Summation
May 2010, pp. 1-4
http://ripon.edu/macs/summation

Android Application Development and Implementation – 3 Dimensional Tic-Tac-Toe

Danial C. Hanson
Department of Computer Science – Ripon College

***Abstract*—This document discusses the Android Mobile Phone Operating System, and the processes involved with developing, implementing and deploying an application for it. Related subjects also covered include monetary opportunities, and androids relationship to java and xml.**

***Key Words*—Android Operating System, *Eclipse*, Mobile Phone, XML**

I. Introduction

The idea of installing a robust operating system on a mobile phone the size of a deck of cards would have been preposterous not long ago. Today it is commonplace and gaining ground. Android is Google's Open Source Mobile Software Environment. It consists of a Linux-kernel based operating system with the underlying code written in C and C++. It uses the Dalvik Virtual Machine to implement programs written in the 'Android' language which is a user-friendly combination of Java and XML. It's important to note that Android is not a hardware platform. For instance, there is no such thing as an 'Android Phone' only 'phones that can utilize the Android Software Environment'.

Developing applications for the Android Mobile Phone Operating System does not require as much preliminary work or in depth learning as one might initially assume. A working knowledge of programming and a willingness to try new things are the two main prerequisites. However, as with anything, it must first be understood just what it is that one is developing for.

Android is not difficult to pick up for anyone with a working knowledge of programming (this is especially true for users with prior java development experience). It is also very affordable to market, and holds many entrepreneurial opportunities for developers interested in taking advantage of an entirely open and new market. The development kit is free of charge and a twenty-five dollar license grants any developer freedom to publish as many applications as they would like on the world-wide 'Android Market'. Monetary opportunities consist of selling an application outright, offering a free 'lite' version and charging for the full version, or working with a program such as 'AdMob' which monopolizes a small portion of the applications screen space, but pays for the opportunity to do so.

The goal of our project was to create a three-dimensional four by four by four tic-tac-toe application with an attractive and accessible user interface and an intelligent opponent. At the time of this project's inception, we had already begun development of such a program in C++, so our plan was to complete said program development, and work to make the finished product available on Android compatible devices by converting it to the Android language.

This article will discuss the intricacies of Android application development and implementation including the *Eclipse* Development Environment, initializing a new Android Program, the steps carried out in turning a new 'blank canvas' into a fully functional Android application and finally how to test that program and make it available to use on Android capable handsets. We will also discuss the processes involved in the development of a 3-Dimensional Tic-Tac-Toe program for Android and things to watch out for when developing Android Apps.

II. The *Eclipse* Development Environment

When developing the Android programming language and a corresponding development environment, Google worked closely with the existing Java Software Development Environment *Eclipse*.

Currently in their 'Galileo' release, *Eclipse* is an open source java software development kit, conceived in 2006, which has many useful features and plug-ins including the free Android Development Tools plug-in developed by Google to allow greater ease of use to Android developers using the *Eclipse* environment.

The Android Development Tools plug-in "extends the capabilities of *Eclipse* to let the developer quickly set up new Android projects, create an application UI, add components based on the Android Framework API, debug their applications using the Android SDK tools, and even export their signed (or unsigned) APKs in order to distribute their application"[1].

This plug-in as well as instructions on installing all Android applicable *Eclipse* elements (and even *Eclipse* itself) can be found on the Android Developers webpage: http://developer.android.com/index.html.

Work presented to the college March 23, 2010. This work was supported in part by the Ripon College Math and Computer Science Department.

Danial Craig Hanson hopes to find a job in software development/engineering, specifically in the realm of java and/or android application programming. (phone: 715-220-7593; e-mail: dch4nson@gmail.com).

Once installed, *Eclipse* and the free Android extensions provide the user with full access to the Android function calls, variable types, packages, and debugger; as well as a fully customizable virtual phone emulator, or the ability to use an Android compatible handset, physically connected to the development machine via USB, to test completed applications prior to publishing.

III. Developing An Android Application

Developing a successful Android application consists of 5 main steps: new project creation, interface development, logic and functionality development, debugging and testing, and publishing and installation. The following section will describe the intricacies of each of these steps in detail, and attempt to educate the reader on the finer points of constructing a simple android application. The material discussed within section III will assume the user has installed and is utilizing the *Eclipse* development kit discussed in the prior section.

A. New Project Creation

Creating a new project in *Eclipse* is as simple as selecting 'File' → 'New' → 'Project' and then selecting 'Android Project'. At this point *Eclipse* will ask for a small number of items: the titles and necessary implications of these items is as follows (as gleaned from the Android Developers' Hello, Android example page [2]):

--*Project Name*: This will be the project's name with *Eclipse* and the name of the directory on the development machine that will contain the files.

--*Application Name*: This will be the human readable title of the application, and the one that will accompany the finished application on the handset.

--*Package Namespace*: Following the same guidelines as packages in Java, the package name is the namespace under which all code will be maintained and the stub activity will be generated. It is imperative that this remain unique across any and all packages installed on the given Android system.

--*Create Activity*: Although optional, this is almost always necessary; this will be the name of the class stub that the plug-in will generate as a subclass of Anrdroid's activity class.

--*Min SDK Version*: This specifies the minimum API level required by the application. For instance, if it is set it to 1, it will work on all level 1 devices and higher. If it is set it to 2, it won't be installable on level a 1 device; but the additional functionality made available with level 2 may be utilized in return for the lack of backwards compatibility.

Upon completion of these fields, clicking 'finish' will complete the project construction, allowing the user to begin application development.

Eclipse will create the application's package and make its resources available to the developer in the navigation panel of the program with the following folders/files:

--*src*: This folder contains the java file in which the logic and functionality coding will reside (discussed in subsection C).

--*gen*: This folder contains the R.java file which is automatically generated by *Eclipse* to provide the java file in the 'src' folder with the proper memory address of any necessary variables declared in the layout file.

--*Android X.X*: This folder will contain the android.jar file corresponding to the specified minimum SDK version. This will include all the built in android functions and variables that may be referenced in the logic.

--*Assets*: This folder is seldom used, but essentially can hold an unstructured hierarchy of files to be referenced later as as raw byte streams[3].

--*Res*: This folder contains the resources, including images to be used in the application, and the *.png icon file. This folder also contains the *.xml layout files, where user interface development will take place.

--*Android Manifest.xml*: This is an XML file that is automatically generated, but can allow access to many of the items specified at the inception of the project including Project and Application Name, package namespace, etc…

--*default.properties*: This is another auto-generated file that will not require any form of user interaction but must be constructed for the sake of the application.

B. Interface Development

Once the program template has been established, the next logical step is to design the user interface. This is done primarily in the layout.xml file found in the 'Res' folder. The coding will be done in XML. Once opened the layout.xml file will allow the user to 'preview' the layout via a tab at the bottom of the opened file rather than emulating the program every time they want to see how a change in code impacts the visual layout. This view is entirely customizable based on the developer's specifications and requirements.

There are four layout options to be utilized by the developer, each offering a different approach to layout initialization:

--*Frame Layout*: This layout is simple but virtually useless for any practical application as it stacks all icons in the upper left corner of the window, not allowing any further input on the developer's part.

--*Linear Layout*: This layout will allow the developer to align widgets either vertically or horizontally and assign 'weights' to them to adjust their size relative to other widgets within their row or column.

--*Table Layout*: This layout is row and column based and allows the developer to grow or shrink those rows and columns based on their own personal needs.

--*Relative Layout*: Allows the developer to assign absolute sizes to each widget and position them relative to the other widgets and the edges of the screen.

Once the developer has selected a layout, the next step is simply to design the user interface using the XML language until they've constructed what they feel to be an appropriate and easy to use UI.

C. Logic and Functionality Development

When coupled with the User Interface Development, the Logic and Functionality Development portion of the application development completes the ‘main phase’ of the project. Whereas with the user interface the developer designs the visual elements of the application, in the logic and functionality portion, they assign functionality *to* those visual elements. For instance, if a button is created in the layout file, what the application is supposed to *do* when the button is clicked will be defined in during the logic and functionality development phase.

The logic is written in the very user friendly java language, and the corresponding file can be found in the ‘src’ file. All common java functions can be referenced and utilized, as well as all Android functions contained within the API specific *.jar file in the ‘android x.x’ folder.

Any widgets defined in the layout.xml file may be referenced from within this code, and any necessary local variables for logic and calculation may also be freely initialized and utilized.

D. Debugging and Testing

As any developer knows, once the project is near completion, it must be tested to ensure that everything operates as expected. *Eclipse* makes this process very painless by incorporating a fully customizable virtual emulator. This can be setup exactly to the developer’s specifications, or based on one of many included templates setup to emulate any of many existing Android compatible phones.

Once an emulator has been initialized, the developer must simply compile and run the program and the emulator will launch, allowing the developer to see how the application operates on a ‘faux-phone’. Another option available to the developer is to simply connect an Android compatible phone to the development machine via a USB cable and run the program directly on that handset. This allows the developer to actually use the program on a physical handset and provides very useful feedback.

E. Publishing and Installation

The final step in successful android application is installing and publishing the application. Installation can be accomplished in one of three ways: emulation, direct file installation or market installation.

The first and easiest (although not practical for multiple devices) is to use the emulation method described above with the target handset. By connecting a handset to the development machine and running a completed application on said handset, the application will automatically be installed.

