**ILC Narrative for: VR Algebraic Math Manipulatives**

**Summer Quarter 2017**

Cosmo Inclan

***Introduction****:* This narrative is meant to give an overview of how the idea for this ILC developed: what my plan was when I first started it, what I’ve learned along the way, challenges I’ve run into, how I’ve approached my primary learning objectives and what I’m going to do moving into the future, and lastly, what I’m submitting. The journey has been far from linear; it began with ideas about how to use virtual reality as an alternative teaching tool, and eventually led me to embark upon my own creative project, one that is still in-progress.

**Part 1: Thought Process and Equipment Donation**

This past year I had been toying with the idea of creating an outside-the-box algebra teaching program for VR (A field that is just now expanding out of novelty to the point that it has its own branch of research.) But before beginning, I chose to answer the following questions:

1. What problem would I be solving?

I believe math should be taught in a way that inspires confidence and interest. When I was first introduced to Algebra I had very difficult time learning it in the traditional non-visual way. It was only till a tutor tried a different visual puzzle oriented approach that I finally got it. This was so inspiring that mathematics and science became the focuses of my time in academia. I have heard, taught, and observed many other students who have similar problems to the ones I had. Thus, I want to invent a program for them, to help them improve their understanding of Algebra and reach them in a manner they haven’t been introduced to yet.

1. Who might be my audience?

My audience will be students who are visual learners and anyone who learns better through doing rather than rote knowledge assimilation. Our schools teach math in a manner that places more priority on rote learning, and that doesn’t work for these types of students. And as an aside, the US consistently rates lower than many other first world nations in mathematics. Evidently, something isn’t working. My hope is that the program will address the needs of these students and give them access to a tool geared toward their learning style.

Should this program prove to be effective, I can also see it being used as an additional aid/tool for the access services departments -- for use by students with learning disabilities, or any student interested in learning math an alternative way to the norm (which is kind of Evergreen's mantra -- the alternative creative thinking model). Anecdotally, I was able to get in touch with the TESC access service director, and she seemed pretty enthused by the idea of this project but told me there are a lot of hoops to jump through to make software access service compliant (the University of Washington has pages of information on the topic) so that would be more of a stretch goal if and when I decide to recruit team members.

1. How will I answer whether or not there is any positive outcome?

I will recruit volunteers from the Evergreen State College, and potentially online communities such as GitHub, so I can measure the effectiveness of my program. Data will be gathered using custom Unity 5.1 Analytics event triggers. To account for ethics while information gathering, all analytics gathered will be disclosed to the program user prior to the program being launched. The program will begin by testing each user’s algebra skills, so I can get insight into a particular user’s performance before they use the program, and then testing them after the program, to demonstrate improvement. The tests will be reasonably difficult, so even people who have taken algebra may not get 100% on the first test. If we see improvement overall, we can then call this a “positive outcome”.

1. What might be a potential 6 month, 1 year, 2 year, etc. rollout plan?

*6 month mark*

> Program has yielded a stable functioning instance that should be in a testable state

> Enough data has been attained to determine efficacy or inefficacy

> Recruitment of additional team members to project to assist with modules like device optimization, additional curriculum updates, and aesthetic diversification

> Program holds up to VR best practices

*1 year mark*

> Program has gone through multiple iterations based on user feedback

> In an optimized state for review and in-person training of interested educators and third parties

> Creation of official user documentation booklet (including suggestions on using the material alongside a standard algebra curriculum)

*2 year mark*

> Program is offered as free and open source

> Move towards a calculus based VR curriculum

1. Who might I potential partner with?

Geogebra (they’re already implementing math concepts into VR, and outside the box algebra puzzles might be the next step towards increasing their curriculum diversity)

Microsoft Schools program (possibly)

Google Expeditions (it’s focused on virtual tours now but after converting the program to mobile, although I’m designing it to be used in congruence with motion controllers, it could be used with Google Cardboard)

MS Minecraft (Most of the puzzles should be able to be represented using built in components (using pre-coded programming, Redstone, switches, etc.) This would allow for multiple side by side puzzles and group collaboration. This might even assist in counteracting the notion that VR is a fundamentally isolated experience.

1. Are there real life applications for this idea?

Assuming all goes well and the program is an effective algebra teaching alternative, the next step would be to integrate the program into a school curriculum. This will require willing adoptees (mainly schools and educators) in addition to finding solutions to the challenging roadblocks VR faces along the way to mainstream acceptance. And there will be the cost of equipment. However, Unity can be used to address both the high and low end VR market (for mobile devices and PCs) so eventually, once affordable quality VR options emerge in the market, VR software opportunities geared toward education in mathematics will have a stronger foot in the door.

With all that answered, I began sourcing means of obtaining the necessary VR equipment. I had begun the process of writing a grant in an earlier part of the year, but decided to shelve the grant for a more direct approach. I decided to start networking with a variety of VR professionals at Oculus and HTC Vive (the major VR companies located in Washington) to see if there was any means of obtaining a direct grant or gift-in-kind. After some trial and error, I managed to get a conference meeting setup with HTC in Seattle. From there, after pitching my project idea, HTC agreed to donate a complete VR headset and workstation to Evergreen. Thankfully, I also managed to negotiate for a dedicated room for the equipment. This means that going into the future, faculty and students will be able to use the hardware in their own projects.

