



IMAGI- Child Friendly Programming Language

Final Report

Prepared by:

Héctor García (hector.garcia@upr.edu)
Edgardo Rivera (edgardo.rivera@upr.edu)
Jariel Laureano (jariel.laureano@upr.edu)

Table of Contents :

Introduction -----	3
Requirements-----	3
Scene Editor Tab-----	4
Code Editor Tab-----	5
Using Variables-----	6
Commands Features-----	7
Command Functions-----	8
Language Development-----	9
Conclusion-----	11

Introduction

Project Motivation

With the technology industry developing at a rapid pace, the need for more programmers increases everyday. Little options exist to introduce programming to kids successfully, this is why our team decided to attack this need. By creating a new programming language focused and designed for children we will be able to encourage and motivate students at a young age to pursue STEM related careers but specifically, computer and software engineering majors. After experiencing IMAGI, kids will be able to understand the basics of programming, learn to have fun with it and be prepared to move to a more standard programming language like python or java in the future.

Project Concept

IMAGI is a child friendly programming language developed with the goal of introducing programming to kids in a new and interesting way from a very young age. Lines of code alone will surely have a hard time arousing any children at such a young age, this is the reason why IMAGI's programming experience is aided by a simple graphical user interface which will allow the programmer to interact with different characters and scenes, and create simple storylines by writing code. One of the challenges of kids as our scope is that this language must try to develop the programming instincts in the simplest way possible, avoiding complicated syntax, overcrowding of commands and ambiguity. Using commonly known words and making the programming language straightforward, as easy as talking, is the focus.

Requirements

The tools and requirements needed to be able to use our programming language are kept to the minimum possible so that it is easy to use in any of the most popular operating systems. It is required that the user has Python v2.7.11, Qt4 and PyQt4 installed to make sure the language can be run from the command prompt.

