the definition of the `\@addmarginpar` command which is responsible for the margin notes. We don't have to understand the whole definition; the important part is the internal variable `\@tempcnta` that is either `\@ne` (1) when the note should appear on the right side of the text or `\m@ne` (-1) when it should appear on the left side.

This is done by the following lines:

```
\@tempcnta\@ne
\if@twocolumn
    \if@firstcolumn \@tempcnta\m@ne \fi
\else
  \if@mparswitch
     \ifodd\c@page \else\@tempcnta\m@ne \fi
  \fi
```