# CS 4115 Project Proposal

## Dmitriy Gromov, FeiFei Zhong, Yilei Wang, Peter Ye, Joe Lee

### Programming Language Description:

We are planning on implementing a programming language for the purpose of creating and manipulating ASCII Art. ASCII Art is a form of drawing pictures using only the characters defined by the ASCII character set. The goal of the language will be to provide simple mechanisms for dealing with ASCII images. It will allow users to create images by adding and manipulating each character individually or by loading in an existing image and converting that to an ASCII format and then dealing with that. The language will provide a way to make functions or modules for the purposes of reuse. This will enable eventual library building allowing the programmers to create more interesting constructs and images using this language.

There will be several basic data types for this language. The most important of which is one single character. We will also represent integers and some decimal numbers for convenience in drawing. For ease of use in data manipulation, this language will also support loops so that one can iterate through the entire image and address each character easily.

### Why is this interesting?

Drawing ASCII art is a fairly difficult task. One has to know where exactly something should be put to look correct, how many spaces there should be between it and the next thing and which characters to use to display what. Sometimes one can find ASCII art representation of other images on the Internet that seemed to take a long time to draw. This language would be a solution to this problem of drawing difficulty. Using this programming language a user can simply input an image of what they want and receive an in memory ASCII image which they are now free to manipulate or just save if they think it is pretty.

While drawing ASCII art, one generally only has the option of varying color intensity. Color intensity in this case is defined by how much of the space that a character takes up is actually taken up by the character. As an example, ‘@’ would describe an intense color, while ‘,’ would describe a not intense color. With this in mind the language will allow users to not worry about which character to use but only its intensity. However, if a user feels the need to use a specific character they will have the option of overwriting the default mapping for the color intensity. Since many characters have similar intensities we will not provide a mapping of all characters to intensities by default. If desired, the user will be able to increase the granularity of intensities as needed.

For the image manipulation the user will have the complete freedom to change any character in that image or change all characters of a given intensity to a different one. This way, an image's color can be "inverted" or simply darkened to conform to the user's tastes. Furthermore, the user will have the ability to apply simple transformations to a range of characters (e.g. shift several cells up and to the left).

One of the main uses for this programing language will naturally be to convert images from other formats into an ASCII format. However, using the various abilities available to the programmer, he/she should quickly be able to set up functionality to draw various shapes and constructs and manually create images.

### Language Details:

#### Basic Types Supported:

* Boolean – A true or false value
* Int – Some 32 bit integer
* Char – A character in the ASCII table

#### Complex Types:

* Array 1D, 2D – 1 or 2 dimensional collection of any type.
* Struct – A grouping of instances of other types to be treated as one type.
* Function references – A reference to a function that can be used as a type.

#### Branching Statements:

**If, ElseIf, Else**

Each statement will have a block of code associated with it. "If” will test some condition (in the form of a Boolean) and execute its block of code if the condition is true. If it is not, it will continue to the ElseIf that will also test a condition and execute its own block if the condition is met. If it is not, it will continue to the next ElseIf statement if one is available or go to the Else if it exists. If the execution gets to the Else, the code there will be executed automatically. If there is no ElseIf but there is an Else the Else block will be executed if the If condition is not met. ElseIF and Else cannot exist without an If.

#### Loops:

**For**

This is a loop with a code block, an associated condition, and the condition variable. In this statement, the user will be able to specify the initial state of the condition variable, and how it should be updated at the end of each execution of the loop. Until the specified condition is met, it will continue to execute the code block.

#### Functions:

A user will be able to create functions that will act as a blocks of code that can be called when desired. They will accept a list of input parameters and return some value at the end of their execution. It will be possible to specify no input parameters or return nothing if this is desired. Recursive functions will be allowed.

#### Variables:

A variable is a named instance of some type in this language. Its scope of existence is limited to either the global level where all code in a program can see it or the function level where it is limited to only the function in which it is defined. Names of variables in the same scope must be unique. A variable in a function can have the same name as a global variable but referring to it from inside of the function will access the variable as defined in the function and not as defined globally.

#### Operators:

* Basic Arithmetic Operators: +, -, \*, /, %
* Equality ==, ~=
* Assignment =

#### Program Linking and Execution:

* **Import keyword**: Will allow you to specify another source file, which will be appended to the current one at linking time.
* If defined, programs will start at a function called **main**. There can be at most one main defined in the program. If main is not defined the program will execute beginning to end as the code is written.

### Functionality:

Reading in an image File:

OCAML image libraries will read in an image, convert it to black and white, and compress the image if it is deemed too big. We will then apply Floyd–Steinberg dithering to the created image to effectively normalize the points we have and see which one's we can get rid of. Then, for each character-sized block of the image we will determine the intensity of the pixels there and create a character to represent that. These characters will get stored in a 2D array and returned to the user for manipulation.