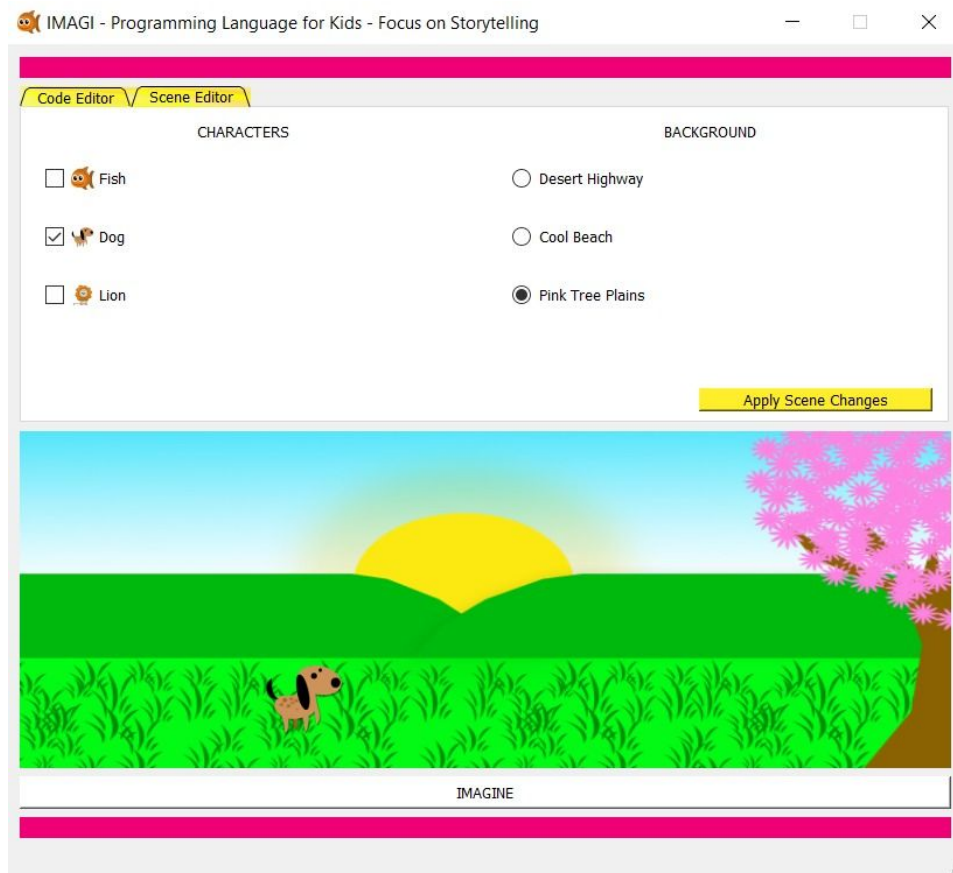
Skills Required

IMAGI is a child friendly programming language thus no previous experience programming is necessary. IMAGI is designed to develop various skills on children, such as: programming, storytelling, spatial perception and imagination.

LETS

IMAGI !!!

Scene Editor Tab



On this tab can be selected the scene by clicking one of the 3 options presented on the background section. The scenes available are:

- Desert Highway
- Cool Beach
- Pink Tree Plains

The character selection is on the left side below Characters. All characters can be used at the same time by selecting each one or you just can choose the one that you like the most. The available characters are:

- Fish
- Dog
- Lion

After selecting the options you want to apply to your scene just click the “Apply Scene Changes” to save them .

Code Editor Tab



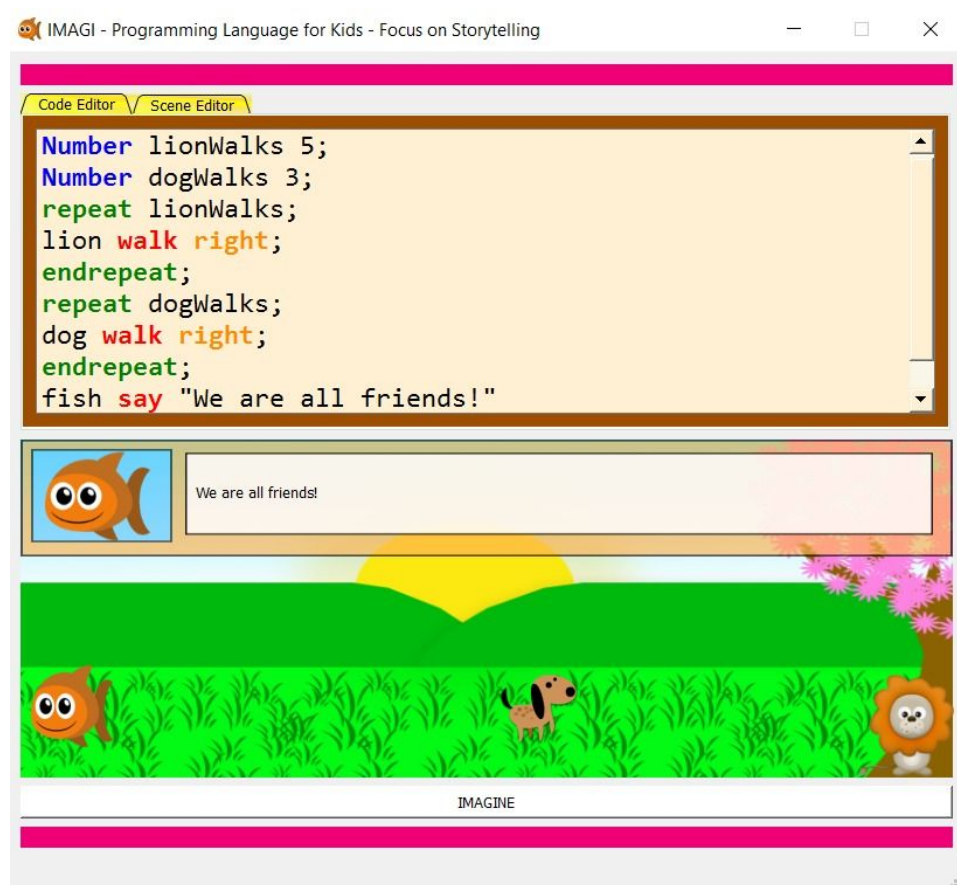
On the code editor tab is where all the magic occurs. It is located on the top of the window. On this tab you can write your program lines, just let your imagination flow. The code written can be tested on any moment by pressing the “IMAGINE” button. The result will be displayed on the scene.

