University of Puerto Rico Mayaguez Campus
Electrical and Computer Engineering Department

# “Animate” Programming Language
**Phase 3: Final Report**

Programming Languages - ICOM 4009 Team
Wilson Rivera Martín J. Rivera Rosa
February 15, 2016 Wilfredo
Montanez Jose
Rodriguez Piña

Illary Lopes Aulet

## 1. Introduction

Drawing has been a pastime for people throughout decades, it being a doodle on a notepad or a masterpiece presented on a canvas. Some with more talent or creativity than others but all with the ability to portray some type of feeling or thought. “Animate” is a programing language that allows even the most inexperienced artist to doodle and create their drawings easily as well as make animated mock up stages and scenarios through the use of basic commands that present shapes and colors. Not only can a combination of commands translate to a drawing but movement can be incorporated to objects within the drawing creating a dynamic amination.

“Animate” like any programing language is composed of fundamental aspects such as syntax, semantics, statements and variable components. Designed in a simple non complicated manner that allows all types of people to be able to use it. Users will write their drawing commands in a text editor therefore creating scripts what will later on be parsed and translated to Python code to be  compiled and ran, presenting the drawing animation to the user as a final product.

## 2. Language Tutorial

The following is a step by step tutorial on how to use “Animate” to create the image below (Figure 1):

