Language Specifications for 1-800-Got-Photos

Angela Gui

December 2022

1 Introduction

1-800-Got-Photos is a language that allows users to modify a raster image's file extension and size easily. It supports image format conversions between Jpeg, Png, Bmp, Gif, Pbm, Tiff, Tga and Webp, as well as basic uniform resizing operations. The main reason behind the construction of the language is the need for a quick and easy way to modify an image file format from the command line. When users upload an image to a website, they could encounter file format and/or size constraints; instead of dealing with the inconveniences of having to find image file format converters online, they could now depend on 1-800-Got-Photos as a handy tool to convert and resize their image.

2 Design Principles

The main guiding principles in 1-800-Got-Photos's language design are simplicity and usability. As I want my language to be easy to use even for users who are not tech savvy, I modeled my syntax as intuitive as possible for an average user who understands English, such as having function names that are syntactically close to English (eg. parsing in/parsing out images using the Input and Output functions).

3 Examples

Example 1:

```
Input "Cat.jpeg"
```

Run the example using "dotnet run example-1.shop". This example loads in an raster image file with its name and path specified by "Cat.jpeg". If the program execution is successful, the command line output should be:

```
Your image Cat.jpeg is loaded

Program Success!
```

Example 2:

```
Input "Cat.jpeg"
OutPng "Cat.png"
OutGif "Cat.gif"
```

Run the example using ”dotnet run example-2.shop”. This example loads in an raster image file with its name and path specified by ”Cat.jpeg”, convert the loaded image file format into png and generate a png image file with its name and path specified by ”Cat.png”, convert the loaded image file format into gif and generate a gif image file with its name and path specified by ”Cat.gif”. If the program execution is successful, it will create two new image files Cat.png and Cat.gif in the same directory of the project. The command line output should be:

```
Your image Cat.jpeg is loaded

Your image Cat.png is saved

Your image Cat.gif is saved

Program Success!
```

Example 3:

```
Info "Cat.png"
Input "Cat.png"
MultSize 4
DivSize 2
OutWebp "NCat.webp"
Info "NCat.webp"
```

Run the example using ”dotnet run example-2.shop”. This example prints out the format, height and width about a raster image with its name and path specified by ”Cat.png”, loads in the same image, multiply its width and height by 4, divide its width and height by 2, convert the loaded image file format into webp and generate a webp image file with its name and path specified by ”NCat.webp”, and prints out information about the image ”NCat.webp”. If the program execution is successful, it will create one new image file NCat.webp in the same directory of the project. The command line output should be:

```
Image Format: Png
Image Height: 480px
Image Width: 481px

Your image Cat.png is loaded

Image size has been multiplied by 4

Image size has been divided by 2

Your image NCat.webp is saved

Image Format: WebP
Image Height: 960px
Image Width: 962px

Program Success!
```

4 Formal Syntax

$\langle expr \rangle := \langle Input \rangle$
$\langle Output \rangle$
$\langle OutBmp \rangle$
$\langle OutGif \rangle$
$\langle OutJpeg \rangle$
$\langle OutPbm \rangle$
$\langle OutPng \rangle$
$\langle OutTiff \rangle$
$\langle OutTga \rangle$
$\langle OutWebp \rangle$
$\langle Info \rangle$
$\langle MultSize \rangle$
$\langle DivSize \rangle$
$\langle LoadedImage \rangle$
$\langle Input \rangle ::= Input \langle filename \rangle$
$\langle Output \rangle ::= Output \langle filename \rangle$
$\langle OutBmp \rangle ::= OutBmp \langle filename \rangle$
$\langle OutGif \rangle ::= OutGif \langle filename \rangle$
$\langle OutJpeg \rangle ::= OutJpeg \langle filename \rangle$
$\langle OutPbm \rangle ::= OutPbm \langle filename \rangle$
$\langle OutPng \rangle ::= OutPng \langle filename \rangle$
$\langle OutTiff \rangle ::= OutTiff \langle filename \rangle$
$\langle OutTga \rangle ::= OutTga \langle filename \rangle$
$\langle OutWebp \rangle ::= OutWebp \langle filename \rangle$
$\langle Info \rangle ::= Info \langle filename \rangle$
$\langle MultSize \rangle ::= MultSize \langle size \rangle$
$\langle DivSize \rangle ::= DivSize \langle size \rangle$

$\langle filename \rangle ::= " \langle String \rangle "$
$\langle size \rangle ::= \langle String \rangle$
$\langle LoadedImage \rangle ::= \langle Image \rangle$

5 Semantics

Input is a String type that applies the Input function to the image whose name and path is specified by the filename.
Output is a String type that applies the Output function to the image whose name and path is specified by the filename.
OutBmp is a String type that applies the OutBmp function to the image whose name and path is specified by the filename.
OutGif is a String type that applies the OutGif function to the image whose name and path is specified by the filename.
OutJpeg is a String type that applies the OutJpeg function to the image whose name and path is specified by the filename.
OutPbm is a String type that applies the OutPbm function to the image whose name and path is specified by the filename.
OutPng is a String type that applies the OutPng function to the image whose name and path is specified by the filename.
OutTiff is a String type that applies the OutTiff function to the image whose name and path is specified by the filename.
OutTga is a String type that applies the OutTga function to the image whose name and path is specified by the filename.
OutWebp is a String type that applies the OutWebp function to the image whose name and path is specified by the filename.
Info is an application that applies the Info function to the image whose name and path is specified by the filename.
MultSize is a String type that applies the MultSize function to the image whose name and path is specified by the filename.

DivSize is a String type that applies the DivSize function to the image whose name and path is specified by the filename. LoadedImage is a terminal of Image type. filename is a terminal of String type. size is a terminal of String type.

6 Remaining Work

The next step in the project would be creating variable and variable assignments so that the language could take in multiple images at the same time. Currently, 1-800-Got-Photos only supports conversion and output of one image at a time; subsequent image load ins will overwrite the previous input, and if Output is called, the program will always generate the new image based on the last image that is loaded in using Input. Implementing variables would allow users to convert, resize, and keep track of multiple images at the same time. In the earlier versions of the language, there were ideas about adding features that allow users to flip and recolor certain pixels of the entire image, however these were discarded due to concerns about usability as well as the complexities of implementations.