Maseeh College of Engineering

Project Proposal

Submission Draft #2.1

A Block of Code

Senior Capstone Project

Team Members:
Nathan Bryant
Daniel Frister
Tyler Hart
Jacob Micikiewicz
Greg Stromire

Erebus Labs:
Dr. Mike Borowczak
University of Wyoming
DR. Andrea Burrows
PSU Advisor:
Roy Kravitz

January 27, 2015

Contents

1 Needs Statement

Todays 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 more and more 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

Creating low cost tactile learning tool to instruct primary school students in the basics of computer programming is the primary goal of this venture. This tool will allow students to arrange blocks to perform basic operations and assignments. 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 project.

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 aimed to produce a learning aid that will function in a classroom setting, doing away with the need have computers present for learning. 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.

In terms of this project the system can be thought of as a, Froebel *Spielgaben*, a subject specific learning module. The purpose of which is to allow students of any age to be introduced to the subject 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 about two inches along the longest edge. 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 it should be included, if possible, on each block, what the current assigned value or function the block has been assigned to. 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 caused 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. Shipment 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 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 a mandatory review of tasks and goals at milestones can help. As in the above process, by conceiving each stage of the project between milestones as a singular smaller project, 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, in which case have been addressed already. For the cases not covered, a solid system of review and collaboration combined with weekly meetings will reduce the likelihood of improperly framed proposed solutions being implemented.

3. **Physical**;

(a) Mechanical;

Mechanical failure can be difficult to foresee, however by always checking proposed designs 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 be ...

(c) Code based;

Running out of time before class feel free to comment on what should be here

4. **Incalculable**;

(a) Components;

While the entire 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

(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.