The second option is to locate the *.apk file in the project directory on the development machine’s hard disk and transfer it to the target handset’s memory. Once this is accomplished, the application can be installed by utilizing the free market available application *Apps Installer*.

The final method is to publish the application to the android market and install it from there. This is the best option for mass publication and profit gaining, although the initial twenty-five dollar investment is to be considered if this is the avenue the developer chooses to take.

Figure 1. Tic-Macs-Toe live in action on the developers’ test phone, a T-Mobile MyTouch 3G (a stripped down version of the HTC Magic).

These five steps encompass the basic procedures involved with application development and implementation for any basic android application. In the following section we will discuss the intricacies involved with the development of the three dimensional tic-tac-toe application.

IV. 3-Dimensional Tic-Tac-Toe

Remember from section I that the goal of our project was to create a three-dimensional four-by-four-by-four tic-tac-toe application with an attractive and accessible user interface and an intelligent opponent. We began by completing a program that we had already begun in C++ with the hopes of porting it into Android.

The first step to creating the android application was to develop the user interface. To accomplish this, we utilized the ‘relative’ layout format. We created un-clickable and invisible buttons with a width of 1px to serve as ‘center posts’ for the 64 clickable spots. The first of these posts was centered horizontally and aligned to the very top of the window, with two clickable buttons to the left, and two to the right. We then created another post located directly below the first, again with two clickable buttons to the left, and two to the right. This process was repeated until all 64 buttons were in place. The top row of each of the four ‘sections’ of buttons was thinnest, with each of the three successive rows growing slightly in width to give the impression of a three dimensional board.

The clickable spots consisted of a 3-D array of button widgets in the [x][y][z] format where [0][0][0] would be the top left corner and [3][3][3] would be the bottom right corner. Finally, a ‘reset’ button was placed at the bottom of the screen that would serve as just that and allow the user to begin a new game upon completion of one.

Next, another layout had to be created to serve as a text pop-up box. This layout consisted of a simple TextView widget, the text of which could be edited from within the java logic file, allowing it to be referenced anytime a popup textbox was necessary.

The logic coding was fairly simple when the time came as it was simply a matter of copying the completed and fully functional C++ code into the java file and changing the syntax as necessary to accommodate the Android language and the new button array.

In the finished product, the user would play blue versus the computer who would play red. Once a square was selected, it's on click listener would be turned off, making it unselectable, and the computer would use alpha-beta pruning with the mini-max algorithm to determine its next move. Once this had been done, the computer would check to see if a win (four in a row in any direction) or cat's game had occurred, and if so would display a message stating such and turn off *all* button listeners except the reset button which when clicked, would clear all buttons, reactivate all button listeners and begin a new game.

The majority of development went as smoothly as possible. The one major incident we encountered involved slow runtime due to memory restrictions. While the computer used to develop the original C++ program had 2GB of RAM and a 2.8Ghz processor, the phone only had 188MB of RAM and a 528Mhz processor, turning an instant computer response into an approximately two-minute wait when implemented on the target handset.

Memory restriction is the main bottleneck of mobile phone operating systems at this time and must be carefully considered when developing programs. In our case, we were lucky enough to be able to speed the program up without having to entirely rework the algorithm. In the C++ version, the algorithm was set to check '4-ply' meaning it would look at every possible combination of the next 4 moves, or, on an empty board 16,777,216 possible move combinations. We dialed this back down to 2, which meant a maximum of just 4,096 checks and the computer's response became instantaneous, yet it was still intelligent enough to be competitive with the user and in most cases still win the match.

V. Conclusion

Google's Android Operating System and its corresponding development environment have proven to be as user friendly and powerful as the search engine that popularized Google to begin with. Application development is a breeze, even for those new to the language, and the opportunities are nearly endless for freelance developers and entrepreneurs. I would like to personally encourage anyone with an interest in software and/or application development to take a look at Android as I truly believe the time will be well spent. The full code for the completed 4x4x4 Tic-Tac-Toe program can be found at the URLs listed below.

4x4x4 Tic-Tac-Toe Code Resources

[1] http://ripon.edu/academics/macs/summation/2010/supp/tix-java.pdf
[2] http://ripon.edu/academics/macs/summation/2010/supp/main-xml.pdf
[3] http://ripon.edu/academics/macs/summation/2010/supp/dialog-xml.pdf

Recommended Further Reading

[1] Reto Meier, Professional Android Application Development Wrox, 2008.

[2] Frank Ableson et al, *Unlocking Android: A Developer's Guide* Manning Publications, 2009.

Acknowledgment

I would like to personally acknowledge my partners in this endeavor Brad Bogenschutz, and Jerry Hardacre, as well as our faculty mentor McKenzie Lamb and the entire Ripon College 2010 Math and Computer Science Department, without whom this project never would have taken place. Special thanks is also due to the fine people at Google for developing such a user friendly software environment *and* great learning tutorials and reference material for the beginning Android developer.

References

[1] http://developer.android.com/sdk/*Eclipse*-adt.html
[2] http://developer.android.com/guide/tutorials/hello-world.html
[3] http://groups.google.com/group/android-developers/browse_thread/thread/879d4d9545f27b26/6032e4fe6941140c?lnk=raot&pli=1

Danial C. Hanson was born in Shakopee, MN in October of 1988. He graduated from Clear Lake Jr/Sr High School in Clear Lake, WI in 2006 and began attending Ripon College in Ripon, WI in the fall of 2006 where he is currently pursuing a Bachelors degree in computer science (tentative graduation May of 2010).

He has been employed by Ripon Medical Center in Ripon, WI as a Helpdesk Analyst since March of 2009 where he plans to continue employment until the fall of 2011, at which point he hopes to begin post-graduate education to obtain a Masters of Business Administration at a yet undecided institution.

Mr. Hanson is a member of the Sigma Chi fraternity and the BSA where he attained the rank of Eagle Scout. He enjoys spending time with his family, playing guitar, and playing with his Black Labrador 'Monty'.

Summation
May 2010, pp. 5-8
http://ripon.edu/macs/summation

Artificial Intelligence: Implementing 3D Tic-Tac-Toe in C++

Bradley D. Bogenschutz
Department of Mathematics and Computer Science – Ripon College

***Abstract*—This paper will discuss how we implementated an intelligent game of 3D tic-tac-toe in C++ using the minimax algorithm and alpha-beta prunting.**

***Key Words*— alpha-beta pruning, artificial intelligence, heuristic, minimax, ply.**

I. Introduction

IMPLEMENTING the game of 3D tic-tac-toe in C++ brought upon challenges. In this paper, our group's implementations decisions for a 4x4x4 version of tic-tac-toe will be discussed along with support of why we chose them. The techniques used to generate an intelligent version of this game have been described in the first paper of this series. These techniques include minimax alorthim, alpha-beta pruning, and heuristics. However, the focus will change from what these techniques are in a general sense to how they specifically affect implementation decisions for our program.

The implementation of our program can be broken into two main parts: creating an intelligent game and C++ coding. The implementations of creating an intelligent game of 3D tic-tac-toe are divided into sections. These sections that will be discussed are the future of the board, the heuristic, the minimax algorithm, and alpha-beta pruning. On the other hand, the implementation of the C++ code is less complex then making the program intelligent, but without it there would be no program. So, that section will discuss ways we utilized C++ code to display the board and represent the board. These two main parts talk about the implementation of our specific program, but an understanding of the game of 3D tic-tac-toe is needed. So, the paper will begin with information about what 3D tic-tac-toe is in order to acquaint the reader with our program.

II. 3D Tic-Tac-Toe

The 4x4x4 tic-tac-toe game is very similar to traditional tic-tac-toe but there are more possible places to make a move because of the extra dimension. The goal in playing a 4x4x4 game of tic-tac-toe is to be the first player to get four pieces in a row. The first to get four pieces in a row is the winner.

A. 4x4x4 Tic-Tac-Toe

The game of 4x4x4 tic-tac-toe can be visualized as four 4x4 grids stacked vertically on top of one another. The four grids allow for more possible wins than the traditional game of 3x3 tic-tac-toe. The number of possible wins is greater because instead of being able to win by filling all possible moves in a vertical, horizontal, or diagonal line on a grid, users are able to make those moves plus the similar lines but this time going through each grid. Examples of every possible move can be seen in figure 1 if you rotate each board and shift the line across the board. Not every possible move can be shown in a simple image because there are 76 possible wins. With the 76 possible wins there are 64 possible places to make a move in order to generate a win. The number of possible moves and wins will be reduced every time someone makes a move. The reduction of possible wins every time someone makes a move makes this an interesting game to program intelligently.

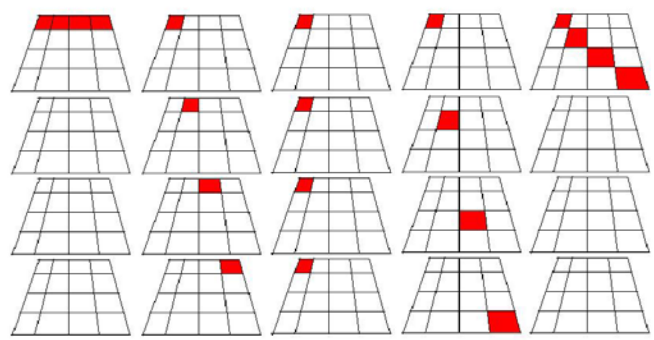

Figure 1. Illustration of possible wins in 4x4x4 tic-tac-toe.

