EDUKAČNÉ PROSTREDIE NA PROGRAMOVANIE HUDBY PRÍSTUPNÉ PRE NEVIDIACICH ŽIAKOV SEKUNDÁRNEHO VZDELÁVANIA

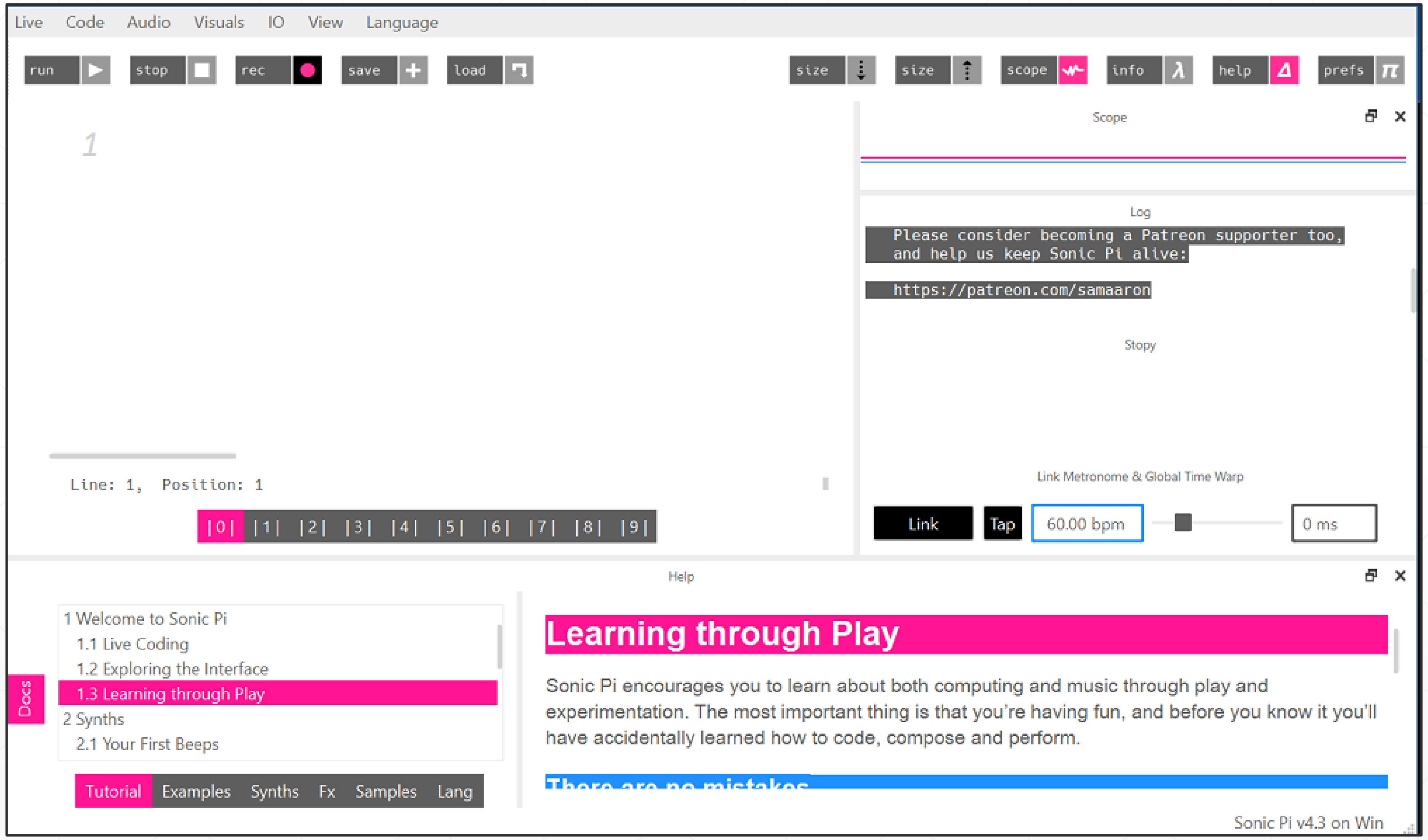
Jakub Švorc

Detaily k práci

- Školiteľka: doc. RNDr. Ľudmila Jašková, PhD.