Figure 1

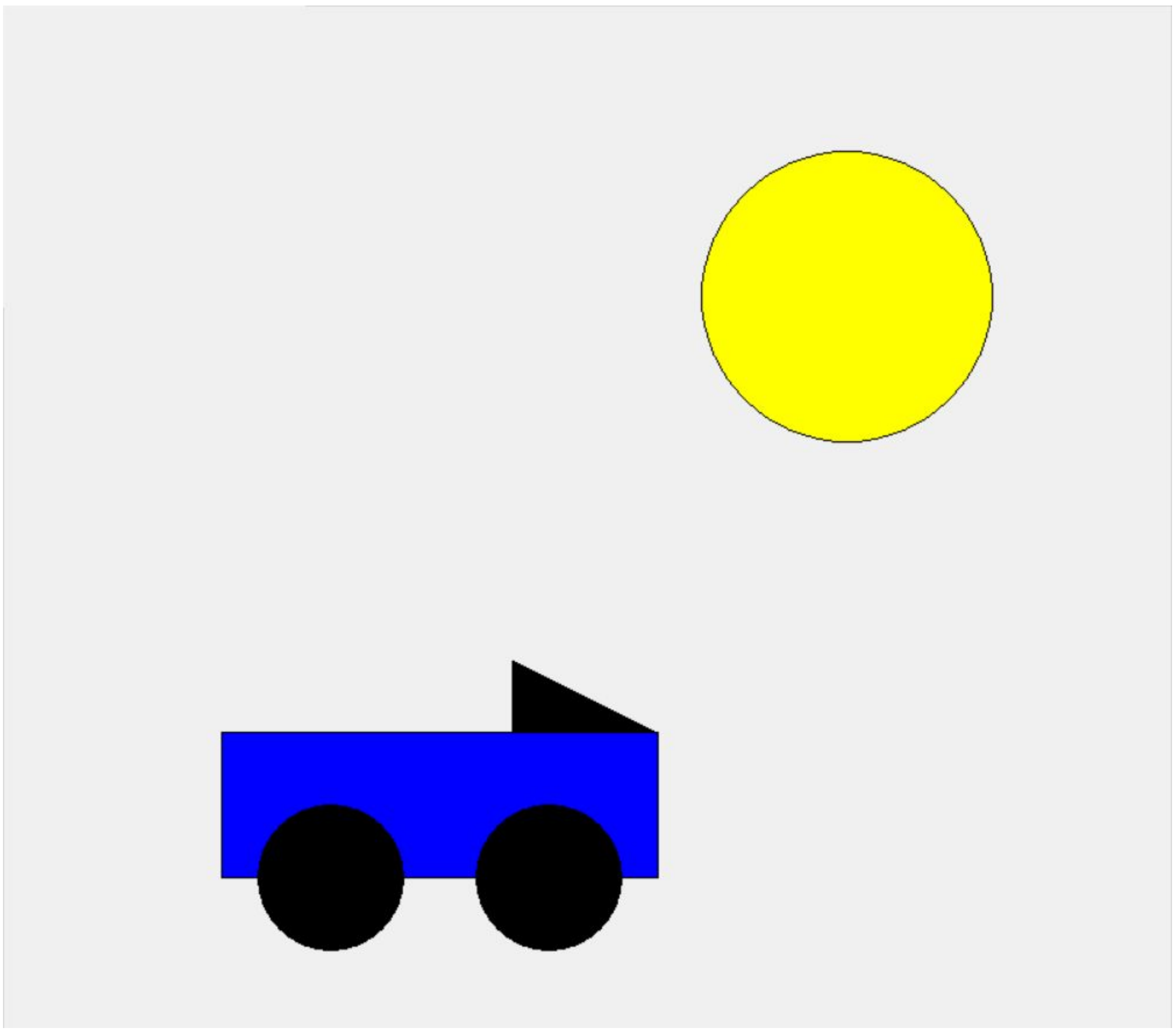

Figure1: Drawing created with “Animate” Language.

**Step1:** Download animate from the github repository: https://github.com/illary/Animate and open the file in your IDE.

***Note:** Assuming that user has installed Python 2.7, has installed all dependencies and is using an Integrated Development Environment (IDE) such as PyCharm.

**Step 2:** Proceed to run Animate_Parser file within the IDE .

**Step 3:** In console window create a frame on which to present your drawing by using the following command:

```
createframe (800, 800)
```

**Step 4:** Once the frame is created you may proceed to add shapes and colors to your drawing. Write the following commands in your console window:

```
blue rectangle (100, 500, 400, 600)
black circle (175 ,600, 50)
black circle (325, 600, 50)
black triangle (300, 500, 400, 500, 300, 450)
yellow circle (530, 200, 100)
```

**Step 5:** Finally, to perceive movement in your drawing write the following command:

```
animate (1, 0, 50, 200)
```

Upon completion of the stated above you should have a drawing similar to the one presented in Figure 1 and you are now ready to use “Animate” to create your own masterpiece. If you get stuck or have questions on what other shapes and parameters should be used within “Animate language” refer to the Language Reference Manual which has detailed instructions of language syntax.

## 3. Language Reference Manual

The following are the commands and instructions used in “Animate” programming language:

**Frame:**

- ➔ createFrame(width,height,background)
  - ◆ Creates the frame in which the graphics will be drawn. The width, height and background color are specified within the constructor.

***Note**: All animation should begin with the creation of a frame. Once frame is created user will receive the following message: *“Frame Created”.* If frame has not been created and user tries to create a shape the following error message will be presented: *"A frame has not been create".*

**Shapes:**

- ➔ Color Identifier Rectangle(x0,y0,width,height)
  - ◆ Creates a rectangle with a given color at the coordinates x0,y0 using the top left corner as a reference with a specified width and height.

- ➔ Color Identifier Circle(x0,y0,radius)
  - ◆ Creates a circle with the given color at the coordinates x0,y0 using the center as a reference with a specified radius.

- ➔ Color Identifier Triangle(x0,y0,x1,y1,x2,y2)
  - ◆ Creates a triangle with the given color and point coordinates.

**Movement:**

- ➔ Animate (deltaX,deltaY,iterations, sec)
  - ◆ Animates the objects in the frame by moving them deltaX and deltaY pixels. The image will be updated the amount of milliseconds stated in the parameter “sec”.

**Image:**

- ➔ loadImage(x0,y0,imageName)
  - ◆ Load an image into a frame at coordinates x0,y0.

**Available colors:**

- ➔ White
- ➔ Black
- ➔ Red
- ➔ Green
- ➔ Blue
- ➔ Cyan
- ➔ Yellow

➔ Magenta

***Note:** Shape will be set to white as default if user sets a color that is not stated above and the following message will be presented: *"Color is not available, setting color to white".*