B. Why a 4x4x4 Board Size

Why would we choose to make a computer game of 4x4x4 game of tic-tac-toe when the traditional tic-tac-toe board is a 3x3 grid and not a 4x4 grid? The reason we chose to not go with a 3x3x3 board is because an advanced player will have won the game after both players made a combined total of seven moves. This means if we wanted to create a program to play a 3x3x3 version of tic-tac-toe we would have it go through an algorithm that looked at the user move and made its move without using any artificial intelligence. This advantage to an advanced player is greatly dependent on the

Work presented to the college March 24, 2010 as the second paper in the *3D Tic-Tac-Toe* series.

B. D. Bogenschutz is pursing his baccalaureate degree at Ripon College, Ripon, WI 54971 USA (e-mail: bogenschutzb@ripon.edu).

fact that a placement of a piece on the center most spot of a 3x3x3 board would eliminate approximately half the possible wins for the other player. For this reason, we chose to consider a 4x4x4 size board for our program. The 4x4x4 tic-tac-toe was proven that an expert player can force a win. However, it is still challenging to an average player.

We did not rule out a 4x4x4 tic-tac-toe board size for our program but we did look into creating a game using a 5x5x5 tic-tac-toe board size. The game of 5x5x5 increased the possible moves from 64 in a 4x4x4 board to 125 possible moves. Because of the increase in possible moves, the game was no longer to simple for an advanced player. However, the problem is now trying to program an intelligent game for this board size. When we program an intelligent game of 3D tic-tac-toe the computer looks at the future of the board by generating all possible future moves up to a certain number of moves ahead of what the board is currently. The searching and generation of the future moves on a given board grows exponentially when the player looks more moves ahead. However, looking more moves ahead will generate a better move because the player can predict the future better. By taking a look at 4x4x4 board verses a 5x5x5 board you will notice a large difference in the number of possible states of the board the program will generate. When only looking three moves ahead, a 4x4x4 board will generate roughly 262,000 future board states compared to roughly 1,953,000 future board states for a 5x5x5. This number is greatly larger and affects the runtime and memory consumption needed to run the program. With this being said, we would run into problems generating an intelligent computer move. Thus, we decided to create the computer game of 3D tic-tac-toe to be of a 4x4x4 board size.

III. Implementation Decisions to Achieve Our Goals

To reiterate the goals of our program, our program should look at the future of the board in order to make a more intelligent move, minimize the amount of time it takes for the computer to process a computer move, and reduce the amount of memory consumption. Our goals will be greatly affected when programming the future of the board, implementing the heuristic, implementing the minimax algorithm, and by alpha-beta pruning.

A. Future of the Board

Looking farther into the future of the game is done by generating all possible moves for the next user and then the possible moves for the next user from the previous moves and so on. Each possible move can be described as the state of the board or a node in a tree. The states or nodes represent a current or possible configuration of game pieces on a board. However, the farther you look into the future, the more intelligently the move is. This is because the computer assumes that, with its heuristic, each move is the best possible move for either the computer or user at that current state. So ideally, I wanted to look as far ahead as possible to make the best move.

However, before we look into how far ahead we can look we must understand how we refer to the future of the board. When implementing the future of the board the computer will generate a tree, an example tree can be seen in figure 2. This tree will look ahead a fixed number of plies. A new ply is created every time the computer looks ahead another move. For example, if the computer is looking four moves ahead then the game is a four ply game. The game must look ahead an even ply. This is because we want the computer to calculate the best possible move when it is the computers move for the next state. Thus, our program can be of two ply, four ply, six ply and so on. As a programmer the game should be coded to look as far ahead as possible but maintain a reasonable response time for a computer move after the user makes a move.

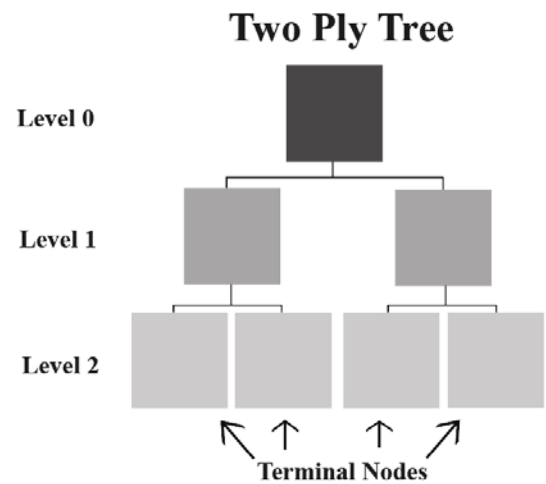

Figure 2. Example of a simple tree.

We chose to go with a four ply game. The reason we went with a four ply game instead of six ply game is because of an issue with time to generate the next computer move. When generating the next move, a tree that looks ahead four ply compared to one that looks ahead six ply will have a notice difference in response time. This is due to the fact that the number of possible states that need to be generated and taking into consideration grows exponentially when going from one ply to the next ply. A four ply game will have to generate and search through roughly sixteen million states which are a lot less then that of a six ply game which generate and search through roughly 68 billion states. Thus, our program looks ahead four ply.

There is a problem with coding a program that looks at the future of the board. The problem arrives when the board becomes almost full. Since the heuristic is calculated at a terminal node, the bottom of tree, the program can return a move of position -1. This is a problem because each position on the board is represented by a numbers between 0 and 63. If this occurs, the heuristic is calculated for the current state and the computer's next move is made based upon the calculation of the current state. This move is still intelligent because the errors will only occur when the computer cannot look 4-ply ahead and when the computer realizes the game is a draw. So, there is no need to look far into the future of the game in order to make an intelligent move.

B. Heuristic Implementation

The heuristic is calculated to find the best possible move at the current state of the game. Our heuristic is calculated by searching through each of the 76 possible wins, then counting the number of user and computer pieces in that possible win, and finally it plugs the counts into an equation and updates the pieces affected by that possible win. The equation is calculated for each piece on the board and can be written as

Heuristic =
((10000*nolw3cp)+(100*nolw2cp)+(10*nolw1cp)+(1*nolw0cp)) –
((10000*nolw3up)+(100*nolw2up)+(10*nolw1up)+(1*nolw0up)).

The nolw#cp stands for the number of lines with # computer pieces. Simular with nowlw#up, except it is user pieces instead of computer pieces. In figure 3, I have simulated part of a game of red(opponient) and green(home) pieces and all of the blank spaces will have a heuristic calculated for. When calculating the heuristic for piece number one, you will get a heuristic of 200 by using the equation above. You can also calculate the heuristic for piece number two and number three. Number two has the heuristic of -80 and number three has the heuristic of -20. The program will then search through the entire board to find the best move. In this case with only three heuristics calculated, number one would have the best chance of winning. This is a very affective way of calculating our good heuristic. The program is very simple and involves very little runtime compared to other ways of calculating a valid heuristics.

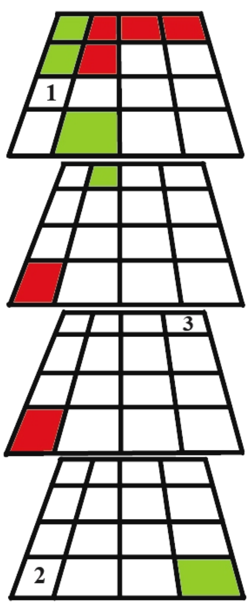

Figure 3. Illustration of a simulated game.

C. Minimax Implementation

Minimax algorithm implementation is implemented in our program by recursion. In order to implement the minimax algorithm you need generate a tree with all possible moves. The problem with generating a tree is the amount of memory needed to store the information. Since our program is a four ply game we will need to generate a tree that has roughly sixteen million nodes. Storing all sixteen million nodes would become a problem. Therefore, we chose to represent the tree by using recursion. Using recursion on a four ply tree will allow for the computer to only store at most five nodes in memory at one time. This is a lot smaller then storing about sixteen million nodes into memory at one time. The use of recursion allows the program to generate possible moves down one branch of a tree until it reaches a terminal node. When the program reaches a terminal node it calculates the heuristic and then returns the best move to the previous node. The previous node then calls the next node until a terminal node is reached. This process repeats until the computer has looked at all the tree's nodes. At that point, the computer will know the next best move for the game and will make that move.

D. Alpha-Beta Pruning Implementation

Alpha-beta pruning is a way of eliminating unnecessary nodes. The elimination of the unnecessary nodes will reduce the amount of runtime because not all the nodes will have to be looked at. In order for a programmer to implement alpha-beta pruning within the recursive part of the program, the programmer will need to keep track of the best move for the computer, alpha, and the best move for the user, beta. The program will also have set to see if it is calculating the best move on a min or max level of the minimax algorithm.

So, when implementing alpha-beta pruning, the program will first look to see if it is at a max or min level in the minimax algorithm. If it is a max level the program will compare alpha and beta. If alpha is greater that or equal to beta then the program knows the possible children in the game tree will not be better than a different node that has its heuristic already calculated. Therefore we can utilize pruning by not doing the computations for the children nodes of that branch since, we know that a better move is not possible due to the nature of how alpha-beta pruning works. Simular, a min level is handeled in the same fashion as a max level but it tests to see if beta is less than or equal to alpha. This technique is a simple test in the programming code as well as keeping track of alpha and beta. However, when certain nodes and their children are not looked at, they will not result in a better move by the computer and the number of computations is reduced. Thus decreasing the amount of runtime it takes for the computer to make a move.