**Part 2: Explorations in Unity3D and C# and Google SketchUp**

When the equipment finally arrived in full, I went about installing it all myself, and learning the basics to setting up a development environment in Unity3D, a cross-platform game engine development environment that I decided to start learning (due to its functionality with the VR product donated, and its support for lower end systems like mobile devices).

I learned how to calibrate the equipment – the two room sensors, headset, and twin motion controllers, and setup a navigational mesh that could be walked on (my notes on how to do this are included in a guide within my GitHub directory.) This part took a fair amount longer than I expected because there was little to no documentation that came with the equipment (owing to it being targeted to the enthusiast consumer game and entertainment audiences), but then I discovered that the official documentation for Unity had a VR section, so after reading that, I discovered free resource packs within the Unity in-app content store and on GitHub that gave me what I needed. The *SteamVR plugin* (by Valve Corporation) – to link the hardware with Steam’s room scale VR experience and utilize common prefab, camera, and model assets, the *Vive-Teleporter* (by Adrian Biagioli) -- a framework for Unity3D that implements a parabolic arc teleportion mechanic for the motion controllers– lessening the nausea that occurs when walking through a VR environment, *The Lab Renderer* (by Valve Corporation) to provide supported shaders and lighting scripts and dynamically maintain framerate and rendering properties in VR, and *ProBuilder Basic* (by Procore),an extension and modeling tool.

At the same time, I still had a lot to learn in regards to the utilizing C# in Unity. I began reading C# tutorials, and creating mockup models of what I expected the program to look like using Google SketchUp. I also began a set of experimental scripting projects within Unity, to teach myself to interact with prefabs, models, materials, control the camera(s), snap objects, adjusting lighting, and how public variables could be used as real-time controls within the inspector toolkit. I referenced the official Unity documentation on scripting (which includes a number Unity-centric C# tutorials, and The C# Programming Language 2nd edition manual by Anders Hejlsberg, as well as two exceptional Unity tutorial channels on YouTube (Renaissance Coders, and AbhiDemkeys.)

Over the course of the summer, after much iteration of the game-environment I had roughly sketched out on paper, I finally settled on a game-table design that seemed to fit the algebra (for at least the first third of the program – which focuses on isolating variables). I am well aware that this game-table is far from perfect, because only had a chance to model one example problem from start to finish, but it works for now. I also created three unity sample projects that I plan to eventually tie together into the final program: one that tracks game object transformations and changes color upon a 100% transform, another that instantiates at various speeds or instantaneously, and one that builds upon the instantiation methods and ties in a custom color mechanic for randomly instantiated game objects.

**Part 3: Challenges**

One of my biggest realizations throughout this ILC has been accepting that, having never undertaken a project like this before, I had drastically underestimated how much time it would take, and that it would ultimately become an ongoing project. I ran into everything from import-export scaling issues with models, to models not having unique local and global coordinates, and multiple times, and I continually found the need to completely scrap a game-table design due to lack of clarity (which I knew was going to be important, due to the highly visual nature of the program.)

There have also been a number of more mentally tasking struggles related to the coding that research has yielded little progress on so far, despite extensive searching and inquiring at communities like Stack Exchange. Specifically, I’m still unsure of how to create my own symbolic math evaluator (much like the kind that Wolfram Alpha and Symbolab employ.) There are a fair number of libraries to available, but if I determine that route is my best option, the trick will be learning which one can help me, since not all have the full functionality (or knowledge base) that I need or be open source (certain libraries are premium, and one of my goals is to ensure the program remains open-source.) One possibility I found may be to use Math.NET Symbolics, an open source computer algebra library for .Net (which can be also used in C#,) but I only discovered it recently, and haven’t had the time to fully test its functionality or read its documentation.

**Part 4: Learning Objectives**

At least as far as meeting the first objective – to learn to create algebra teach program, I wasn’t able to. I don’t have a finished project – and I’m still unsure of how all the various parts of my project will tie together, but I am continuing to make progress. I’m also not yet sure it will (ever) be a viable teaching tool for students or anyone wishing to learn algebra because I simply don’t have a means to test what I have. In regards to the latter objective, to create Google SketchUp walkthroughs, I do feel that I managed to meet this. I have an extensive and organized library of game models, objects, and a number of videos that walkthrough manipulative puzzle examples along with documentation.

Moving forward, my plan for this project is to approach the code with greater structure and planning, fully mapping out the logic on paper, so that I have a better grasp of where to start. I want to get the math evaluation solved first, whether it be through the use of array-rules, tag-associations, or via a free library.

**Part 5: Submission Materials**

GitHub Directory: <https://github.com/cinclan/vr-algebra-manipulatives>

My submission is being slowly but surely uploaded to a GitHub directory. The directory contains game models, links to video walkthroughs, and code snippets from various Unity test samples.

All Unity sample project videos have their relevant code snippets linked in the video descriptions.

Ex: color assignment && cube instantiation at recorded places

<https://youtu.be/MForQmJJo_>

SketchUp walkthrough videos are meant to show the player walking through the example problem A + (B + C)/X = Y, solving for variables B and X. (B represented by a yellow cube, X represented by a light-blue cube)

Ex: v7\_1 Isolating X\_var (blue cube), table iteration 7

<https://youtu.be/djWiiLsppew>

An overview of the repository hierarchy structure is shown below:

