SKAN Language Proposal

S*cott Lukas (System Architect)*
K*atie Brady (Tester)*
A*lex Cohen (Manager)*
N*ick Cancar (Language Guru)*

Motivation

The motivation for our project is to create a language that is designed to manipulate images. We want to create a dynamically typed language with an imperative design that allows programs to operate on multidimensional arrays and matrices in order to alter, combine, or filter images. We want to combine functionality and efficiency so that the programmer has the ability to create their own or use a built-in function to efficiently alter and process an image.

A key example of this would be taking the double gradient of all the pixel intensities in an x-ray picture and adding that to the original image in order to exaggerate the edges to create a sharper rendering.

In terms of syntax, we want to combine elements of C, Java, and Python to increase the readability and clarity of programs. SKAN will also be weakly and dynamically typed, which will allow for interesting operations to be performed on images. Rather than stop the user from performing an operation on an image, this will allow them to try unconventional functions that alter images in various ways.

Paradigms and Features

Skan will be an imperative language that is weakly, dynamically typed and statically scoped. It will also have strict evaluation order and will be modeled after a combination of C, Java, and Python.

Skan will have higher order functions and anonymous functions to allow functions to be easily used, called, and returned in other functions. Furthermore, as Skan will be dealing with pictures from a variety of file locations, we will need to implement I/O capability in order to load/write images to/from locally sourced file locations. Skan will do this by utilizing OCaml's I/O system to read/write files to channels. An example of how we could code I/O capabilities:

```
# let ic = open_in "/etc/motd" and oc = open_out "/tmp/feh";;
```

Resource: https://www2.lib.uchicago.edu/keith/ocaml-class/io.html

"Hello World" Program

```
main {
      Image cute_dog = load_image("./mycutedog.png");
      print_image(cute_dog);
}
```

"Language in One Slide" Program

The example program applies several imaging processing techniques seamlessly to an image and displays the output image. It initially reads in the `lion` image and brightens it by applying a brightening function 3 times. It then reads and applies a mask image to the `lion` image, which removes portions of content from the image. Lastly, it writes and saves the `lion_new` image to a file before printing it.

```
main {
      Image lion = load_image("./lion.png");

      $$ function that brightens each pixel by 5. Weakly typed
$$
      function increase x = {return x+5};
      function apply_3(f, x) = {
            return  f(f(f(x)));
      }

      $$ brighten image by 5 three times $$
      lion_new = apply_3(increase, lion);

      Image mask = load_image("./mask.png");
      lion = lion * mask; $$ apply mask image to lion image $$

      write_image(lion_new, "./lion_new.png"); $$ write image to
      lion_new.png file $$

      print_image(lion_new);
}
```