- Cieľ práce: Vytvoriť programovacie prostredie umožňujúce programovať hudbu pozostávajúcu z viacerých paralelne znejúcich melódií. Dôraz bude kladený na zabezpečenie plnej prístupnosti a efektívnej práce s editorom kódu pre žiakov so zrakovým postihnutím
- Cieľová skupina: 2. stupeň ZŠ

Podobné aplikácie

SonicPi


Live Code Audio Visuals IO View Language
run stop rec save load size size scope info help prefs
Scope
Log
Stopy
Link Metronome & Global Time Warp
Link Tap 60.00 bpm 0 ms
|0| |1| |2| |3| |4| |5| |6| |7| |8| |9|
Help
Docs
Tutorial Examples Synths Fx Samples Lang
Sonic Pi v4.3 on Win

```
Line: 1,  Position: 1
```

```
Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps

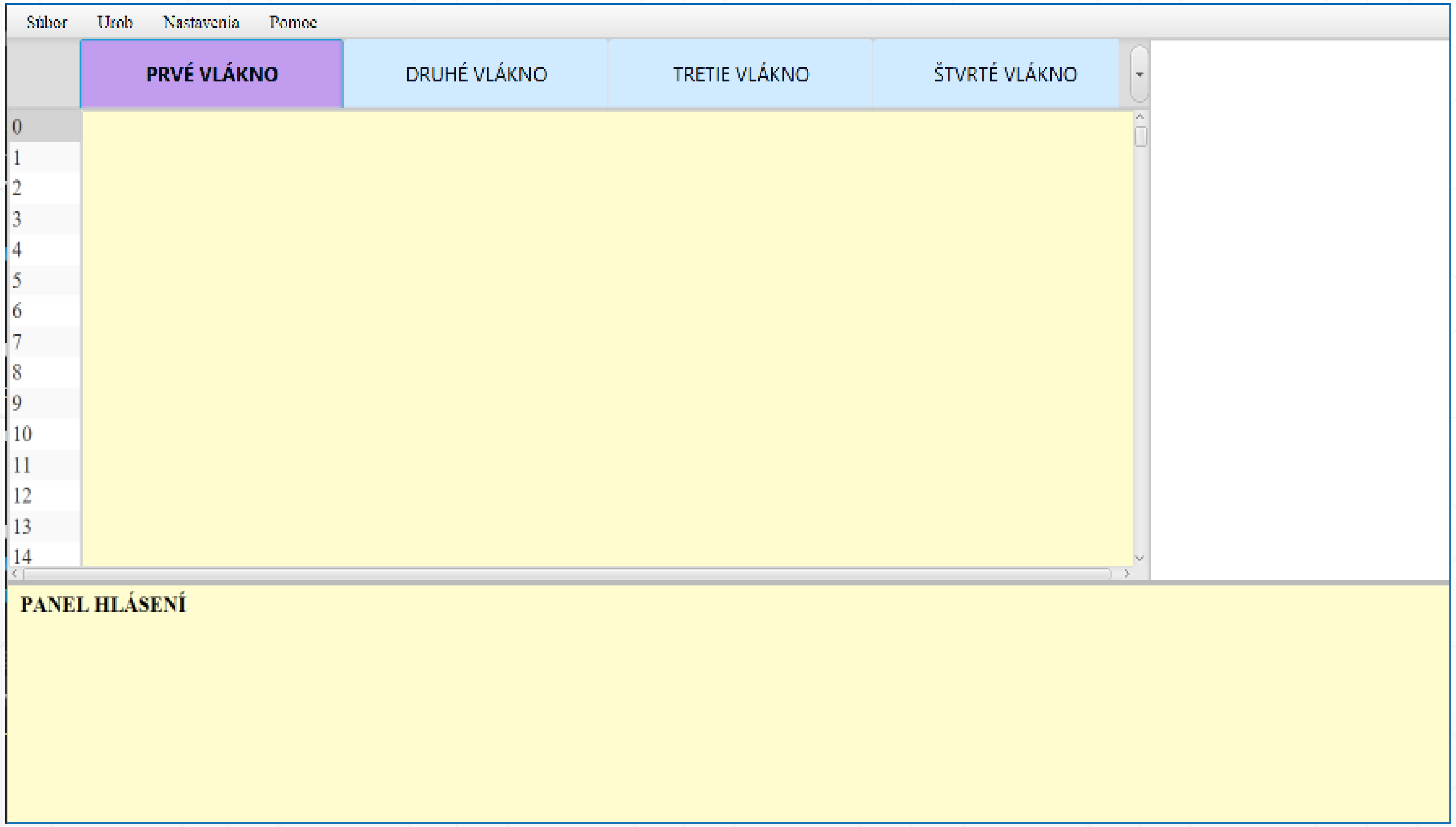
Learning through Play

Sonic Pi encourages you to learn about both computing and music through play and experimentation. The most important thing is that you're having fun, and before you know it you'll have accidentally learned how to code, compose and perform.

