Report - DesignerTools

Omar Armando Neira Ordoñez[1]

[1]Computers Science III – Distrital Francisco Jose de Caldas University

oaneirao@udistrital.edu.co

Abstract. *This document presents the development of a specific purpose language for graphical designers. The language is supported by Python and multiple libraries to perform various tasks (mainly by Pillow). The grammar is in English to make it accessible to everyone, especially those who do not know how to program. The language is designed to automate tasks such as enhancing images, applying filters, and more.*

1. Introduction

This specific purpose languaje is a languaje made for graphical designers, it is intended to make the work of the designer easier, it will be supported on python using multiple libraries so it can do multiple things, the grammar will be on English so everyone can use it, it is intended to be used by people that doesn't know how to program.

2. Justification

This languaje was made thinking on graphical designers that need to work with a lot of images, so they can simply use this languaje to automatize tasks; for example they need to enhance a lot of images to a specific brightness, or apply a filter on them, they only need to select all those images with the languaje and tell it to auto apply it for them, a thing that manually can be really tedious and time consuming to make, but with the help of the languaje can be made really fast and easy.

Table 1. Table of keywords

Lexem	Token	Description
START	Keyword	Defines the start of the code.
END	Keyword	Defines the end of the code.
START_OP	Keyword	Defines the start of a block of code.
END_OP	Keyword	Defines the end of a block of code.
TO_IMAGE	Keyword	Defines the location of an image.
TO_FOLDER	Keyword	Defines the location of a folder of images to be use.
sepia	Keyword	Constant filter vintage.
negative	Keyword	Constant filter, inverts colors.
black_white	Keyword	Constant filter, turns colors to black and white.
dark	Keyword	Constant filter, lowers brightness.
red	Keyword	Constant filter, turns colors into red.
green	Keyword	Constant filter, turns colors into green.

Continues in the next page

Lexem	Token	Description
blue	Keyword	Constant filter, turns colors into blue.
blur	Keyword	Constant filter, blurs an image.
contour	Keyword	Constant filter, affect the shadows and glows.
detail	Keyword	Constant filter, improves sharpness and brings out detail.
edge	Keyword	Constant filter, sharpens edges of images.
find_edges	Keyword	Constant filter, shows mostly the edges of an image.
smooth	Keyword	Constant filter, blends the pixels along the edge.
sharpen	Keyword	Constant filter, focuses soft edges to increase clarity.
grayscale	Keyword	Constant filter, turns colors into a greyscale.
emboss	Keyword	Constant filter, darkens high-contrast edges.
blur_gaussian	Keyword	Special filter, applies a blur depending on a float.
brightness	Keyword	Special enhance, rises brightness depending on a float.
contrast	Keyword	Special enhance, rises contrast depending on a float.
color_enhance	Keyword	Special enhance, rises color saturation depending on a float.
definition	Keyword	Special enhance, rises definition depending on a float.
flip_horizontally	Keyword	Constant transform, flips the image horizontally.
flip_vertically	Keyword	Constant transform, flips the image vertically.
rotate	Keyword	Special transform, rotates depending on a integer angle (float).

3. Lexemes and Tokens

Table 2. Table of lexemes and tokens

Lexem	Token	Description
APPLY_FILTER	Operator	Applies predefined filters to img/folder.
APPLY_TRANSFORM	Operator	Applies predefined transformations to img/folder.
APPLY_ENHANCE	Operator	Applies predefined enhancements to img/folder.
"(" and ")"	Special char	Defines the content of a function

4. Generative grammar

This grammar will help to validate if the syntax is well written.
$Note$: The grammar $image\_url$ and $folder\_url$ are the respective url to edit, this has to be placed on the $common/img\_resources/$ folder of Designer Tools.

$\langle S \rangle$ $\Rightarrow$ "START" $\langle block \rangle$ "END"

$\langle block \rangle$ $\Rightarrow$ "START_OP" $\langle statement \rangle$ "END_OP" $\langle block \rangle$ | λ

$\langle statement \rangle$ $\Rightarrow \langle image \rangle \langle operation\_type \rangle \langle operation \rangle$
| $\langle folder \rangle \langle operation\_type \rangle \langle operation \rangle$

$\langle operation\_type \rangle$ $\Rightarrow$ "APPLY_FILTER" | "APPLY_TRANSFORM"
| "APPLY_ENHANCE"

$\langle operation \rangle$ $\Rightarrow \langle filter \rangle$ | $\langle s\_filter \rangle$ | $\langle transform \rangle$ | $\langle s\_transform \rangle$
| $\langle s\_enhance \rangle$

$\langle filter \rangle$ $\Rightarrow$ "sepia" | "negative" | "black_white" | "dark" | "red" | "green"
| "blue" | "blur" | "contour" | "detail" | "edge" | "find_edges"
| "smooth" | "sharpen" | "grayscale" | "emboss"

$\langle s\_filter \rangle$ $\Rightarrow$ "blur_gaussian " $\langle number \rangle$

$\langle transform \rangle$ $\Rightarrow$ "flip_horizontally" | "flip_vertically" | "rotate" | "crop"

$\langle s\_transform \rangle$ $\Rightarrow$ "rotate " $\langle number \rangle$ | "crop " $\langle number \rangle$

$\langle s\_enhance \rangle$ $\Rightarrow$ "brightness " $\langle number \rangle$ | "contrast " $\langle number \rangle$
| "color_enhance " $\langle number \rangle$ | "definition " $\langle number \rangle$

$\langle number \rangle$ $\Rightarrow$ any real number including negatives

$\langle image \rangle$ $\Rightarrow$ "TO_IMAGE ("$image\_url$")"

$\langle image \rangle$ $\Rightarrow$ "TO_FOLDER ("$folder\_url$")"

5. Results

5.1. Programming

The proposed solution to the problem was implemented in Python (3.13.0) language with virtual enviorenments, the main library for this project is $Pillow$ (PyPI) other libraries used are re (for regular expressions on the input_analizer), os (operative system to control some funcions such as opening files or using directories), using VSCode as the IDE. The code was developed in a modular way, with the use of classes and functions, to facilitate the understanding and maintenance of the code. The code is available in the following repository: https://github.com/OmarNeira/ComputersScienceIIIOmarNeira/tree/main/FinalProyect/DesignerTools.
The code is divided into the following parts:

- **main.py**: This file connects the compilator with the text written by the user.
- **comp.py**: This file contains the Compiler class, which is responsible for compiling the input text and applying the operations to the images.
- **input_analizer**: This group of files contains the classes responsible for analyzing the lexical, the sintaxis and the semantic to generate a valid token - value output.
- **features**: This group of files contains the classes responsible for applying the operations to the images. Each feature has its own control class and can have some child classes as default (pillow default functions), special (functions that needs more parameters) and others (functions defined by me, the programmer).

- **paths.py**: This file contains the functions to use paths to the images and folders that will be used in the project.

5.2. Designer tools use

First, we need to locate the folder called "common/img_resources/" (inside Designer Tools folders), it must contain all the data that you want to modify.
Next as an example we will use the following images:

Figure 1. image.jpg

Figure 2. design.png

The following example shows the result of the grammar applied to a text that works on Designer Tools and that will be applied to the images.

Algoritmo 1. Designer Tools example of use

```
1  from comp import Compiler
2
3  # ========== Example usage ========== #
4  def example(compiler_: Compiler):
5  """This␣function␣is␣an␣example
6  of␣how␣to␣use␣the␣compiler."""
7  input_text = r"""
8  START
9  START_OP
10 ␣␣␣␣TO_IMAGE␣(image.jpg)
```

```
    APPLY_FILTER sepia
END_OP
START_OP
    TO_FOLDER (example)
    APPLY_ENHANCE contrast 8
END_OP
START_OP
    TO_IMAGE (example/design.png)
    APPLY_TRANSFORM rotate 180
END_OP
END
"""
compiler_.compile(input_text)

if __name__ == '__main__':
    compiler = Compiler()
    example(compiler)
```

The result of the previous code is to apply the filter sepia to the image.jpg, that is located at the "common/img_export/" folder, with this result:

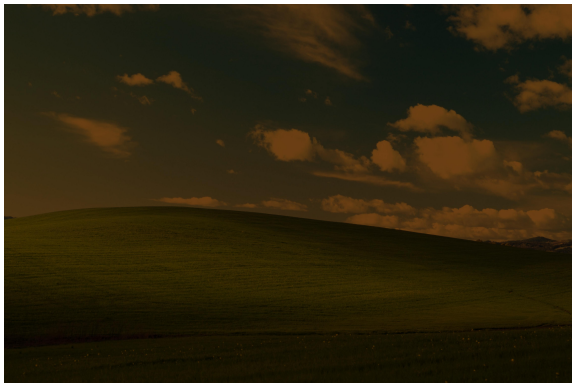

Figure 3. image.jpg with sepia filter

Then apply the contrast enhancement to the images at the "example" folder located at the "common/img_export/example", with this result:

Figure 4. design.png with contrast enhancement

And finally rotate the image "design.png" located at the "example" folder 180 degrees.

Figure 5. design.png rotated 180 degrees

6. Conclusions

The develop of a specific pourpose languaje is a hard challenge, a lot of things has to be considered, such as the sintaxis, the semantic, the grammar, the operations, features, etc. Making first the design is really important because it gives an orientation to the development, thanks to identificate the pourpose, the lexems, tokens and keywords it became easier to develop the grammar and the operations for the languaje.

Thanks to pillow library, the operations were easy to implement, the main challenge was to make the operations work with the compiler, the compiler is able to "read" the text, analyze it, and apply the operations to the images translating it to Python.

The objective of the project was successfully achieved, the Designer Tools project is able to apply operations to a lot of images in a simple way, the user only has to write the text with the operations and the compiler will do the rest, making the graphical designer life easier.

Now Designer Tools can be expanded to have more features thanks to the way that the code was developed, it can be used by other developers to add more features to the project.