IV. Implementation in C++ Coding

Although the majority of this program's complexity is in the code that makes the program artificially intelligent, the program is still being coded in C++, which required me to make other coding decisions. When coding the program of 3D tic-tac-toe we had to determine how to represent the board, return the correct value to a certain function, how to display the board to the users, and also how to test the program for correctness.

When deciding how to represent each piece on the board, we needed to decide what kind of array we were going to use in order to contain the 64 possible moves. We chose to use a one-dimensional array because of simplicity of programming. Our program required a large amount of searching to test, find, and output data. In doing these operations we used FOR loops. By choosing to represent the board as a one dimensional array the program did not require nested loops. This made it easier to program and find errors in our code.

However, when choosing how to have a user input their next move, we made our program ask the user to input the position they want in a row, column, and level format. This type of entry is easier for a user to determine where a spot is on the board.

When displaying the game we had to consider the idea of how to represent a three dimensional game onto a two dimensional surface. So, in our C++ version of the game we display the board by outputting four 4x4 grids and place them in a vertical line. This output is fairly simple to read for being displayed in two dimensions.

Another problem we had when making this program was returning the correct value of the best possible move back to the main program. Since we used recursion to simulate the tree, we needed to return the value of the best move within the recursive function. However, when the recursive function was done searching through the tree and it was time to return the best move back to main function, we did not want our program to return the best value but to return the position of the best valued move. Thus, we added extra code that tested to see if the program went through the entire tree and was at the top level and if it was we returned the position of the best move.

These problems along with other coding problems were caught by testing code. Testing code was crutial in making our program work properly. This required additional code to be added to the program. This additional code usually output the values of current variables. The values were analized and corrections to the code were made based upon our analysis. Overall, the testing code allowed us to make sure our results were correct and the proper move was chosen.

V. Conclusion

The goals of the program have been obtained by creating a successful game of 3D tic-tac-toe. Also, the game was intelligent by looking at the future of the board and used a good heuristic, it used recursion to reduce memory consumption, and it used alpha-beta pruning and was four ply to maintain efficient response time.

Appendix

See link for the code:
http://ripon.edu/ academics/macs/summation/2010/supp/3DTTTcpp.pdf

Acknowledgment

I would like to thank my fellow group members, Jay Hardacre and Danny Hanson, for their assistance in the design and analysis of this project. I would also like to thank Professor McKenzie Lamb for his guidance during this project. Finally, I would like to thank the rest of the faculty in the Ripon College Mathematics and Computer Science Department for all their help and support.

References

[1] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Upper Saddle River, NJ: Pearson Education, 2003.

[2] Steven Tanimoto. *The Elements of Artificial Intelligence Using Common LISP*. New York, NY: Computer Science Press, 1990. 194-202.

Bradley D. Bogenschutz was born in Appleton, Wisconsin on February 19th, 1988. He graduated from Hortonville High School in Hortonville, Wisconsin in 2006. Brad is currently studying at Ripon College in Ripon, Wisconsin to receive a bachelor degree. His field of study is in computer science.

He is currently employeed by Ripon College and works in the information technology services department. Over the past few summers he has worked for Sudexo as a painter and for Neenah Papers as a mill hand. He has also maintained the Theta Chi-Delta Omega chapter's website while at Ripon College.

Mr. Bogenschutz is a member of the Ripon College Track & Field team and Theta Chi Fraternity. He is also involved in intermeral sports. In his free time Brad enjoys to fish, hunt, snowmobile, and photography.

Summation
May 2010, pp. 9-15
http://ripon.edu/macs/summation

APOS Theory as a Conceptualization for Understanding Mathematical Learning

Sarah R. Weyer
Department of Mathematics and Computer Science – Ripon College

***Abstract*—There are different learning theories in the field of mathematics. APOS theory is one particular theory that can be applied to learning mathematics at the collegiate level. This theory will be described in detail and applied to learning the concept of function, which is a crucial concept in mathematics.**

***Key Words*—action, APOS theory, process, object, schema**

I. Math Learning Theories

THERE are different theories about the learning process. For example, Piaget's theory is a general learning theory that provides a framework for how the learning process takes place. Not only does Piaget's theory describe how the body and brain develop cognitively, but also physically and emotionally. Due to this, Piaget's theory can be thought of as more of a development theory. Piaget's theory provides a general framework for thinking about the stages of development, and it can be applied to any area. Cognitively Guided Instruction, or CGI, however, is a diagnostic step-by-step theory that is applied specifically to learning mathematics. CGI focuses on a lower level of learning mathematics because at the higher levels, learning is not strictly linear. A learning theory that focuses on this higher level of learning mathematics is APOS theory. Dubinsky did a great deal to help define APOS theory. APOS theory is a framework for the process of learning mathematics that pertains specifically to learning more complex mathematical concepts [4]. APOS theory can be thought of as an extension of the last stage of Piaget's theory, the formal operation stage, which takes place at about age 16 [4].

A. *What is a Learning Theory?*

Before looking more carefully at APOS theory, it is necessary to understand what a learning theory is. A learning theory is a theory or prediction about how people learn, including different stages of the learning process they may go through. A learning theory seeks to discover ways to promote deeper, more meaningful learning. The purpose behind learning theories is to improve education and educational experiences in order to help learners move through the steps in the learning process more quickly if possible, which as a result will make the learning curve steeper. Dubinsky and McDonald explained their purpose behind applying a theory of learning to mathematics, stating "we concentrate on how a theory of learning mathematics can help us understand the learning process by providing explanations of phenomena that we can observe in students who are trying to construct their understandings of mathematical concepts…" [4, pg. 1].

B. *Characteristics of a Learning Theory*

There are six essential characteristics of a learning theory according to Dubinksy. First, it supports prediction, which means that if some circumstance happens, then another circumstance will happen as a result. The first circumstance to happen should be something that happens as a result of instructional practice. A learning theory also should possess explanatory power, be applicable to a broad range of experiences, help organize thinking about learning experiences, serve as a tool for organizing data, and provide a language for communication about learning [2].

II. APOS Theory

A. *The Development of APOS theory*

Looking more closely at the development of APOS theory will help provide a better basis for understanding the theory. A common thread between Piaget's learning theory and APOS theory is constructivism. Constructivism is the "theory of learning, introduced by Piaget, in which knowledge is constructed by an individual in her or his mind, as opposed to being intrinsic from birth or existing independently of human interaction" [3, pg.18]. This idea of constructivism stresses the thought that learning is an active process in which students use previous knowledge as well as information from teachers to learn [3].

B. *APOS Theory*

After looking into what learning theories are, their purpose, and how APOS theory developed, it is easier to understand what exactly APOS theory is. APOS theory is a "theory of how learning a mathematical concept might take place" [2, pg. 11]. It is based on the hypothesis that "An individual's mathematical knowledge is her or his tendency to respond to perceived mathematical problem situations and their solutions by reflecting on them in a social context and constructing or reconstructing mathematical actions, processes and objects and organizing these in schemas to use in dealing with situations" [2, pg. 11]. First, the general ideas of APOS theory will be explained, and then the theory will be applied to a more specific example.

Work presented to the college March 23, 2010.
S. R. Weyer hopes to work as a secondary education math teacher (phone: 262-224-6421; e-mail: weyers@ripon.edu).

C. *Stages of APOS theory*

APOS theory entails four primary stages, including an action, process, object, and schema stage, hence the name APOS [4]. An action is “any repeatable physical or mental manipulation that transforms (mental[ly] or physical[ly]) to obtain other objects” [3, pg. 17]. An action conception is “a form of understanding of a concept that involves a mental or physical transformation of mental or physical objects in reaction to stimuli that the subject perceives as relatively external” [3, pg. 17]. In the action stage, the transformation of objects is thought of as external, and the student only knows how to perform an operation from memory or clearly given instructions [4]. A process conception is defined as “a form of understanding of a concept that involves imagining a transformation of mental or physical objects that the subject perceives as relatively internal and totally under her or his control” [3, pg. 19]. In the process stage, students can perform the same action or transformation without external stimuli. Basically, they have internalized the procedure. Students in this stage can also “think of performing a process without actually doing it” and think about reversing the process as well as using it with other processes [4, pg. 3]. An object conception is “a form of understanding of a concept that sees it as something to which actions and processes may be applied” [3, pg. 19]. The student in this stage sees the procedure as a whole and understands that transformations can be performed on it [4]. Encapsulation is the term used to describe “the mental construction of a process (transformed by some action) into a cognitive object that can be seen as a total entity (or coherent totality) and which can be acted upon (mentally) by actions or processes. The only way to mentally construct a mathematical object” [3, pg. 18]. A schema is “a collection of actions, processes, objects and other schemas, together with their relationships, that the individual understands” in connection with some topic of study [3, pg. 20]. In the schema stage, a student is capable of jumping back and forth among the four stages. The ultimate goal of education is for students to reach this stage of the learning process. While it is useful to think of the stages of APOS theory in this order, in reality these constructions are not really made in this linear sequence. Instead, they are made in more of a partially ordered sequence [4].