There are no mistakes

- Sam Aaron
- Podpora štandardného aj zvukového programovania
- Jednoduchá aj komplexná syntax (veľa parametrov) a základné programové konštrukcie
- IDE s vlastným jazykom
- Bohatá škála vytvárania zvukov a ich rôzne úpravy

TonIK2



- Práca s hotovými príkazmi
- Output panel
- Práca s premennými
- Programové konštrukcie
- Plne ovládateľné klávesnicou

Implementácia

Postup pri riešení práce

- Spracovanie článkov
- Prehľad vhodných technológií
- Zadefinovanie syntaxe
- Aplikácia na test s čítačom obrazovky
- Porovnanie kompilácie proti interpretácií jazyka
- Začatie implementácie

Implementácia

- Voľba vhodného nástroja
  - Python vs. JavaFx vs. .NET WPF + knižnice
- Malá testovacia aplikácia
- Tvorba GUI
- Interpreter
  - naivný spôsob spracovania programu
- Kompilátor a virtuálna mašina
  - zjednodušený bez programovacích konštrukcií

Implementácia - pokračovanie

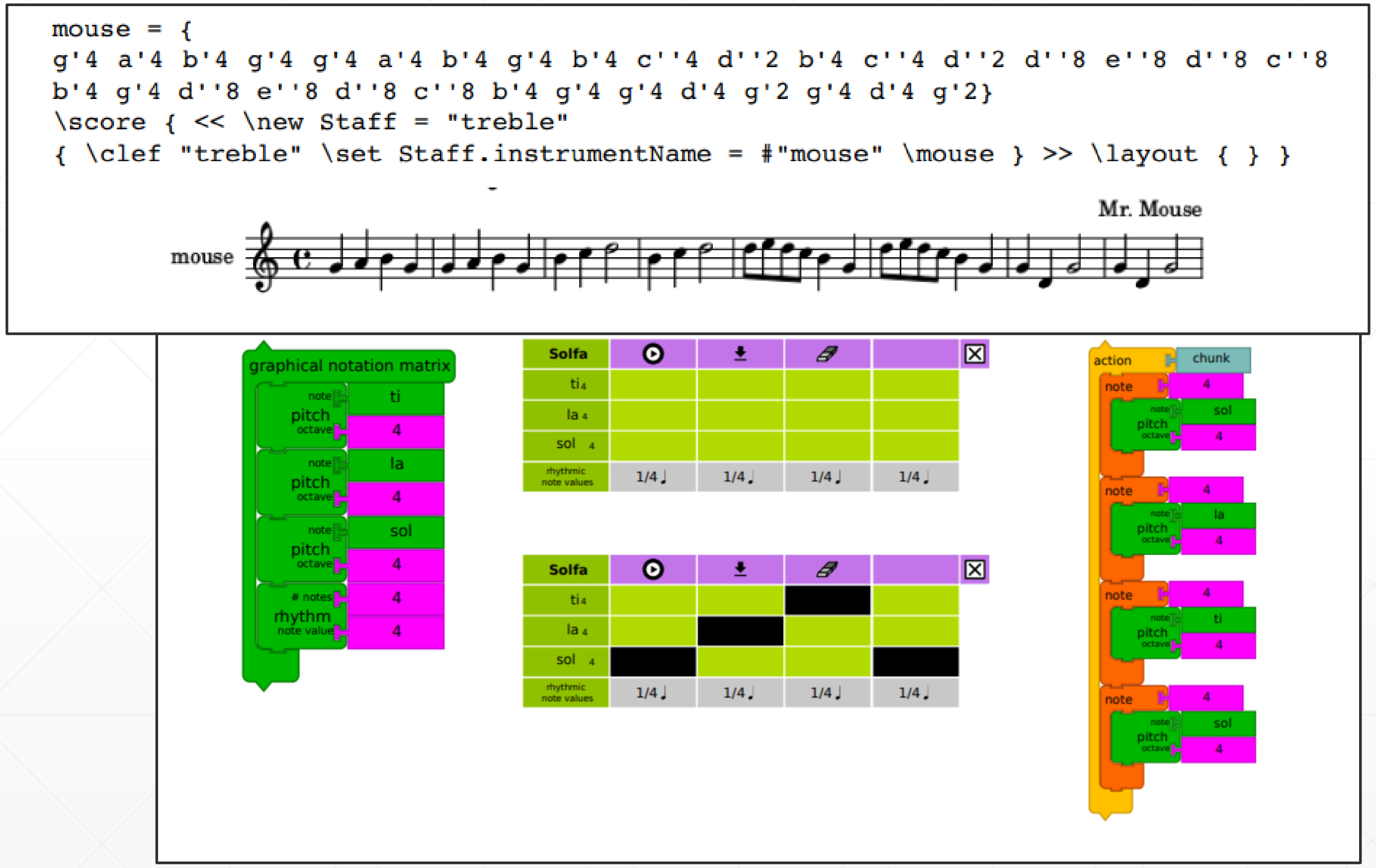
- Prehrávanie tónu
  - Spracovanie dodatočných parametrov
- Zmena nástroja
- Cyklus s počtom opakovaní (for loop)
- Kontroly syntaxe počas kompilácie
- Terminál na debugovanie a hlásenie chýb

Ukážka

Zdroje

Music Blocks: A Musical Microworld

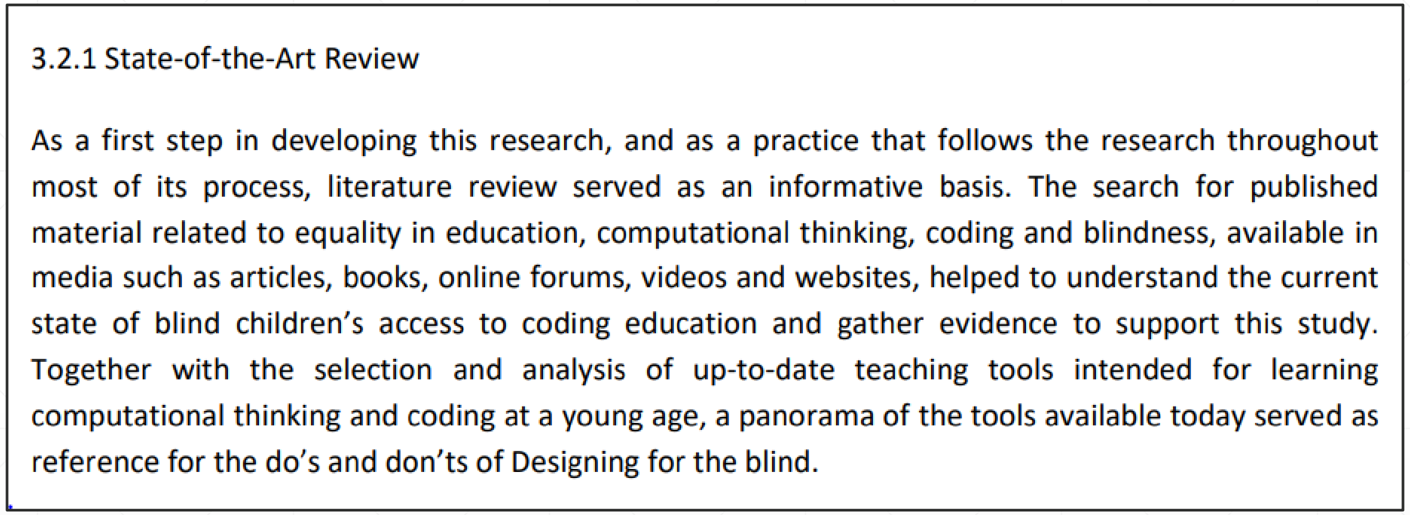
- Walter Bender
- Devin Ulibarri
- Yash Khandelwal



```
mouse = {
g'4 a'4 b'4 g'4 g'4 a'4 b'4 g'4 b'4 c''4 d''2 b'4 c''4 d''2 d''8 e''8 d''8 c''8
b'4 g'4 d''8 e''8 d''8 c''8 b'4 g'4 g'4 d'4 g'2 g'4 d'4 g'2}
\score { << \new Staff = "treble"
{ \clef "treble" \set Staff.instrumentName = #"mouse" \mouse } >> \layout { } }
```

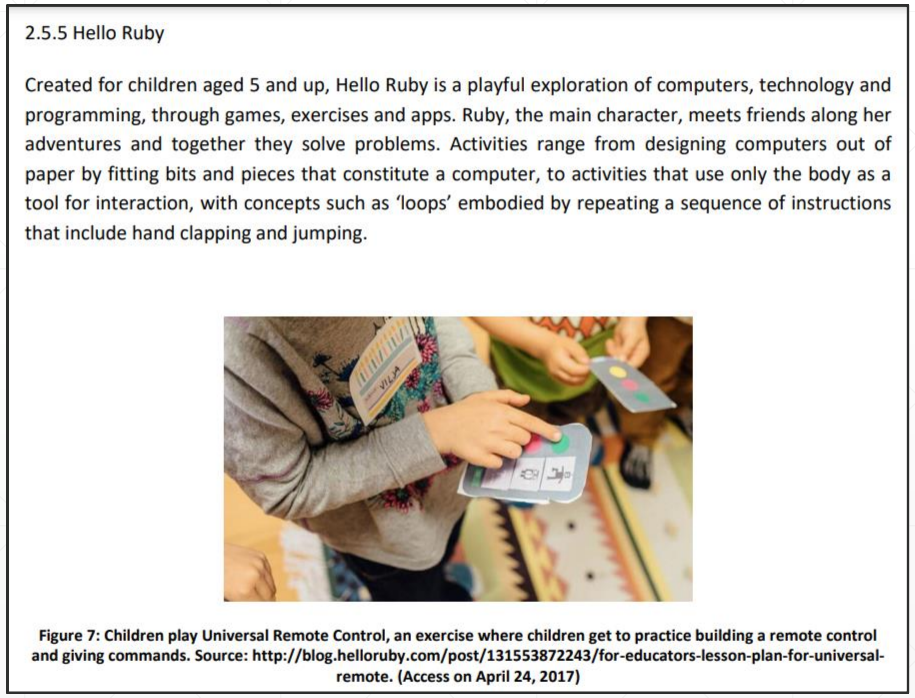
Designing educational programming tools for the blind: mitigating the inequality of coding in schools

- Clarissa Correa de Oliveira



3.2.1 State-of-the-Art Review

As a first step in developing this research, and as a practice that follows the research throughout most of its process, literature review served as an informative basis. The search for published material related to equality in education, computational thinking, coding and blindness, available in media such as articles, books, online forums, videos and websites, helped to understand the current state of blind children's access to coding education and gather evidence to support this study. Together with the selection and analysis of up-to-date teaching tools intended for learning computational thinking and coding at a young age, a panorama of the tools available today served as reference for the do's and don'ts of Designing for the blind.