Copy paste the following lines depending the characters you are using for an preview:

```
fish say "Hello my name is Fish!";  
dog dance;  
lion jump;  
lion say "Roar";
```

Then just click "IMAGINE"

Using Variables



Using variables on IMAGI is very simple. There are only two types of variables on this language and they are:

- Number
- Word

To declare a variable you have to follow the following syntax:

VariableType varName Value;

An example of this can be see on the above image. The first two lines of the code declare two variables of type Number.

Name	Value
lionWalks	5
dogWalks	3

Example Wordtype: Word salute “Hello”;
Once that you have declared an variable, you would be able to use it later on code.

Commands Features

Commands/Tokens	Target	Number of Attributes	Attributes Description
jump	Character	N/A	No attributes needed.
walk	Character	1	Direction: left or right.
say	Character	1	Text to say: raw string or word type
domath	Character	3	Operator: +, -, * or / Number/s to add: can be a raw number or a number type described below.
dance	Character	N/A	No attributes needed
endrepeat	N/A	N/A	No attributes needed
grow	Character	N/A	No attributes needed
shrink	Character	N/A	No attributes needed
ask	Character	2	Text to prompt: raw string or word type Variable to store input: word or number type
repeat	N/A	1	Times to repeat, can be a raw number or an NumberType variable
run	Character	1	Direction: right or left
Word	N/A	2	Variable Name, raw string
Number	N/A	2	Variable Name, value

Commands Functions

jump: insert a jumping animation to the given target.

walk: insert a walking animation with the desired direction to the given target.

say: create a dialog box with the given message.

domath: resolve the given mathematical expression (add, subtract, multiply, divide)

dance: insert a dancing animation to the given target.

repeat: the lines between the repeat statement and endrepeat, will repeat given times.

endrepeat: ends a repeat loop.

grow: the target character will grow.

shrink: the target character will shrink.

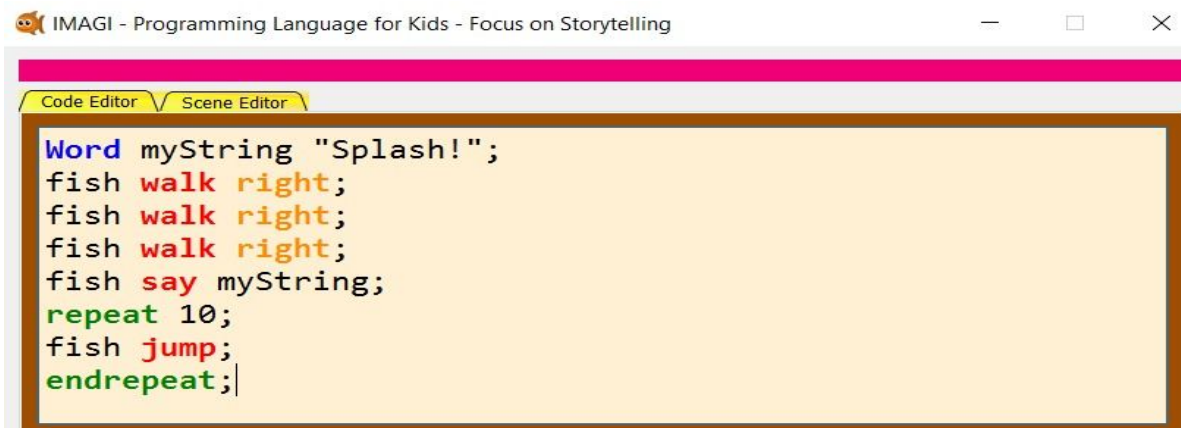
ask: create a dialog box with the given question.

run: insert a running animation with the desired direction to the given target.

Word: create a variable of type word with the given string.

Number: create a variable of type with the given value.