III. Applying APOS Theory to the Concept of Function

Now APOS theory can be applied to the concept of function. Think about your own definition of function and what it means to you. This information will be used later. When in the action stage, a student will typically understand the most basic ideas behind the concept of function [3]. In this stage, students might not understand that y and f(x) mean the same thing in that they both represent the output [1]. However, they are capable of substituting numbers into a function expressed algebraically, and then calculating to obtain an answer [3]. The student can think about the problem in a step-by-step manner and look at one step at a time [4]. For example, when given the equation $f(x) = (x + 2)^2$ and asked to solve the equation for when x = 1, a student in this stage would go through the following work to get an answer:

$$f(x) = (x+2)^2$$
$$f(1) = (1+2)^2$$
$$f(1) = 3^2$$
$$f(1) = 9$$

The student would be able to understand that 1 is the input, the expression $(x+2)^2$ represents the procedure or transformation, and 9 is the output. Another common way to introduce functions is by talking about machines. In the action stage, a student understands some value is put into the machine and the machine gives out a value.

When a student has a process conception of function, he or she is able to think of the transformation as an entire activity and internalize the procedure that’s going on. The student looks at the word “function” as a verb and sees a function as doing something. For example, when considering equations, a student in this stage can look at an equation and see the procedure as a whole without having to plug in all the specific values [3]. For instance, when looking at the equation $f(x) = x^2$, a student in this stage can draw the graphical representation by plotting only a few of the points. They can also make general arguments about the function. Along with this, a student in the process stage is capable of understanding the idea of increasing functions and that exponential functions and the parabola $y = x^2$ when the input is zero or positive values are examples of increasing functions.

Students are in the object stage when they realize “it’s possible to carry out actions, resulting in some kind of transformation on a function” [3, pg. 19]. It is in this stage that a student would think of the word “function” as a noun and see a function as something that’s being acted on. The following are examples of the object stage in regards to the concept of function. When considering the graph $y = x^2 + 1$, a student in the object stage would see this representation of a function as taking the graph of the function $y = x^2$, as if it were an object, and shifting the whole graph up one unit to obtain $y = x^2 + 1$. When considering the graph $y = (x+2)^2$, a student in this stage would think of this as shifting the object or graph of $y = x^2$ to the left two units. An additional example would be the definition h = f + g where h(x) = f(x) + g(x) for all x in domain. In the object stage, a student will see this function, h, as the sum of two other functions, viewing these functions as objects. Another example would be implicit functions. For example, the equation $x^2 + y^2 = 25$ represents a circle with radius 5. This equation implicitly defines two functional relationships between x and y. In the object stage, a student would be able to look at the graphical representation of this function and see that it is not a function because it does not pass the vertical line test. However, if they were to look at the graph as an object which could be split along the x-axis into two separate graphs, they could see that the two resulting graphs or objects are representative of functions.

In the schema stage of APOS theory, a student has “a collection of actions, processes, objects, and other schemas, together with their relationships, that the individual understands” in connection with functions [3, pg. 20]. The student is able to jump back and forth between the stages of action, process, object, and schema in relation to the concept of function, which were previously discussed. For example, in

Calculus I, students are introduced to the concept of derivatives and they need to be able to take derivatives given various representations of functions, including graphs, equations, tables, etc. Therefore, in order to learn about taking the derivative of functions, students need to have a schema for functions.

IV. Research Study

Specific research studies have been done pertaining to the idea of applying APOS theory to the concept of function. One such study was done by the researchers Dubinsky and Harel, whose main objective was to investigate "how students develop an understanding of the concept of function" [3, pg. 85]. This study was performed on 22 undergraduate students in a Discrete Mathematics course. The students' beginning points of knowledge on functions mainly consisted of basic conceptions and action conceptions, with very few having a process conception. During the course of the study, which took place over a semester, students were instructed with the intention of increasing their understanding of functions to the process conception. The research question of the study was, "How far beyond an action conception and how much into the process conception was each student at the end of the instructional treatment?" [3, pg. 86]. The students were asked to give their definition of function. Then, they were asked to look at different situations and write whether they thought the situations were functions. At the end of the course, the students were interviewed and again asked to give their definition of function and then explain whether they felt various situations were functions and explain if they felt the same as before the course began or whether their answers had changed and why [3]. During the study, the researchers determined the process conception of function is very complex and there were varying stages of progress. Therefore, in order to assess the students, the researchers looked at some particular issues. There were three main restrictions students had concerning functions, including a manipulation restriction, a quantity restriction, and a continuity restriction. A student with a manipulation restriction thinks something must be manipulated in order for something to be a function [3]. A quantity restriction deals with the fact that a student thinks the inputs and outputs must be numbers in order for something to be a function. Lastly, students with a continuity restriction think that in order for something to be a function, it must have a continuous graphical representation. Besides just looking to see if students had any of these restrictions, the researchers also looked at the severity of the restrictions in each case to better assess each student. Other assessments involved students' capability to create a process when one wasn't clear and whether they confused uniqueness to the right with one-to-one [3]. At the end of the course, the students were interviewed on six to eight of 24 situations to determine what stage of APOS theory the students were at in regards to the concept of function . The 24 situations the students were given fall under the following eight categories: ISETL procedures, finite sequences, character strings, graphs, sets of ordered pairs, a table, equations, and statements. Figure 1 displays some of these situations [3].

A. ISETL Procedures

ISETL, or Interactive SET Language, procedures consist of computer code that gives specific, step-by-step procedures within a computer program. These procedures describe how the inputs are transformed to get an output. Students who don't understand how the computer procedure is related to a function have a prefunction response. In an action response the student would be able to think about the procedure in terms of its individual steps. A process response would consist of being able to think about the procedure without having to think about the steps [3]. Dubinsky and Harel further described the process stage in the category of ISETL procedures, stating, "If he or she is able to…discuss the input and its transformation in general terms, to combine the procedure with other procedures and even to reverse the action of the procedure, then one might consider that the student is displaying a process conception" [3, pg. 91].

B. Finite Sequences

Of the 24 situations, finite sequences made up another category. The finite sequences in Figure 1 include S2 and S10. A sequence is a list of elements that can be random, where the order of the elements is important. A sequence by itself is not representative of a function. Therefore, a student must add

Situations Used in the Study

S1) {[x,2x+1]: x in the set of all integers}

S2) [2n+n3: n in [1…100]]

S10) {$2^n > n^2 + 3n$: n in [1…100]}

S17) {[*x*,*y*]: *x*,*y* in the set of all rational numbers}

S19) A swimmer starts from shore and swims to the other side of the lake.

S20) The club members' dues status.

Name	Owed
Sue	$17
John	6
Sam	27
Bill	0
Iris	6
Eve	12
Henry	14
Louis	6
Jane	12

S21) "ELPRHZAUPQDRMW"

Figure 1. These are some of the situations students were presented with during the study done by Dubinsky and Harel. The students were asked to explain whether or not these situations were representative of functions.

something to the condition in order to make it a function. It is hard for students in the action stage to see finite sequences as functions because they don't see the input as the position and the output as the element in the sequence in the given position [3]. However, if a student in the action stage does see this relationship, they may think that the representation of the sequence is not continuous and therefore it is not a function due to the continuity restriction. In a process response, a student can think about each individual term in the sequence in order to create a procedure. In this stage, the relationship between position and the elements in the sequences becomes apparent [3]. They are able to match the elements in the sequence to their corresponding position. For example, students were given the sequence S2 or [$2n+n^3$: n in [1…100]] [3]. Looking at this sequence, students in the process stage would see the relationship between the position as the input and the corresponding element as the output. They could see that by plugging in values 1..100, they would get the sequence 3, 12, 33, 72,… From this, they could determine that the input would be 1, the first position, and this would be paired with the output of 3. Similarly, the input 2, or the second position, would be paired with the output of 12, and so on. Another sequence the students were given was S10 or {$2^n > n^2 + 3n$: n in [1…100]} [3]. When in the process stage students would be able to develop the sequence false, false, false, false, false, true, true,… by plugging values into the given information. For instance, they would see that when n=1, they have 2 > 4, which is false, when n=2, 4 >10 is false, and so on. Then, after developing this sequence, they could see it as a function due to the relationship between input as the position and output as the result of "true" or "false". For example, when the input is 6, or the sixth position, the output is "true".

C. Strings

Along with sequences, strings were another category. The string in Figure 1 is S21. A string is a special case of a sequence. In a string, the order is still important and the input is still the position, but the output must be characters [3]. In this study, the strings students were given were represented as "fait accompli," or "sitting there after having been formed" [3, pg. 92]. For these particular situations, the students had to create "the process of going from the index (domain) to the value (character) as we described for sequences" [3, pg. 92]. One of the examples students encountered was S21 or ELPRHZAUPQDRMW. In the process stage, students would again be able to see the relationship between input as position and output as the corresponding character in that position. For instance, they could determine if 3, or the third position, is the input, then the output would be P, the character in the third position. Students in the process stage can understand this while students in the action stage typically cannot because again students in the action stage need the transformation to be given, would usually not see the input as the position, and might have a quantity restriction, a problem since in strings the output must be characters [3].

D. Graphs

Graphs were also a category [3]. Given a graph, it is difficult to see an action conception since no description of a transformation is provided. Therefore, the graph situations were mostly used to show if a student has a process conception of function. In order to be in the process stage, the student must create "a process of locating a domain value on one axis and then moving in the direction of the other axis until the curve is met. The amount moved is the output of the function" [3, pg. 92]. For example, looking at the graph of the equation $y = x^2$, a student in the process stage would be able to identifying the value of 2 on the x-axis as the domain, move vertically upward until the curve was met, and identify the distance moved, which is 4, as the output.

