Portland State UNIVERSITY

MASEEH COLLEGE OF ENGINEERING

PROJECT PROPOSAL

SUBMISSION DRAFT #2.4

A Block of Code

Senior Capstone Project

Team Members:	*Erebus Labs:*
Nathan BRYANT	Dr. Mike BOROWCZAK
Daniel FRISTER	*University of Wyoming*
Tyler HART	DR. Andrea BURROWS
Jacob MICIKIEWICZ	*PSU Advisor:*
Greg STROMIRE	Roy KRAVITZ

‘

February 12, 2015

Contents

1 Needs Statement

Today's world is one of interconnected technologies, at the heart of which are devices like computers and micro-controllers. As the technological complexity of our world increases we will find our society increasingly reliant on the men and women who know how to operate and control these devices.

However as of today no universally accessible curriculum includes a focus on the knowledge and skills needed to program these machines. We have need of any system that can help children learn and understand these concepts. Additionally, the long term effectiveness of any system will depend on both it's ease of use and it's accessibility across a large range communities, especially those under financial duress.

2 Objective

The primary goal of this project is to create a low cost tactile teaching aide capable of helping a wide range K-12 students learn the basics of computer programming. Our proposed teaching aide is a set of blocks that will allow students to perform basic programming operations and assignments through the arrangement of the blocks. Verification of valid code structures will provide feedback for learning, and control of simple outputs connected to the system can give students a goal for their projects.

3 Background

Currently there are limited methods of teaching younger groups of students about computer programming, and those that do exist often rely heavily on access to computers. This project is meant to produce a learning aid that will function in a classroom setting, doing away with the need have computers present for learning about programming. The use of a visual-tactile system to teach new skills to children has a long and well established history dating at least as far back as Froebel, who believed that freedom to play and experiment was essential to real learning.

In terms of this project the system can be thought of as a, Froebel *Spielgaben*, or a subject specific learning module. The purpose of which is to allow students of any age to be introduced to coding using a familiar process that promotes creativity. Allowing the module to be untethered, separate from a computer, will also allow the teaching of programming without the other distractions available on a PC, and give students that have trouble with abstraction a set of objects to focus on.

4 Marketing Requirements

The final package will be a set of between twenty to thirty blocks typically two or less inches across a side. The set will function as it's own interface device, where the topographical arrangement of elements (blocks) will determine the *program*. The individual blocks must be identifiable as belonging to a specific programming construct-group. At a minimum these groups must include; numbers, variables, operators, and controls.

Additionally , if possible, a block should be capable of indicating what value or function has been assigned to it. Blocks should be easy to assemble, allowing users to experiment with layouts, but provide area specific error feedback and thereby increasing understanding and limiting confusion.

Finally, the set should open source and be sufficiently accessible, both by cost and teaching use, as to promote universal access and encourage development and expansion by others.

5 Risk Management

Identifying potential risks at the beginning of a project can help by allowing a certain amount of mitigating action to be built in to the design process.

5.1 Defining Risks

Risk types:

1. **Scope**; risks of scope are those that cause the project to be delayed, too complex, or trivial. Frequently these are cause by poorly defined project specifications, attempting to do much, or becoming unnecessarily focused on specific details.
2. **Conceptual**; risks belonging here are the products of some failure in understanding a problem or it's intended solution. One example could be attempting to predict position by triangulation but forgetting to consider velocity.
3. **Physical**; risks of this type include problems that manifest physically, most commonly refereed to as *bugs*(software bugs *are* included here.)
4. **Incalculable**; risks of this nature include random and unforeseeable events. Shipping delays, inclement weather, and distributor component substitution are all examples of such risks.

5.2 Mitigating Risk

Action by type:

1. **Scope**;

   (a) Over reaching and scope creep;

   Controlling the scope size can usually be implemented by instituting system of interdepartmental checks, and by setting project goals by stages. The later, infers making the scope such that it satisfies only the minimum project specifications, while the former, in this case infer double checks are done between team and advisor, or other available expert resources.

   In both cases the scope can be added to, but by building a semi-bureaucratic requirement into the process, the chances of adding unrealizable goals to the project have been reduced.

   (b) Built in obsolescence;

   In contrast to point (a) is the risk defining the project so precisely as to reduce the final product's usability. Incorporating mandatory reviews of tasks/ goals at milestones , and use of a check list or pre-scripted questionnaire can help ensure essential functionality has not been overlooked.

   As in the above (5.2 1a) process, by conceiving each stage of the project between milestones as a singular smaller projects, we improve the chances of including new and relevant additions to scope.