Example of various commands



The screenshot shows a window titled "IMAGI - Programming Language for Kids - Focus on Storytelling". Inside the window, there are two tabs: "Code Editor" and "Scene Editor". The "Code Editor" tab is active and displays the following code:

```
Word myString "Splash!";
fish walk right;
fish walk right;
fish walk right;
fish say myString;
repeat 10;
fish jump;
endrepeat;
```

Language Development

Translator Architecture

The translator architecture of this language is composed of various modules and it has a shell programming structure. The lexical analysis and parsing work together to make this language a working one. There is a module that is in charge of tokenizing every single program line and another module that checks if the statement is a valid one. Then if the statement is a valid one, there is a Command Processor that is in charge of verifying if the command is a valid one and executing the corresponding executable. All this is controlled by our own coded compiler.

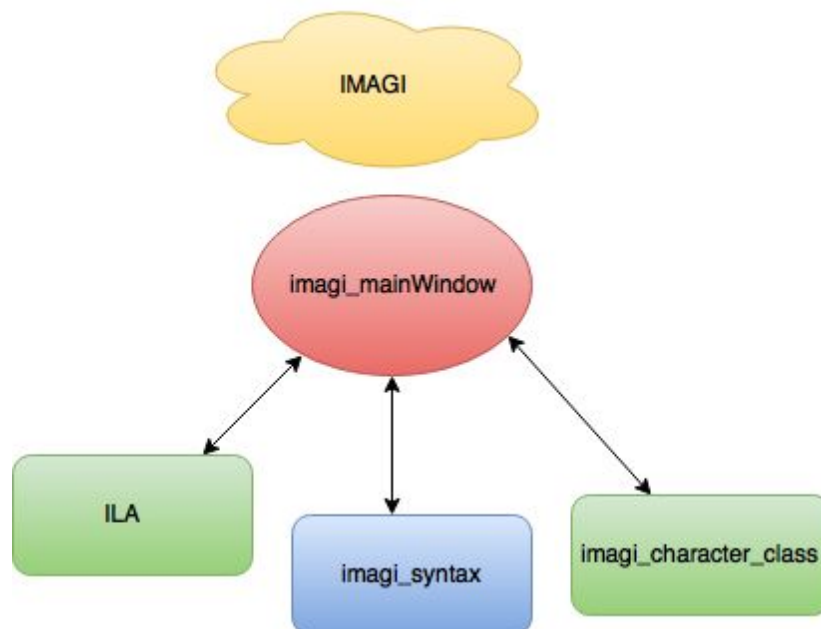
Order of execution:

- Read Code Line from the code editor
- Send Line to the Compiler
- Tokenize Line (classify tokens)
- Check if Valid Statement
- Check if Valid Command
- Execute Command
- Repeat Again Until All Lines are Translated
- Run All Animations (Presented on the Scene)

Module Interface

IMAGI consists of the following modules:

- **imagi_mainWindow** is in charge of joining all the components that make up IMAGI.
- **imagi_syntax** is in charge of the syntax highlighting seen in the code editor of IMAGI's main window.
- **imagi_character_class** is used for creating character objects, the protagonist of each story creates in IMAGI.
- **ILA** is the IMAGI language analyzer. Is in charge of all the lexical analysis and parsing.



Software Development Environment

Because we had a restricted budget, we decided to use PyCharm Community Edition. As stated in their website, it provides code completion, error highlighting, automated code refractors and more. It's smart code editor provides support for a

number of different programming languages like JavaScript, CSS, TypeScript and Python which was the language used for the development of this project. Pycharm also provides support for version control, letting you link your Github repository with your project, making the team interaction and development a lot easier and agile.

Test Methodology During Development

Because of the time constraints we had for this project and the complexity of the translator, automated testing was not implemented, since it would have consumed a large amount of time to develop. Instead the best and easiest way to test our translator was to write code snippets and run them for the translator to analyze and present the result on the output console, using simple print commands to make sure the correct functions were called during translation and the output was displayed as expected. Once we identified flaws or errors in our output console, we ran the same code snippet once again, but this time following the program execution with the debugger. Doing this we were able to identify where our translator was flawed and were able to find a fix in no time.

