Plan for Independent Study -- Spring 2015

Terminology

I frequently use the term *visual element* in this proposal. It is a general, imprecise term. It is best described as a self-contained visual object -- a single polygon, an imported image, a line, a block of text, etc. As a rule of thumb, when I think of a visual element, I think of something that would have its own layer in a Photoshop document. Note that I may use words like *object* synonymously.

Objective

To create a piece of graphics processing software. The software would be intended to let the user perform two types of task:

1. Laying out visual elements on a page: traditionally, laying things out on a canvas is a laborious manual process that involves lots of small calculations and positional nudging. The program should make it as easy and quick as possible to lay out visual elements such as images, polygons, lines, blocks of text, etc. with pixel-perfect precision.

   Potential use cases of the software:
   - Designing PowerPoint slides
   - Designing/composing a poster, banner, or other cover art
   - Creating diagrams (e.g. flowcharts)

2. Algorithmically generating and modifying visual elements: the program should facilitate handling visual elements in an iterative way. This is to say that for basic operations on visual elements (rotation, transform, color shift, creation, deletion, etc.), the program should make it easy to iteratively apply such an operation to a set of visual elements.

The program should have the following features:

1. A programmatic way to modify visual elements on a canvas: a domain-specific language (DSL) and a textual interface that permit the user to modify single visual elements or groups of visual elements in a quick, intuitive, and consistent way.

2. A graphical user interface that will allow the user to modify visual elements on a canvas in the click-and-drag way common to most graphics processing programs.

3. The graphical and textual interfaces should be reactive, such that any change in either is propagated (synchronised) across both interfaces. The two interfaces should also

integrate to make the workflow as smooth and rapid as possible.

4. The textual interface should lead to the creation of files that allow for easy templating: a layout created with certain visual elements should be adaptable, such that visual elements may be mass-replaced while preserving the general layout. (Going further: including a mode for automatically adapting layouts to templates would be good.)

Thus, what I want to create is a *designer's tool*, to be used when working with low-complexity visual content.

Remark: I think that the usefulness of graphics processing software rapidly diminishes when attempting to use the software for a non-intended purpose. Doing graphic design well involves -- to no small extent -- choosing the right tool for the right job. As such, I think that more specialization in the intended purpose of the software is better than less. To be absolutely clear, below are some common tasks for which this program is **not** intended:

- General graphics processing in the vein of Photoshop
- Image retouching/fine editing
- Digital painting
- (Large-scale or high-complexity) data visualization
- Typesetting/textual image creation
- Analytic image processing (fourier analysis, edge-detection, region etc.)

What I'm Going To Do

What I will do in the course of this project can be divided into three categories. They are in the order in which I will work on them:

1. Things I will investigate, read, and learn about
2. Things I will think and hope to develop original ideas about
3. Things I will build

The third point -- what I intend to build -- is summarized reasonably by the preceding **Objective** section. Naturally, what I intend to build will change as I make progress in the first two points, i.e. as I learn more about this specific domain and think about particular engineering challenges.

The second and first points I'll discuss in the two following sections.

Things I Will Investigate, Read, and Learn About

The four following topics strike me as the major areas of investigation:

1. Compilers and implementation of programming languages. The program will invariably contain a small DSL. I'll have to implement this language, which is something I don't really have any experience with thus far. I'll have to learn about lexers, parsers, abstract syntax trees, etc. I've gathered that the Dragon Book is a standard reference.

2. Constraint-oriented programming and direct manipulation. You've started me off with a few papers. I'll try to read more literature in the area.

3. I'll need to survey existing tools in this field (i.e. that have both textual and graphical modes for image manipulation, or in which graphical manipulation yields textual output), examine their approaches, solved problems, and challenges.

4. I will need to dig into the literature and into the user-experience (UX) industry to research workflow optimization: a key goal of this program is to provide a good user experience when doing layout. A really tricky aspect here will be that the graphical and textual editing modes need to be *well-integrated*: there is potential for creating a workflow that uses the best of both so the user can get things done as effectively as possible. On the other hand, poorly-integrated editing modes would be comparatively disastrous.

5. I need to research and devise a method to synchronise the graphical and textual representations on each timestep.

Things I Will Think and Hope to Develop Original Ideas About

One of the things I like about this project is that there isn't anything quite like what I'm proposing on the market, which opens up space for experimentation. Following are some of the things I'll need to think about, i.e. problems I'll have to solve:

1 There's a (qualitative) multivariate-optimization problem to this project. Several of its properties can be placed on spectrums, such as:

- Design workflow: text-driven vs. graphically-driven
- Language syntax: geared to groups of elements vs. single elements
- Manual vs. automatic setting of layout parameters
- Language syntax: geared to object creation vs. object modification
- Textual representation: short, complex vs. long, simple code

In creating this software, there are many of these design decisions that involve choosing the extent to which I want to follow some particular paradigm. This will require some careful thought/planning/research.

2 There's a ton of design decisions related to the language: how do I create a language that supports both single object manipulation and group object manipulation equally well? How should I even represent visual elements? What should the capacities of the language be (i.e. how much abstraction should it allow for)? Above all, how do I ensure that no sequence of manipulations leaves the user with an unnavigable, messy textual representation?

Open-Ended Parts

There are some parts of this project on which I'm currently not yet decided. I should decide these as I make the above design decisions. Some examples of important decisions I'll have to make:

1. Should I aim to include a mode for automatic template adoption?
2. Should the software run in the browser, or be a desktop application?

Goals, Milestones, Other Thoughts

Though I don't consider these to be set in stone, I'll need to first do the research phase of this project. I don't expect this to take terribly long -- perhaps one or two weeks. I want to mostly familiarize myself with the current state-of-the-art in relevant domains and read up on compilers so I can build my language.

Next, I'll need to think about the design issues related to the software and to the language. I expect this to be difficult -- the design has to be really good, else it'll fail in normal use. Making a really good design is difficult, as any UX/interface designer can tell you. I'll probably have to go through lots of drafts and vet them against one another, iteratively develop concepts, etc.

This might take two weeks, maybe even three depending on the extent to which I can bounce ideas off other people. This is definitely the phase of the project on which I'll need the most advice.

Finally, I'll have to build the software. Depending on the ideas I have for the language and for the interface, this will probably take quite long, and I'll end up working on it well into the summer holiday, but that's fine. I should be able to make significant headway into this phase of the project within the rest of the quarter. The development speed of this part of the project is also a function of the platform on which I choose to build the project: I expect that if I choose to make this an application that runs in the browser, then it might take me longer as I haven't had much JS development experience. (I'm tentatively considering using Elm for this.)