Philipps-Universität Marburg
Department of Mathematics and Computer Science
Programming Languages and Tools group

A Block-Based Programming Language for Visually Impaired and Normally Sighted Students

Context

Block-based (graphical) programming languages such as Snap![1] or Scratch [2] are popular in programming education. Since code can be assembled from predefined blocks that even have an intuitive visual representation, programming novices do not have to remember keywords and are even actively prevented from making syntax errors. This is why such a programming approach is also called *low-code* and often used in settings with inexperienced programmers such as end-user programming. The education-centered languages also usually come along with an intuitive visualization of the program execution, e.g., because the program objects correspond to graphical sprites with state that is changed by the program. A state change is then immediately reflected in a change of the graphical representation.
While being easily usable and intuitive for students with normal vision, the graphical paradigm is not easily accessible to students with vision impairments. Graphical models are known to be space-consuming and related elements may be shown in distant places. This makes it challenging for students with low vision to capture a block-based program as a whole. For blind, graphical block-based program representations are almost not usable at all, since screen readers cannot translate them to meaningful spoken text.

Assignment

In this assignment, an initial framework should be developed for a programming language and development environment that makes the benefits of a block-based, educational programming language accessible to students with vision impairments. To support mixed groups of students with and without vision impairments, as well as teachers with and without vision impairments, the environment should facilitate usage in different ways, including purely graphical and purely textual interaction.
The suggested solution approach is to use the Eclipse-based tools for the development of domain-specific languages (DSLs). In this approach, the Eclipse Modeling Framework (EMF) [2] is used to define a metamodel, which corresponds to the grammar of the programming language. Based on this metamodel, tools like Xtext[3] or[4] can be used to generate textual and graphical editors for the language. When these editors are used, Eclipse maintains an in-memory model of the edited program, which is independently of the representation. The main tasks in this assignment are:

- Design a simple imperative educational programming language with a syntax that is suitable for textual and scalable block-based representation.
- Integrate simple input/output features into this language, e.g. via text messages.
- Implement that language (e.g., with EMF and related tools) including a textual and a graphical editor, as well as an interpreter to execute programs.
- Demonstrate that the textual and graphical notations can be linked.

Info

Block Coding	
Java	
(e.g.) EMF, Xtext, GMP	
Context-free grammars, (meta-)models	
Research	▰▱▱▱▱
Develop	▰▰▰▰▱
Thesis (BSc, MSc), FoPra	
1–2 people	
Prof. Christoph Bockisch	

[1]The Snap! homepage: `https://snap.berkeley.edu/`.
[2]The Scratch homepage: `https://scratch.mit.edu/`.
[3]Homepage of the Xtext project: `https://eclipse.dev/Xtext/`.
[4]Homepage of the Graphical Modeling Project: `https://eclipse.dev/modeling/gmp/`.

References

[1] A. Mountapmbeme, O. Okafor, and S. Ludi. Accessible blockly: An accessible block-based programming library for people with visual impairments. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '22, New York, NY, USA, 2022. Association for Computing Machinery.

[2] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, Upper Saddle River, NJ, 2nd edition, 2009.