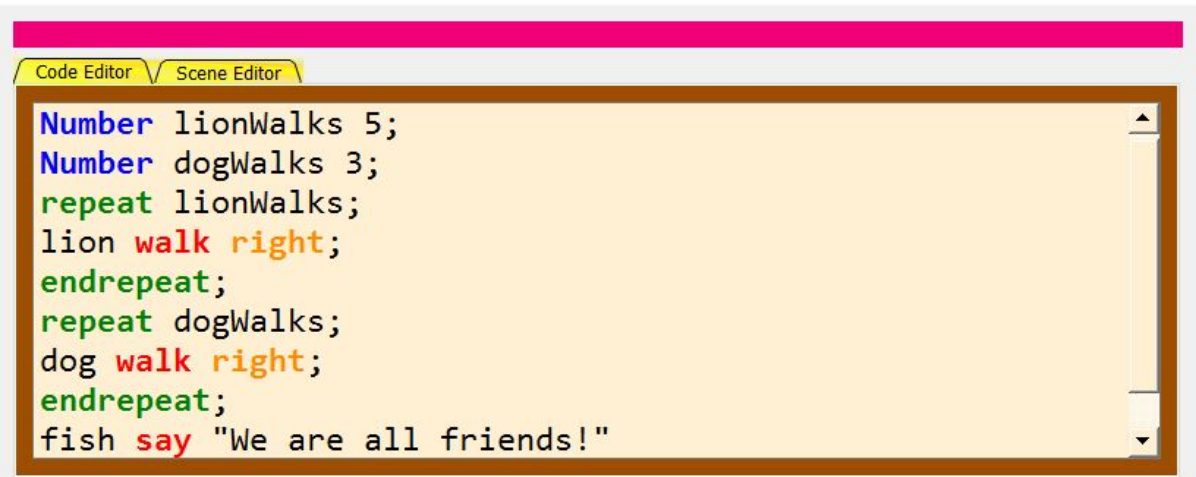
Example Programs

The following are images of example programs made during the development of IMAGI.



The screenshot shows a window titled "IMAGI - Programming Language for Kids - Focus on Storytelling". Inside the window, there are two tabs: "Code Editor" (selected) and "Scene Editor". The code editor contains the following script:

```
repeat 4;
dog walk right;
endrepeat;
dog say "Let's Play go fish!";
dog walk right;
lion walk left;
fish walk right;
```

A screenshot of a code editor window titled "IMAGI - Programming Language for Kids - Focus on Storytelling". The window has a pink header bar and a white title bar with standard window controls. Below the title bar, there are two tabs: "Code Editor" (selected) and "Scene Editor". The code editor contains the following Python code:

```
Number lionWalks 5;
Number dogWalks 3;
repeat lionWalks;
lion walk right;
endrepeat;
repeat dogWalks;
dog walk right;
endrepeat;
fish say "We are all friends!"
```

Conclusion

During the development of IMAGI, we had to learn many new technologies we did not have experience or were not completely familiar with, these include but are not limited to: Python as a programming language for a large scale project, Qt Designer for designing the GUI component of our project, PyQt4 for integrating the Qt Design into Python and using a GitHub repository for project version control. To learn these technologies, we had to read complex documentation from the Qt API which was done in C++ while we were using Python and watch several tutorials to get the hang of the technologies before starting to work on our project which was a very ambitious one since we deal with many animations. Working with a team was also a new experience for most of us, so we had to organize ourselves the best we could to avoid problems. From the very beginning of the semester we made a schedule with milestones that we tried to accomplish the best we could, this helped us in keeping our project organized and avoid the last minute deadline stress. Each team member was in charge of a specific part of the project and whenever a doubt was presented, the respective team member was the specialist and could help solve the problem. Integrating the work done by each member was another challenge, thankfully, code comments helped guide us through the implementation of it. The overall experience of designing and developing IMAGI was a great one, each of us learned about project management and other important software development skills such as motivational team speaking when we doubted ourselves because of our ambitious project, at the end we believe we did a great job and enjoyed our time working with IMAGI.