2.5.5 Hello Ruby

Created for children aged 5 and up, Hello Ruby is a playful exploration of computers, technology and programming, through games, exercises and apps. Ruby, the main character, meets friends along her adventures and together they solve problems. Activities range from designing computers out of paper by fitting bits and pieces that constitute a computer, to activities that use only the body as a tool for interaction, with concepts such as 'loops' embodied by repeating a sequence of instructions that include hand clapping and jumping.

Figure 7: Children play Universal Remote Control, an exercise where children get to practice building a remote control and giving commands. Source: http://blog.helloruby.com/post/131553872243/for-educators-lesson-plan-for-universal-remote. (Access on April 24, 2017)

APL: Audio Programming Language for Blind Learners

- Jaime Sánchez

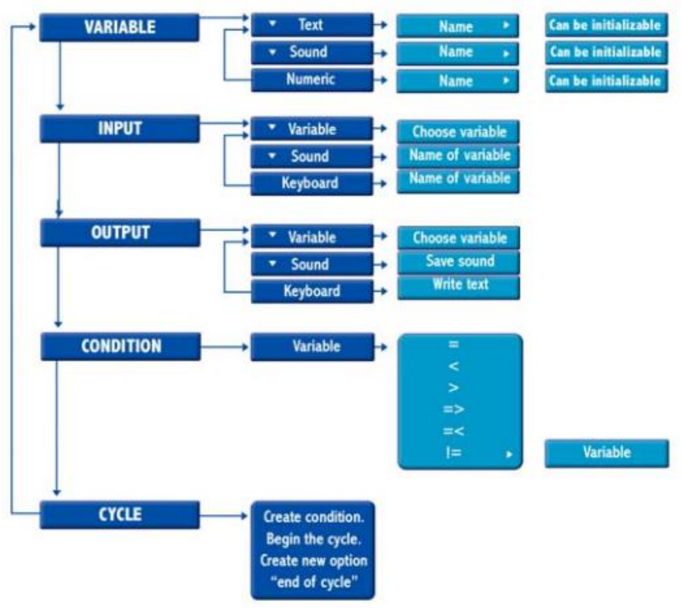


Fig. 3. Command list tree

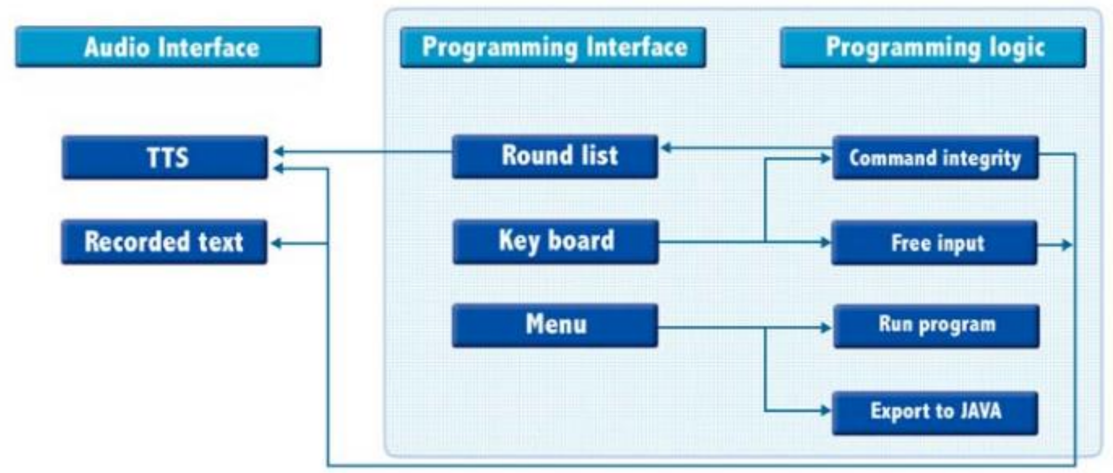


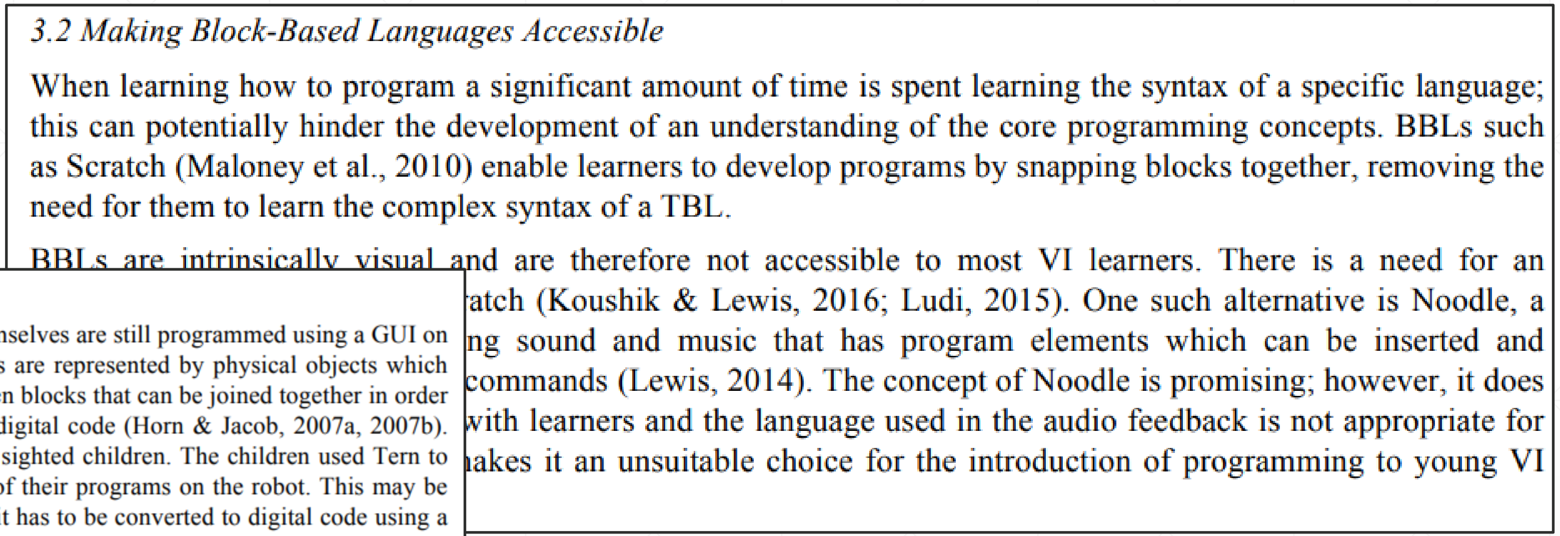
Fig. 2. Layers of APL

3.2 Implementation

APL was developed by using Java 1.4.1 and FreeTTS Java speech synthesizer. These are fine tools for media management and close to the machine logic. APL has the following modules: DataBase, Integrity, Kernel, CHI and generador de JAVA.

