A Graphical Programming Language Editor

Candidate No. 198719

November 3, 2020

Contents

1 Introduction

Graphical programming can be a fantastic and intuitive way to introduce new programmers to the scene. When programmers ask others for assistance, those helping typically do so in a visual style, using whiteboards, drawing flowcharts, with boxes and arrows indicating the flow of the program. Why can't we make programs in the same style if we find it so helpful to read? The concept behind graphical programming is specifying the elements of the program graphically rather than textually [2].

Popular examples of graphical programming include Scratch, as well as a personal favourite that I used during my GCSE Computing education, App Inventor! What's clever about Scratch is its simplicity due to the block-based visual programming, aimed towards younger children to help them get into coding!

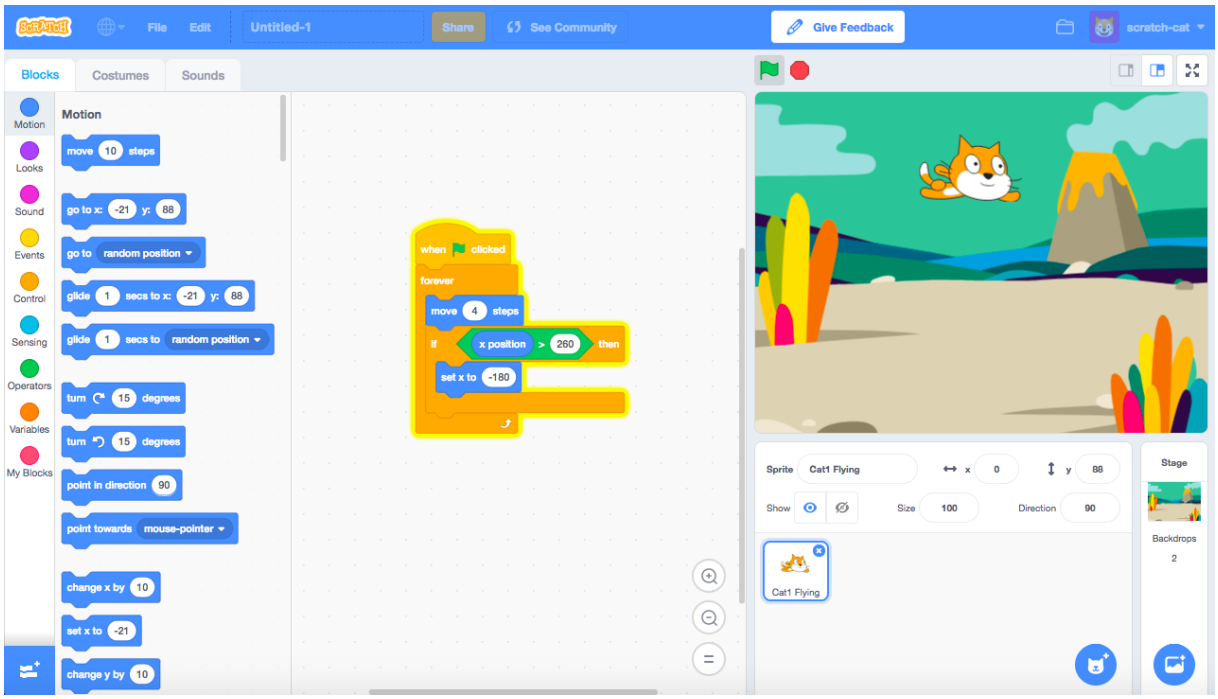

Figure 1: Scratch's visual scripting over a white canvas [3].

MIT App Inventor is one of my personal favourites after experiencing it myself during GCSEs, having to building an on-campus application for a University! It uses a graphical user interface, similar to Scratch, as well as providing the blocks that you can use, visible on the left hand side.

Figure 2: MIT App Inventor graphical programming [1].

Both what Scratch and App Inventor do superbly is colour code their blocks to identify exactly what they are representing. There are colour coded blocks for variables, events and controls, and so on! For instance in App Inventor, variables are coloured in Orange blocks, and procedures and show in a big purple block, encapsulating all the blocks like a function.