E. Ordered Pairs

An additional category was sets of ordered pairs. The situations S1 and S17 in Figure 1 fall under this category [3]. A student must be in a late process conception to understand the difference between being a function, being one-to-one, being onto, and having a one-to-one correspondence. Before explaining these concepts, a few terms need to be clarified. The domain is the set of inputs. The codomain is the set of all possible outputs, and the range is the set of actual outputs. There will be graphic representations of different relationships between two sets, and the left set will be the inputs and the right set will be the outputs.

The first relationship to be addressed is a function. A function is a relationship where for every x, there is only one y. A common way to tell if a graph represents a function is by using the vertical line test. If any vertical line only crosses the graph at one point, then it is a function because for every x, there is only one y. Figure 2 represents a function. The ordered pairs created from this representation are (1,5), (2,6), (3,6), and (4,7). Looking at these ordered pairs, one can see that for every x value, there is only one y value.

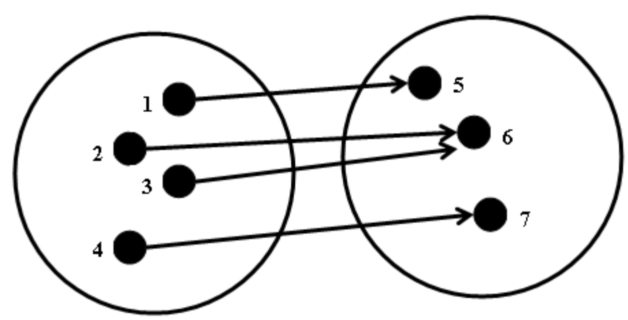

Figure 2. This graphic represents a function.

Figure 3 does not illustrate a function. The ordered pairs created from this representation are (1,4), (2,5), (2,6), and (3,7). Looking at the ordered pairs, it is apparent that for every x, there is not only one y. For instance, for the x value 2, there are two y values, 5 and 6.

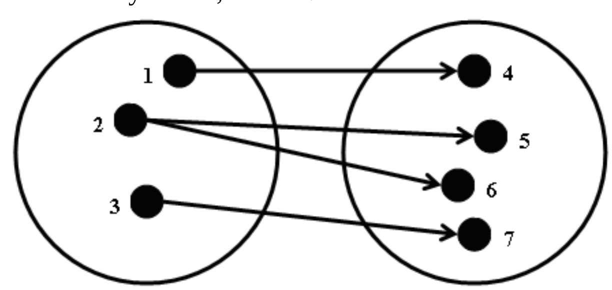

Figure 3. This illustration does not represent a function.

A one-to-one relationship means that for every y, there is only one x. An effective way to tell if a graph is one-to-one is by using the horizontal line test. If any horizontal line only crosses the graph at one point, then it has a one-to-one relationship. Figure 4 is representative of a one-to-one relationship, but not a function however. The ordered pairs in this representation include (1,4), (2,5), (2,6), and (3,7). Looking at these ordered pairs, it is evident that for every y value, there is only one x value.

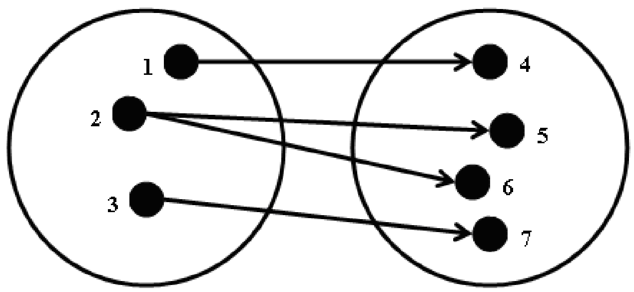

Figure 4. This graphic displays a one-to-one relationship.

Figure 5 does not represent a one-to-one relationship. The ordered pairs created from this representation are (1,5), (2,6), (3,6), and (4,7). Looking at these ordered pairs, it is apparent that for every y value, there is not one and only one x value. For instance, for the y value 6, there are two x values, 2 and 3. Therefore, these sets do not have a one-to-one relationship.

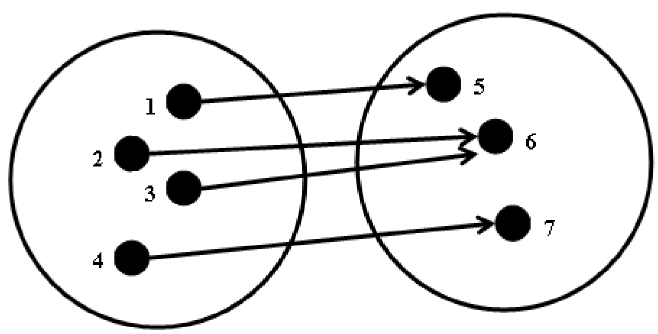

Figure 5. This representation does not depict a one-to-one relationship.

When a relationship is onto, the range is equal to the codomain. Figure 6 depicts a relationship that is onto. The ordered pairs for this representation are (1,5), (2,6), (3,7), and (4,8). For these sets, the range is the y values 5, 6, 7, and 8, and the codomain is the values 5, 6, 7, and 8. Therefore, the range is equal to the codomain, and we have an onto relationship.

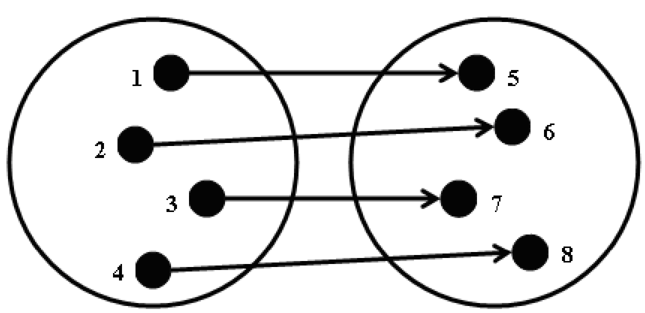

Figure 6. This representation displays a relationship that is onto.

Figure 7 displays a relationship that is not onto. The ordered pairs for these sets are (1,5), (2,6), (3,7), and (4,8). However, looking at the outputs, the codomain includes 5, 6, 7, 8, and 9, while the range is only 5, 6, 7, and 8. Thus, the range is not equal to the codomain, and the two sets do not have a relationship that is onto.

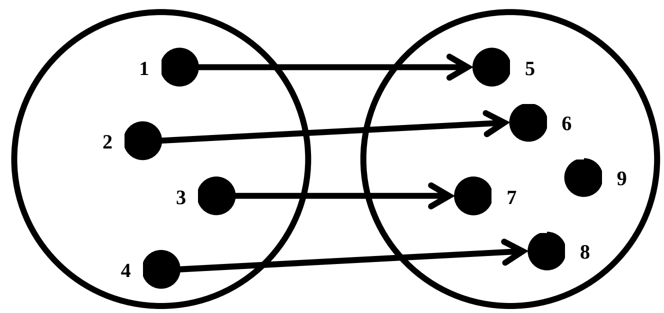

Figure 7. This graphic illustrates a relationship that is not onto.

When there is a one-to-one correspondence, the representation is both one-to-one and onto. Figure 8 displays a one-to-one correspondence. The ordered pairs for the representation are (1,5), (2,6), (3,7), and (4,8). Looking at these sets it is evident that for every y value, there is one and only one x value. Thus, the sets have a one-to-one relationship. Also, looking at the sets, it is evident that the range is equal to the codomain. Thus, the sets have a relationship that is onto. Therefore, the two sets have both a one-to-one relationship and a relationship that is onto. The function is thus a one-to-one correspondence between the sets.

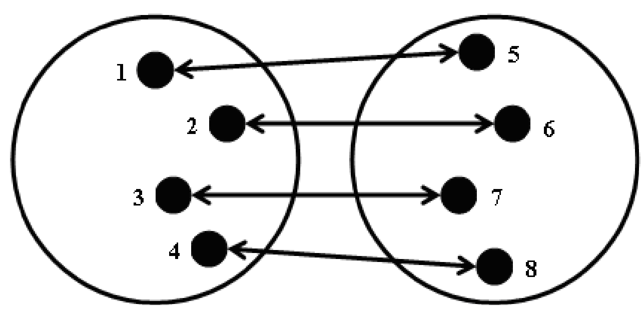

Figure 8. This graphic displays both a one-to-one relationship and a relationship that is onto. Therefore, this function is a one-to-one correspondence between the sets.

When students need to draw these pictures and use arrows to show relationships, they are in the action stage. However, when they can look at the ordered pairs and see the relationships without drawing these pictures out, they are in the process stage.

F. Tables

Among the situations, there was also the category of tables [3]. There is one table, S20, in Figure 1 [3]. An action response to the table would be to try looking for "some rule that relates the first number to the second" [3, pg. 93]. A process response, on the other hand, would be to create a procedure "as the act of going from a quantity in the first column of the table to the corresponding quantity in the second column" [3, pg. 93]. When considering the example S20 the students were given, the following would indicate an action versus a process response. The relationship between the person and the amount they owe would be recognized by a student in the action stage. However, a student in the process stage would not only see the relationship, but they would get away from the common restrictions found in students. In particular, in this case they would get away from the quantity restriction, which is the thought that inputs and outputs must be numbers in order for something to be a function [3]. As a result, they would be able to see the input is in fact the name, which is not a number, and the output is the dollar amount of money that person owes, and the relationship is in fact representative of a function.

G. Equations

Equations were also a category the 24 situations were broken into. Equations and being able to identify whether someone is in the action versus the process stage were previously discussed [3].

