Problem Statement: A Code in Sheep's Clothing

CS 472

Fall 2019

Johannes Freischuetz

Abstract

The main problem with teaching programming to new programming students today is that there is no easy way to introduce students to programming, especially for students who are younger. This project proposes to develop a Domain Specific Language for students to be able to define simple board games that could be played. This would allow the students to be able to relate to the programming that they are doing as well as learning core programming topics without needing to understand abstract concepts. The restrictions would also allow for better error messages and simplified instructions. Once defining the Domain Specific Language, a specialized editor would allow for more restrictions or visual editing that is not currently possible in other languages. This will help new students who have no experience significantly.

1 Definition

The main issue that is solved is that there are currently no easy programming languages for students that are early in computer science can use to be able to gain familiarity with concepts of computer science without needing to fully understand those concepts. The goal would be for students as early as middle school to be able to understand the topics and be able to begin working on projects independently after short lessons. When attempting to do this with current programming languages the burden of learning all the required material is simply overwhelming and requires too much time. For example, teaching the concepts of C or C++ to a middle schooler so that they can learn basic topics in computer science is simply a waste of time.

There are languages that attempt to make it easier for people who are new to programming to learn, such as Scratch, but the issue with this is that many of the concepts do not transfer well to programming languages that are used in industry. Using one of these beginner languages can lead to its own challenges. It can be a very difficult transition to a more common language such as C++ or Java. The algorithms, for example, may be transferable, however the fundamental skills that are being taught will likely be lost in the transition.

To make learning computer science topics easier, many times teachers will use algorithmic analogies to understand the concepts without having needing to have an in depth understanding of a specific programming language. This prevents the waste of learning a language that does not relate to a common programming language. While this solves some issues, ideally both would be learned at the same time.

Another major issue when learning programming is that many times the easiest introduction to programming is math-based introduction. Middle school students are likely to have learned some concepts of algebra, geometry, and probability but are unlikely to have experienced more complex mathematical concepts such as algorithms or series. For students that are not math based, especially middle schoolers, this may turn them away from programming. Many Students come to computer science from gaming, and in turn want to be able to make their own games. Ideally the introduction for students into programming would focus on their interests. For many of these students that would mean something interactive.

2 Solution

The solution to this problem is to define a domain specific language or a DSL to be able to give students easy access to programming while also allowing the students to have an immediate result. Students are often introduced to programming through games. To bring these two ideas together the DSL would focus on defining a language for students to be able to define the rules of a board game. This does not necessarily mean that every board game would be definable, but rather a set of core board games must be able to be defined, and then a wide variety of custom games made by students could also be easily defined.

The main goal of the language is not to be able to define every type of game possible, but rather make it easy to define simple games. It is more important to be able to define a simple game easily than to be able to define a complex game. Because of the limitations of middle schoolers programming abilities, it is more important for the language to be simple even if at comes at the cost of restrictions. For this reason a set of board games has been chosen that must be able to be defined so that the requirements can be easily restricted. The list is: Tic Tac Toe, Battleship, and Connect 4. With these variations there are obvious restrictions that can be made, such as that the board will always be in a rectangular grid, and that there will be pieces that can be placed on the grid. Defining the language is a large part of the project and it will develop with the project, but the goal of keeping them simple will continue to be the main goal. Adding as many features as possible would be ideal, but simplicity is still more important.

There are many specifics of this language that have yet to be decided. There are some constructs however that will certainly appear. The language must have some capability to perform loops. Further, all board games will be expected to conform to a rectangular grid. Students should be able to define a unique board, how movement will take place, what affect those movements have on the game state, how player's turns should progress, and ultimately play the game. This means the language will need to be able to interpret the student's commands and produce a functional board game. All these constructs should be written in a way that is easy to understand and reprogram new games. The language should also resemble typical programming languages, so that the skills learned can be transferred.

3 METRICS

There are multiple things that must be finished for this project to be considered completed. The main thing that needs to be finished is for the language to be properly defined so that students can be able to write in it and develop their own board games and then run through them. This is the bare minimum requirement with a Haskell implementation of the language. This will allow for further development in the future and will essentially act a definition for the language for further work.

Ideally there would be special features added to this language to make it something special compared to what already exists. There are two things that could be added to the language to make it something that stands out. The first extension to the language would be for there to be an editor for the language that would allow for even more restrictions and error checking than would be in a text editor. This could either work like a text editor or an IDE. Another extension that could be done is to make it a visual language. This would mean that to define the movement of a piece, for example, you could simply see a board and draw all of the possible movements that a piece can make visually. This would be similar to how when looking at the movement allowed for a piece on a chess board the possible squares are highlighted.