**Reserved Words:**

- ➔ createframe
- ➔ rectangle
- ➔ circle
- ➔ triangle
- ➔ animate
- ➔ image

## 4. Language Development

### Translator architecture

Figure 2

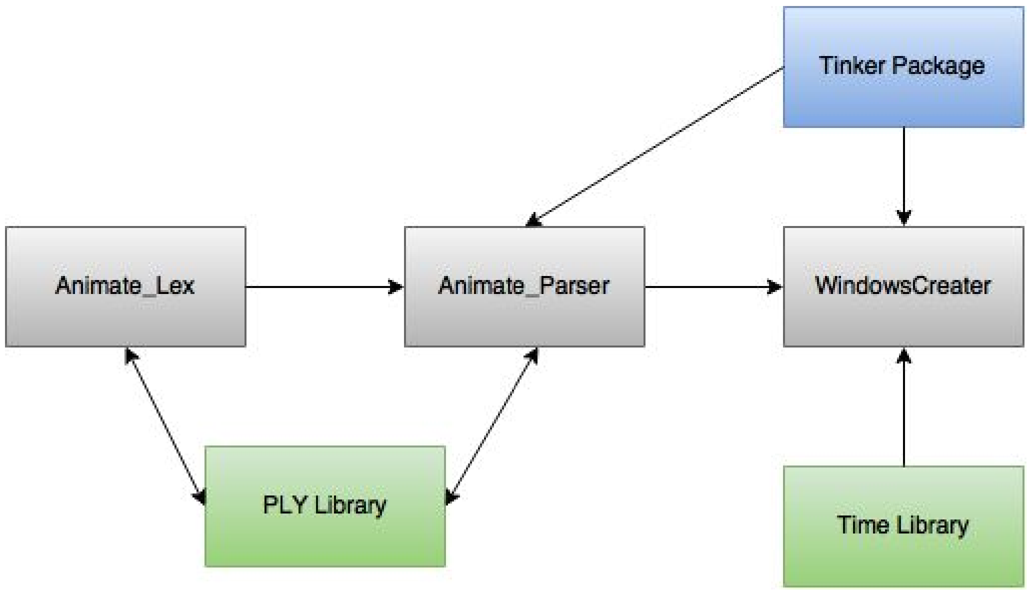

Figure 2: Diagram of “Animate” language translator architecture

### Interfaces between modules

The translator architecture seen in Figure 2 is composed of 3 main modules; Animate_lex, Animate_parser and Window Creator. Animate parser being the main focus of the programming language since within this module the main program is ran, and the user input commands are received. User input is then parsed by Yacc using the PLY library which uses python functions stated by the developer. These functions define grammar characteristics of the language. Yacc parser must match the functions found in Animate_parser module with the code entered by the user through the consideration of specific tokens defined in the Animate_Lex module. These tokens are composed of token name and regular expression needed to match and compare user input within the Animate language. Finally the WindowsCreater module is called to execute input instruction using specific python functions.

In addition to the three main modules, “Animate” language uses the following libraries, packages and modules:

**PLY** - is a parsing tool written purely in Python which re-implements the Lex and Yacc originally found in C language. It provides excellent debugging and error reporting facilities and it uses LR parsing to incorporate large grammars. Both Animate_lex and Animate_Parser use this library.

**Tkinter** - is a Graphical User Interface (GUI) package for python used in the WindowCreater module to provide user with a visual element.

**Time** - is a module that provides various functions that deal with time and dates within a day. This module is specifically used to support the animate command.

**Software development environment**

**Pycharm** - is an Integrated Development Environment (IDE) used for programming in Python developed by the Czech company JetBrains. It was used in the development of “Animate” because it provides a space for code analysis, debugging and it's great with version control.

**GitHub Desktop -** was used to upload software to a github repository because it is an easy to use tool that allowed the “Animate” team to share and collaborate during the entire  developing process.

**Test methodology**

Each module was tested separately to insure correct functionality before combining all submodules. The development team took turns debugging code and identifying syntax errors. All tokens and identifiers were verified for grammar errors as well as insuring that case sensitive entries were not a problem upon executing and calling the corresponding function.

**Programs used to test translator**

Code used to test translator is found in the language tutorial section of this report.

## 5. Conclusion

“Animate” was created with the purpose of facilitating the drawing experience for adults and children of all ages. It’s a tool that makes programing a fun and easy experience through basic commands that present shapes, colors and movement.

The development  of this program language was a challenge to understand at first but once the learning curve was achieved the implementation of the team's vision for the project was a success.

## Bibliography

[1] Docs.google.com, "Documentos de Google: crea y edita documentos online de forma gratuita.", 2016. [Online]. Available: https://docs.google.com/document/u/0/?hl=es&pli=1. [Accessed: 28- Jan- 2016].

[2]"MS Project Turorial", 2016. [Online]. Available: http://www.ese.upenn.edu/seniordesign/resources/MS_Project_Tutorial.pdf. [Accessed: 30- Jan- 2016].