H. Statements

Statements were the last category for the situations. The situation S19, in Figure 1, is identified as a statement [3]. The statements the students were provided with were "vague descriptions of physical or geometrical situations and constituted the most open-ended of all the situations" [3, pg. 94]. The purpose of this was to see what function conception the subjects would create when given the least amount of structure [3]. The situation S19 states, "A swimmer starts from shore and swims to the other side of the lake" [3, pg. 90]. Students in the process stage can see a relationship when the relationship is not stated [3]. Here specifically, they may see a relationship with time as the input and distance the swimmer has swum in that time as the output. They may also see other relationships, including time and position, time and how far above the water the swimmer is, and distance and depth.

I. Interviews

Besides this general overview of the difference between an action conception versus a process conception when considering each of the eight categories, 13 interviews were performed with the students about the various situations. Two of these interviews, which were representative of the findings among all the interviews, will be investigated. The two interviews were conducted with girls identified as Ay and Dn [3].

First, Ay's responses during her interview will be analyzed. At the beginning of the interview she is questioned on the term function [3]. The following is a transcription of the conversation with Int representing the interviewer and Ay representing Ay's responses [3, pg. 94-96]:

> Int: "What is a function, what does it mean to you?"
>
> Ay: "Um, a function to me is where you have a set of numbers, your domain, that are numbers that you are going to put into some type of process, and then the process may or may not manipulate the numbers that you are inputting, and then give you some numbers out. And then those numbers that you, I don't know, you just get some numbers out."

When analyzing this response, the fact that she talks about a general set of numbers going in resulting in numbers coming out suggests a process conception. However, she seems to emphasize numbers, which is indicative of a quantity restriction and an action conception [3].

In the following scenario, Ay is talking about why she thinks S17, which is shown below, is not a function [3]:

> S17) {[x,y]: x,y in the set of all rational numbers}
>
> Ay: "…no because you're not really doing anything to the x and y that you're…I mean you're just putting them into a set of tuples. You're not really manipulating the numbers in any sort of way. You're just taking two numbers and just setting them aside in a tuple. You're putting something in and I just, I just don't see it doing anything."

In this response, she contradicts her previous response about what a function is where she says "the process may or may not manipulate the numbers" [3, pg. 94]. Here, she says that S17 is not representative of a function because nothing is being manipulated, which is the manipulation restriction and indicates an action conception of function [3]. However, in the following scenario, "Ay does appear to recognize that the presence of manipulation is not absolutely necessary…[when she] distinguishes [S2] from [S10]" [3, pg. 96]:

> S2) [$2n+n^3$: n in [1…100]]
>
> S10) {$2^n > n^2 + 3n$: n in [1…100]}
>
> Ay: "You have numbers being inputted, and a process being done. And then you're coming out with an answer. In this case being true or false."

In this response, she indicates the manipulation and numbers are not necessary for something to be a function. She indicates that the outputs in S10 are in fact "true" or "false," which is getting away from the manipulation and quantity restriction, and thus represents more of a process conception of function [3].

Overall, some of Ay's responses indicate the action stage while others are indicative of the process stage. Thus, it was determined that Ay is in the transition from the action to the process stage in regards to the concept of function [3].

Dn is first "asked to explain what the term 'function' meant for her" [3, pg. 98]. She gave the following reply [3, pg. 98]:

> Dn: "I look at function as a process where you put something in, go through some sort of process, and you get an output."

The definition of function that Dn provides looks at the procedure as a whole with inputs, a process, and outputs, which shows a process conception [3]. The interviewer and Dn then have a discussion about S1. The following are three excerpts taken from the discussion [3, pg. 99]:

> S1) {[x,$2x$+1]: x in the set of all integers}
>
> Int: "…with your definition of function, if I say, does this, the way it is, represent a function to you, what would your answer now be?"
>
> Dn: "Yes, this is a function."
>
> Int: "Because?"

Dn: "Because it is a set of tuples. Your first…This is a set of tuples. Your first element of your tuple is going to be an integer, and if it's the set of all integers you're not going to have any repeats in there."

Int: "Okay. So go back to your definition for me. So how does that satisfy your thought of what it needs to be?"

Dn: "Okay, when I look at these I think of putting them into the computer. Okay, I'm going to put this set into the computer. Then like if I were going to call up like t(1), okay, I'm going to get out 3, is going to be my answer. I input the first element of the tuple, output comes the second, the second element of the tuple."

Int: "…What about that manipulating, that process you were talking about?"

Dn: "The process is you input the first element of tuple. It looks at its tuples and finds one that had its first element in it, and outputs the second element."

Specifically, in the second excerpt, Dn creates her own process of using computer code to think about the situation. Dn has the capability to construct a process when one is not given, which points to a process conception. Additionally, she recognizes in the first excerpt that no value appearing more than once as the first component satisfies the definition of function. This again signifies a process conception. Thus, the researchers found Dn to have a process conception of function [3].

J. Conclusions of Research Study

Some conclusions can be drawn from the findings of this study. The findings show that even after years in math classes, education is not getting people past the process stage. Math majors are typically the ones who get into the object and schema stage.

V. Weaknesses of APOS Theory

Compared to the ideal learning theory, APOS theory does have some weaknesses. One weakness is APOS theory isn't very predictive. For instance, APOS theory is very different from CGI in that it is not diagnostic and someone can't say "this is where they are, so this is what I need to do to teach them." APOS theory is more of a framework for the way that people learn mathematics. Along with not being very predictive, there are many questions and factors to be considered in order to determine where a student is at in their learning process. Due to this, applying APOS theory can be time-consuming and interpretive [3].

VI. Conclusion

Overall, APOS theory is a partially ordered learning theory on the way people learn mathematical concepts, including the stages of action, process, object, and schema [2]-[3]-[4]. The research done indicates that it is essential for teachers to present information in multiple ways instead of repeating the same information over and over. This will help students to make connections and have a deeper understanding of a particular concept. This should help students move through the stages. For example, think about the definition of function you had at the beginning of this paper and whether or not it may have changed after being exposed to information pertaining to eight different categories of situations regarding functions. Ongoing research is being done in this area, and it must be done by mathematics educators. This is due to the fact that the theory is applied to a higher level of math so the researchers need to have a strong mathematical background.

Acknowledgment

I would like to thank Professor Scott for all his help in this research process and throughout my educational experience in college. I would also like to thank Professor Diane Beres, Professor Hess, Professor Lamb, and Professor Peters for their excellent instruction. Additionally, I would like to thank all the students in Ripon College who have helped me throughout my educational experience here.

References

[1] Breidenbach, D., Dubinsky, E., Hawks, J., & Nichols, D. (1992). Development of the process conception of function. *Educational studies in mathematics journal,* v23, 247-285.

[2] Dubinsky, E. Using a Theory of Learning in College Mathematics Courses. *Newsletter 12 of TaLUM (Teaching and Learning Undergraduate Mathematics),* 10-15. Retrieved from http://ltsn.mathstore.ac.uk/newsletter/may2001/pdf/learning.pdf

[3] Dubinsky, E., & Harel, G. The concept of function: aspects of epistemology and pedagogy. *Purdue University, 85-106.*

[4] Dubinsky, E., & McDonald, M. APOS: A Constructivist Theory of Learning in Undergraduate Mathematics Education Research, 1-10. Retrieved from http://www.math.kent.edu/~edd/ICMIPaper.pdf

Sarah R. Weyer was born in Menomonee Falls, WI on January 10, 1988. She will be graduating with a Bachelor of Arts degree in business administration and mathematics with a minor in secondary education from Ripon College, located in Ripon, WI, 54971 in the United States, in May of 2010.

She has worked as a waitress at Corfu Restaurant in Hubertus, WI, a cashier at Menards in West Bend, WI, an accounting tutor at Ripon College in Ripon, WI, and a waitress at Alma's Café in Allenton, WI. She hopes to work as a secondary education math teacher upon graduation.

Summation
May 2010, pp. 16-20
http://ripon.edu/macs/summation

Jean Piaget's Cognitive Development Theory in Mathematics Education

Kristin E. Reedal
Department of Mathematics and Computer Science – Ripon College

***Abstract*— Jean Piaget's cognitive development theory discusses how an individual progresses through the learning process in stages. This paper will discuss Jean Piaget's developmental stages and apply this theory to the learning of mathematics. More specially, this theory of learning will be applied to the concepts of one-to-one correspondence and comparisons.**

***Key Words*— child development, infinite sets, Jean Piaget, mathematics, cognitive development**

I. Introduction

JEAN Piaget's cognitive development theory has strongly influenced the way we view how individuals learn and the process that people go through while constructing their own knowledge. This is especially applicable in the mathematics discipline. Mathematics concepts build off of one another and intertwine and Jean Piaget's theory considers the steps through which process occurs.

II. Jean Piaget Background

Jean Piaget was born in 1896 in Switzerland and addresses how we learn and therefore how we learn mathematics [2]. Piaget focuses his theory on the idea of constructivism, which is that learning is constructed from each individual's experiences and connections between previously learned concepts and new ideas [3]. This leads into his idea of disequilibrium, which drives the learning process. Piaget's theory of disequilibrium describes when new ideas and concepts do not fit with what we already know and therefore we are forced to adjust our thinking to incorporate this new information [2]. Finding how concepts are connected and fit together allows us to reach a point of equilibrium. "…children acquire new systems of cognitive operations (structures) that radically alter the form of learning of which they are capable" [4]. Therefore he believes that before any new concepts are presented, a student's capability of understanding should be assessed [4]. Although on the average children will pass through stages at generally predictable ages, according to C. Kamii, Piaget believed that the amount of time each child spends in each stage of development will vary based on the individual [9]. Therefore it is even more important to assess where a child is before presenting new information because not all children will be at the same stage of development at same ages. So while each child will progress through the stages of development that Jean Piaget identifies, each child should be an active participant in this learning process.

