GymInf | Informatikausbildung für Gymnasiallehrkräfte | Formation en informatique destinée aux enseignant-e-s au gymnase

Revealing Programming Language Abstractions

An Excerpt of nand-to-tetris – in Reverse – Using Smalltalk

GymInf Individual Project

Simon Bünzli
from
Bern, Switzerland

Philosophisch-naturwissenschaftlichen Fakultät
der Universität Bern

01. August 2025

Prof. Dr. Timo Kehrer, Prof. Dr. Oscar Nierstrasz

Software Engineering Group
Institut für Informatik und angewandte Mathematik
University of Bern, Switzerland

UNIVERSITÄT BERN

UNIVERSITÉ DE NEUCHÂTEL

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Abstract

Not an *abstract* yet, but the original project description:

Ziel des Projekts ist, ein empirisch abgestütztes Instrument für den Programmier-Unterricht am Gymnasium zu entwickeln, in welchem Schüler:innen verschiedene Abstraktionsebenen interaktiv erleben können.

Auf der Basis von Processing mit Python Syntax (https://py.processing.org/) soll der einerseits der visuelle Ablauf eines Programms, aber auch die Parsing-Schritte und die Übersetzung in Byte-Code Seite-an-Seite sicht- und untersuchbar gemacht werden, damit Schüler:innen die Auswirkungen ihres Programmcodes auf die Maschine live erleben können.

Die Entwicklung des Produkts wird theoretisch begleitet und das Produkt selbst empirisch geprüft werden.

Als Basis der Umsetzung dient Glamorous Toolkit, eine Entwicklungsumgebung basierend auf Smalltalk/Pharo, welche u.a. von Oscar Nierstrasz für Master- und Doktoratsstudiengänge weiterentwickelt worden ist.

Contents

1 Introduction

Programming with Processing vs. ”Little Man Computer” or ”Human Resource Machine”

2

Leaky Abstractions when Teaching Programming

On the lack of connecting high-level languages with lower-level concepts

2.1 Multitier Architectures

A common concept in computer science as seen in computer architecture, network stacks, but also outside in Mathematics, language, etc.

2.2 Leaky Abstractions

Where knowledge of lower abstraction levels might help, concept introduced by [10].

2.3 Didactic Approaches

See e.g. [9], [5], [3] or [4] only focusing on one aspect

2.4 Abstractions in IDEs

Brief overview over views offered by common IDEs (such as VS Code) but mainly didactic ones such as Thonny ([1]), Mu ([11]), *etc.*.

2.5 Teaching Top Down

Working downwards from gaming, as in [13]

2.6 Teaching Bottom Up

Running Tetris on NANDs as described in [12], [7]

3 Technical Background

Background knowledge required for understanding the following chapters.

3.1 Processing

Brief overview over the ”Processing” programming language (along [8]) and reasons for using it.

3.2 Glamorous Toolkit

Brief introduction into GT for the uninitiated and reasons for using it. ([2])

3.3 Moldable Development

Referring to [6].

4
Processing Abstractions

4.1 Development of ”Processing Abstractions”

Excerpts from gt-exploration Lepiter pages

4.2 Abstraction Levels

For each a short problem description and a presentation of the chosen approach:

4.2.1 Source Code

4.2.2 Abstract Syntax Tree

4.2.3 Transpilation/IR

4.2.4 Machine Code

4.2.5 Output

5 Teaching with PA

How to use it

5.1 Lesson on Computer Architecture

Using PA to demonstrate what happens under the hood when running a program in a high level language.

5.2 Lesson on Compilers

Using PA to demonstrated the steps of lexing, parsing, transpiling, compiling and optimizing.

5.3 Lesson on Introduction to Programming

Using PA as a live programming environment.

6 PA in Practice

PA has been used twice with students (on 2025-05-12 and 2025-06-30).

6.1 First Round

6.1.1 Setting

6.1.2 Observations

6.1.3 Student Feedback

6.1.4 Learnings

6.2 Second Round

6.2.1 Setting

6.2.2 Observations

6.2.3 Student Feedback

6.2.4 Learnings

7 Conclusion

7.1 Future Work

A

Installing and Using `Processing` Abstractions

B

Data from Questionnaires

Bibliography

[1] Aivar Annamaa. Introducing thonny, a python ide for learning programming. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, Koli Calling '15, pages 117–121, New York, NY, USA, 2015. Association for Computing Machinery.

[2] Tudor Gîrba and Oscar Nierstrasz et al. *Glamorous Toolkit*. feenk.

[3] Stefan Hartinger. *Programmieren in Python*. Universität Regensburg, 2020.

[4] Irene Lee, Shuchi Grover, Fred Martin, Sarita Pillai, and Joyce Malyn-Smith. Computational thinking from a disciplinary perspective: Integrating computational thinking in k-12 science, technology, engineering, and mathematics education. *Journal of science education and technology*, 29(1):1–8, 2020.

[5] Eckart Modrow and Kerstin Strecker. *Didaktik der Informatik*. De Gruyter Studium. De Gruyter Oldenbourg, München, 2016.

[6] Oscar Nierstrasz and Tudor Gîrba. Moldable development patterns. 2024.

[7] Noam Nisan and Shimon Schocken. *The elements of computing systems : building a modern computer from first principles*. The MIT Press, Cambridge, Massachusetts, second edition edition, 2021 - 2021.

[8] Casey Reas and Ben Fry. *Processing: A Programming Handbook for Visual Designers and Artists*. The MIT Press, 2 edition, 2014.

[9] Sigrid Schubert and Andreas Schwill. *Didaktik der Informatik*. Spektrum Akademischer Verlag, Heidelberg, 2. auflage edition, 2011.

[10] Joel Spolsky. The law of leaky abstractions, 2002.

[11] Nicholas H. Tollervey. Code with mu, 2023.

[12] Ünal Çakıroğlu and Mücahit Öztürk. Flipped classroom with problem based activities: Exploring self-regulated learning in a programming language course. *Journal of Educational Technology & Society*, 20(1):337–349, 2017.

[13] David Weintrop and Uri Wilensky. Playing by programming: Making gameplay a programming activity. *Educational Technology*, 56(3):36–41, 2016.