2. **Conceptual**;

   (a) Improperly defined problems;

   The methods of limiting these types of risk are very closely related to those of controlling scope. By ensuring clarity of scope, it then becomes easier to dissect the problem as it pertains to project.

   After dissection, it is up to each team member to voice any concerns or confusions. To which as a group, providing an open, non-judgmental forum is imperative.

(b) Fallacious solution sets;

   This type of risk usually, but not always, stems from having improperly defined problems, which we have addressed already. For the cases not covered, a solid system of review and collaboration, combined with weekly meetings will reduce the likelihood of implementing improperly framed or proposed-solutions.

3. **Physical**;

   (a) Mechanical;

   Mechanical failure can be difficult to foresee, however by rigorously checking each proposed design for possible simplifications can help mitigate potential pitfalls. General mechanical design considerations should always consider; least number of [connections, moving parts, seams, joints]

   (b) Electrical:

   As with mechanical, where checks should also include maximum dissipated power, potential bounces, and bit drift.

   (c) Code based;

   Some of the electrical/ mechanical risks can be mitigated with software, however constant review and sections-testing is necessary to ensure that new bugs are not introduced with coded solutions. Before considering a section done, all attempts should be made to break it.

4. **Incalculable**;

   (a) Components;

   While the very nature of this section precludes prediction, there are certain actions that can limit damage done in event of component based problems. Problems may include, random device failure, shipping delays, distributor substitutions, and more.

   i. avoid dependence on non generic components

   ii. avoid, where possible, proprietary technologies.

iii. always check licensing

iv. never order minimums

v. always build in parallel

(b) Events;

The most effective way to mitigate damages done by events like unforeseen weather or closure of services needed, is to have a flexible, organic schedule (see next).

(c) Scheduling;

Designing a system of scheduling, that allows for the most efficient use of available resources can greatly reduce strain on project timelines. For this project we have implemented a generic, *by ticket* system, which allows team members to *bid* on project tasks as dictated by their own constraints. This combined with ensuring that no individual task is longer than the time between meetings (1-week) will help ensure work flow.

6 Current Proposed Solutions

The current solutions we propose below are subject to change, as feasibility and effectiveness will become more clear as prototyping commences. Each level of description is limited to an overview in this paper, as we address each solution further in the *Product Specifications* document. The overviews are given to give enough familiarity to address the *pitfalls* subsection.

6.1 System Level Solution

Our product will consist of identical *blocks*, each with enough digital circuitry to accomplish the next subsection. Blocks will electrically connect with one another via metal-on-metal contact, forming a grid. Some of these metal nets will form a mesh grid (local communication), while some of them will form a global bus (power and I2C-or-similar bus). We will address this network as a *matrix* in this paper.

On the last line of the matrix, Row 1, the user will place the Master Processing block, which is not identical to the other blocks.

6.2 Blocks And Communication

Each block stores its own associated data. Each block stores a *position vector*, which represents where on the matrix the block is located, and stores a *unique ID*, which allows the block to be uniquely addressed on the global bus. Upon initialization, the position vector of each block is unknown.

6.2.1 Determining Location

The process for each block to identify its position vector is as follows:

1. First the positions of each block in column 1 are discovered:
    (a) The processing block passes a

    ```
    <0,0>
    ```

    vector to its northern neighbor via the local bus.

    (b) Until the end of the column is reached, whenever a block receives a vector from its southern neighbor, each block increments the y component of its position vector, emits a message on the global bus containing

    ```
    [position vector, unique ID]
    ```

    then passes its position vector to its northern neighbor.

    (c) The last block on the column increments the y component of its position vector, then emits a message on the global bus containing

    ```
    [position vector, unique ID, END_OF_DIMENSION]
    ```

2. The processing block collects this info in its own local-cache of the matrix.

3. To identify each block in each row:

(a) The processing block iterates over each Column 1 entry in its local-cache. Using the emitted Unique IDs for each entry, the processing block sends a

    ```
    BEGIN_RASTER
    ```

    message to each Column 1 block via the global bus.

(b) When each Column 1 block receives a

    ```
    BEGIN_RASTER
    ```

    message on the global bus, it passes its Position Vector to its eastern neighbor via the local bus.

(c) Until the end of the line is reached, whenever a block receives a vector from its western neighbor, each block increments the x component of its position vector, emits a message on the global bus containing

    ```
    [position vector, unique ID]
    ```

    then passes its position vector to its eastern neighbor.