Making Programming Accessible to Learners with Visual Impairments: A Literature Review

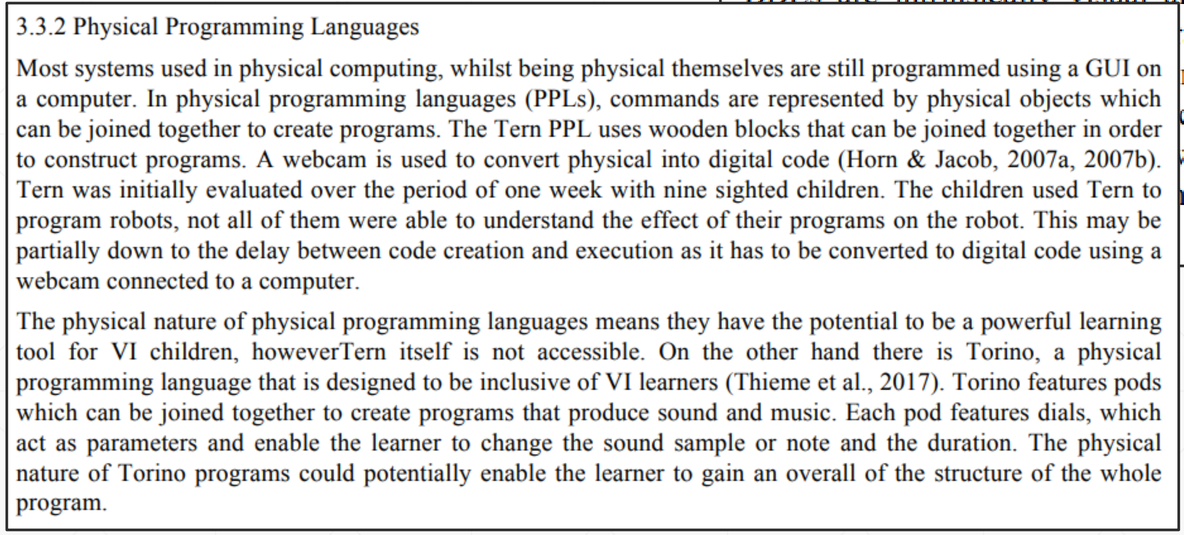
- Alex Hadwen-Bennett
- Sue Sentance



3.2 Making Block-Based Languages Accessible

When learning how to program a significant amount of time is spent learning the syntax of a specific language; this can potentially hinder the development of an understanding of the core programming concepts. BBLs such as Scratch (Maloney et al., 2010) enable learners to develop programs by snapping blocks together, removing the need for them to learn the complex syntax of a TBL.

BBLs are intrinsically visual and are therefore not accessible to most VI learners. There is a need for an atch (Koushik & Lewis, 2016; Ludi, 2015). One such alternative is Noodle, a ng sound and music that has program elements which can be inserted and commands (Lewis, 2014). The concept of Noodle is promising; however, it does with learners and the language used in the audio feedback is not appropriate for akes it an unsuitable choice for the introduction of programming to young VI



3.3.2 Physical Programming Languages

Most systems used in physical computing, whilst being physical themselves are still programmed using a GUI on a computer. In physical programming languages (PPLs), commands are represented by physical objects which can be joined together to create programs. The Tern PPL uses wooden blocks that can be joined together in order to construct programs. A webcam is used to convert physical into digital code (Horn & Jacob, 2007a, 2007b). Tern was initially evaluated over the period of one week with nine sighted children. The children used Tern to program robots, not all of them were able to understand the effect of their programs on the robot. This may be partially down to the delay between code creation and execution as it has to be converted to digital code using a webcam connected to a computer.

The physical nature of physical programming languages means they have the potential to be a powerful learning tool for VI children, howeverTern itself is not accessible. On the other hand there is Torino, a physical programming language that is designed to be inclusive of VI learners (Thieme et al., 2017). Torino features pods which can be joined together to create programs that produce sound and music. Each pod features dials, which act as parameters and enable the learner to change the sound sample or note and the duration. The physical nature of Torino programs could potentially enable the learner to gain an overall of the structure of the whole program.

Budúce ciele

Budúce ciele

- Práca s premennými
- Podprogramy
- Vetvenie
- Vlákna a súbežné prehrávanie
- Kompatibilita s NVDA
- Skúmanie používania v praxi spolu so žiakmi cieľovej skupiny

Ďakujem