1.1 Aims

1.2 Objectives

Needs of intended users.
Brief description of the problem area.
Motivation

Project relevance.
What will you do?
How will you do it?
Why will you do it?

1.3 Problem Area

The main challenge of implementing an interface to be able to program on is the required time and skill set. This project may never be fully complete as more work can always be done to improve the functionality of the interface, or making the interface more user-friendly.

The project can be broken down into key areas that require focusing on:

- Graphical Design - modelling, interface, simplicity, GUI.
- Functionality - programming.

I have limited GUI design in Java from my 2nd year module, Further Programming, therefore this process can be said to take up a large amount of time. Careful consideration is needed to not overspend my time on the GUI design process, and rather get the functionality of said interface to work.

Functionality requires programming to be clean and efficient, so not to go back over the code and forget what was written. Code must be documented and commented along the way to make sure anyone who reads the code, including myself in the future, is able to understand what was written and why it was decided.

1.4 Expected Outcomes

The expected outcomes of this project are a fully working, funtional graphical programming language interface. The user should use the tool independent from the backend Java textual code, and be able to perform the certain tasks:

- loops - create and work with basic for & while loops.
- data structures - create and interact with data structures such as lists.

2 Professional and Ethical Considerations

Discuss the ethical issues your prject is likely to deal with:
- awareness of technological procedures and standards.
- understanding and complying with legisaltion.
If need humans, include plan and timetable for obtaining ethical approval.

3 Related Work

Scratch and App Inventor - I used app inventor for my GCSE assignment, making an application for a university campus.

Do some background research.
What has already been done in this field?
How is your approach different and better than others?
What is the novelty of your project, compared to others?
What is your contribution to the field?
Do not re-invent the wheel!

4 Requirements Analysis

In this section, I have taken into careful consideration the aims and objectives for this project. Below you will find a compiled list of software requirements, split into two categories: **mandatory** requirements, the fundamental requirements that is needed for the project to work, and **desirable** requirements, optional further requirements that can be met to provide further quality of life experiences for the user.

4.1 Mandatory Requirements

Mandatory Requirements	
Requirement	**Specification**
1. Application will work on any java-run environment.	Whether the user is on a Windows, Mac OSX or Linux system, the application will work.
2. The interface shall accept user input via keyboard and mouse.	User will be able to use keyboard and mouse to create and interact with the canvas.
3. The program shall run until is it killed.	The process will not stop running until either it is killed via user or forced shutdown.
4. The program will output any errors, if any.	If there are any run-time errors, the terminal will output those for the user to see.
5. The user shall be able to use functional loops such as for and while loops.	The user will have the ability to interact with loops in the interface.
6. The user shall be able to create variables.	The user will be able to create and interact with variables.
7. The user shall be able to create functions or procedures.	Users will have the ability to create functions /methods that perform a variety of tasks.
8. The user will be able to create data structures.	Data structres such as lists will be available for the user to implement and interact with.

The main

Does your project meet the needs of a target group of users?
What are the needs of these target users?
How would an ideal system meet their needs?
To what extent does your solution contribute to this?
What do you expect to achieve within the given time?
What will you definitely not achieve?
You should stick to this analysis till the end of your project!

5 Project Plan

Describe what you have already done.
Schedule what you still want to do.
How are your tasks interdependent?
Writing the draft report should be one of your main tasks.

6 Interim Log

Log of the meetings with your supervisor.
Should cover discussions of every important project stage.
Record the prupose and the outcome of every meetings.
Note how these meetings helped you follow your plan.
Include this log as an appendix to your report.

7 Appendices

7.1 Project Proposal

References

[1] Richard Albritton. "What is App Inventor?" In: *https://learn.adafruit.com/mit-app-inventor-and-particle-io* (2018).

[2] REMI Dehouck. "The maturity of visual programming". In: *http://www.craft.ai/blog/the-maturity-of-visualprogramming* (2015).

[3] The Scratch Team. "3 Things To Know About Scratch 3.0". In: *https://www.medium.com/scratchte blog/3-things-to-know-about-scratch-3-0-18ee2f564278* (2018).