(d) The last block on the line increments the x component of its position vector, then emits a message on the global bus containing

    ```
    [position vector, unique ID, END_OF_DIMENSION]
    ```

4. Using the messages passed on the global bus, the processing block is able to fill in its local-cache with all of the Unique IDs of the blocks in the network.

6.3 Language, Parsing, Processing

After filling in its local-cache, the processing block can interpret the language in a traditional manner using a Lexer, Parser, and AST-Interpreter. This all happens in software.

6.3.1 Lexing

The processor is coded with an alterable lookup table containing the token associated with each Unique ID. The lexer does a raster scan on the processors local-cache matrix, and for each Unique ID it encounters, looks up the lexical value of the ID in the lookup table. The lexer hands this token to the Parser.

6.3.2 Parsing

The parser is a traditional LR parser that constructs an abstract syntax tree from the lexical tokens handed to it by the lexer. It passes the root of the abstract syntax tree to the interpreter.

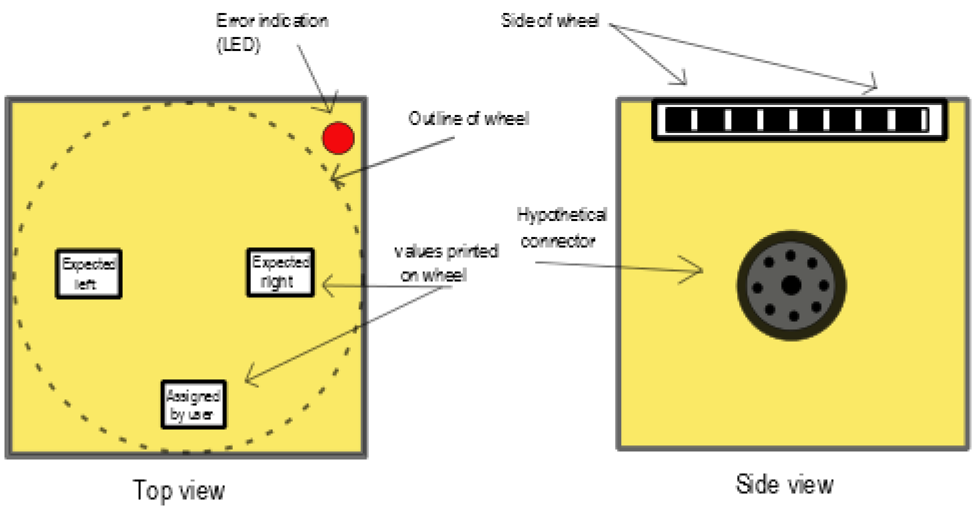
6.3.3 Interpretation

The interpreter descends the abstract syntax tree, performing the actions associated with each node.

6.4 Visual Feedback

Currently the scheme for allowing the user to set the block token (see 6.3.1) to be handed to the parser involves a system of selectable visual cues on the top face of the block. Either a rotary encoder or potentiometer will be attached to a coded wheel that sits under the top face of the block, with a radius such that just the edge of the wheel maybe manipulated by the user through a cut out on the side face of the block (see figures below).

6.4.1 Wheel Encoder Diagrams



6.4.2 I/O

User interface by rotating the wheel to select the desired value (token), this corresponds to a number processed via Grey Code or ADC which is then sent to the lexer for lookup.

When the interpreter has finished compiling the program image, if any errors exist, the control unit identifies which unique ID holds the token that caused it and sends out on the bus;

```
[unique ID]
FLASH_LED
```

7 Pitfalls

Possible pitfalls for the current proposed solution are discussed in this section, and likely mitigation techniques are discussed for each pitfall.

1. Varying path lengths on each bus may result in bit drift and/or clock skew.

   (a) *Mitigation:* Throughput on buses is not likely to be a significant factor in system performance (whereas interpreter performance is), so slower baud rates are a cheap solution.

2. A user can write an infinite loop

   (a) *Mitigation:*User programs will run in their own process which will be killed whenever blocks are rearranged (then the new program will start in a new process).

3. Manufacturing uncertainties: Connectors do not align; enclosures are not square. Magnetic interference.

   (a) *Mitigation:* Expose connectors for proof-of-concept revisions to demonstrate functionality.

4. Communication algorithm was inherently flawed.

   (a) *Mitigation:* We plan to release our first revision at the halfway mark before "product launch". Reimplementing the Location-Vectoring algorithm using a tray as a fixed-grid reader of Unique IDs would be trivial from a design perspective.