Work presented to the college March 23, 2010 as the second talk from a series of three about Cognitive Development and the Learning of Mathematics group.

Kristin E. Reedal is currently pursuing her Bachelors degree in mathematics with an education minor from Ripon College in Ripon, Wisconsin.

III. Cognitive development Theory

Jean Piaget identifies four stages of cognitive development that all children will progress through at some point in their lives.

A. Sensorimotor Stage

The first stage of development that Piaget identifies is the Sensorimotor Stage. This is generally between birth and two years old, although children will progress through this stage at their own pace [9]. At this point, children are learning using their five senses and need concrete experiences to understand concepts and ideas [2]. Also at this stage, children are limited in their world and what they are able to comprehend. Children are egocentric and can only see the world from their own perspective [2].

If we specifically address how the Sensorimotor Stage applies to mathematics, we can see how learning is developing. First, children begin to have an understanding of object permanence in which they are able to find objects that have been taken out of their view [9]. Piaget demonstrates this through an experiments in which he displaces an object under a pillow, and children are able to correctly find the object, illustrating they know the object still exists even though they cannot see it [9]. Children are also beginning to see how numbers link to objects and therefore approach being capable of counting on their fingers [9]. They understand that one finger matches with the number one and can count concrete objects using this concept. At this point in development, children don't understand point of view but can distinguish themselves from the rest of the world [9]. For example, they recognize that a difference exists between themselves and their parents.

If we try applying this stage to the concept of one-to-one correspondence, it becomes clear that this idea is starting to develop even at this early stage of learning. One-to-one correspondence is the ability to match numbers to objects and objects to objects. Children begin to match one object to one person or one toy to one person. Individuals struggling in this stage of understanding this concept will be able to recite the first ten numbers but might not be able to match the number to objects.

B. *Preoperational Stage*

The second stage of cognitive development identified by Jean Piaget is the Preoperations Stage, during two to seven years old [2]. "During this period, children are able to do one-step logic problems, develop language, continue to be egocentric, and complete operations" [2]. These children's development continues, and this stage marks the beginning of solving more mathematically based problems like addition and subtraction.

In mathematics, children at this stage are able to solve one-step logical problems but still are primarily limited by working with concrete materials. The children need to incorporate any materials that are available such as blocks, and counters [9]. Because they are still egocentric as in the previous stage, children are "…restricted to one aspect or dimension of an object…" [9]. Therefore they are also limited in their rational and logical thinking, and thus limited in their mathematical abilities. Also, since children are restricted to one dimensional thinking, they will be primarily influenced by the visual representation of things. More specially, if two things look different, the child is likely to conclude they are different while really the visual representation could simply be a different perspective. Since they are now able to start completing basic operations, children may be able to see how a sequence changes. For example, if a child was presented with the sequence 1, 2, 3, 4, 5 and asked to compare it to 2, 4, 6, 8, 10, the child would recognize there is a difference but might not articulate how the first sequence transforms into the second sequence.

Some development becomes obvious when we consider how a child is capable of understanding the idea of one-to-one correspondence. At this point, since children are beginning to solve one-step logical problems; they can apply the idea of a one-to-one correspondence when working through the process of solving problems using manipulatives and can also apply this concept to addition and subtraction problems. A child in the preoperational stage is still limited to the concrete world, using manipulatives to represent problems can be a very useful problem solving tool and without having an understanding of a one-to-one correspondence, this strategy would not be very beneficial because the problems would not be represented accurately.

C. *Concrete Operations Stage*

The next stage of development that Piaget discusses is the Concrete Operations Stage which generally recognizes a child between the ages of seven to eleven years old [2]. A child would be able to think logically and start classifying based on several features and characteristics rather than solely focusing on the visual representation [9].

Mathematically speaking, this stage represents a remarkable new development for a child. Since children can now classify based on several features, they are able to consider 2 or 3 dimensions [2]. While children were previously limited to their own point of view, they can now take into account others perspectives. They can also begin to understand the ideas of seriation and classification more thoroughly while also developing how to present solutions in multiple ways [2]. In order to develop the ability in a child of presenting multiple solutions, discussions in a classroom can be very helpful. This experience of sharing solutions could open the child up to the idea that there is not always one right way to solve every problem; there could be multiple ways that are equally correct solutions. The child in this developmental stage will also have the skills to make basic observations and routine measurements which increases the ways they classify objects/situations [2].

The concept of one-to-one correspondence would further develop this stage and a child would construct a firmer understanding of functions. They will not completely understand functions yet but they may have a loose understanding of what defines a function that has a one-to-one correspondence. A child may recognize that a one-to-one corresponding function will have one unique input that matches up with one unique output, while having the same number of inputs as outputs. So for example they might understand x^2 as a function is not a one-to-one correspondence.

D. *Formal Operations Stage*

The last stage of development that Piaget identifies is the Formal Operations Stage, which children enter roughly between the ages of eleven to sixteen years old and continues throughout adulthood [10]. This marks the distinct change of a child's thinking to a more logical, abstract thinking process [2].

At this point in cognitive development, children do not need the concrete experiences they required to understand mathematics in the previous stages. They form their own hypotheses and determine possible consequences which stems from seeing situations from differing perspectives [9]. The child will also begin to understand abstract concepts which lead to much more complicated mathematical thinking [9]. Since they understand more abstract concepts, mathematically speaking, they may be capable of estimating the area under a curve, which is not based on a concrete experience. They also begin to think about the concept of infinity and understand how to estimate what an infinite series converges to.

If we look back to the idea of one-to-one correspondence, you can further see how mathematical concepts are developing. A child begins to understand how to determine if sets are the same size, through applying the idea of a one-to-one correspondence. For example, they can compare negative and positive integers using a one-to-one correspondence. If you apply this idea, it becomes clear -1 matches with 1, -2 matches with 2, and so on, which allows us to see that the sets of negative and positive integers have a one-to-one correspondence and therefore have the same number of elements.

IV. Application to Comparisons

To further examine how a concept grows through these stages of development, let's look at the idea of comparisons.

A. Sensorimotor Stage

At the Sensorimotor Stage of Piaget's cognitive development, children will be limited in how they compare objects or situations. At this point, children will be able to compare themselves to their parents or compare sizes of toys. Since they are unable to see different points of view, they will be limited to how situations or objects appear to them and will be restricted to concrete experiences and objects.

B. Preoperational Stage

The second stage of development allows a little further understanding of how to compare more objects or situations but is still restrained by the concrete world. Children in the Preoperational Stage will be able to compare containers that are the same height and volume correctly but will be unable to compare different containers. For example, one of Piaget's studies tested for conservation and had a child compare two containers that were the same height and width [9]. Then, the contents of one of the containers was poured in a thinner container, which changed the height of the liquid [9]. At this point, the child was unable to accurately compare the two different containers and concluded the container that had a higher level of liquid must have more liquid than the other container while really they had equal amounts [9]. The child uses the idea of height to compare the two different containers with liquid but doesn't understand the effect of width on height

C. Concrete Operations Stage

At the Concrete Operations Stage, the child can accurately compare the two different containers [9]. The child understands that pouring the liquid into a different container does not change the amount of liquid [9]. Also, the child understands that containers that have the same amount of liquid but different dimensions will appear to have differing amounts of liquid because they have different levels of liquid. The child takes comparing these situations to a higher level and accurately compares these two containers of liquid.

D. Formal Operations Stage

In the Formal Operations Stage, the child further understands how to compare different mathematical situations. At this point in children's development, since they are not limited to the concrete world, they can compare fractions, the possibilities of different events, and infinite sets. While infinity can cause an extended period of disequilibrium, the concept is beginning to be further understood. Having students examine infinite sets provides a situation in which we force the child to experience disequilibrium, in order for them to generate a deeper understanding of infinity. When comparing infinite sets, we use the idea of one-to-one correspondence to determine if two sets have the same number of elements. For example, when we compare two line segments that have different lengths, we can apply the idea of one-to-one correspondence to conclude they have the same number of points.

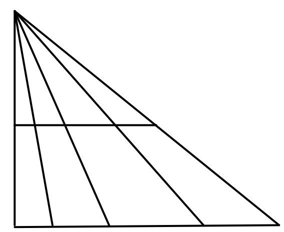

Figure 1. Two line segments that have different lengths both have the same number of infinite points, because there is an one-to-one correspondence between the points on the lines.

Figure 1 shows we can match up every point on one line to a unique point on the second line. Further, each point on the second segment corresponds to a point on the first segment. So even though one line may be contained in the second line, since there are an infinite number of points on one line, there are equally many points on the second line segment.

Another example arises if we compare {1, 2, 3…} and {1, ½, 1/3…}. We can use one-to-one correspondence to see that these sets do in fact have the same number of elements. The unbounded set {1, 2, 3…} clearly has an infinite number of elements but it may be less clear when examining the second set, which is bounded, since many people's intuition of infinity includes the idea of unbounded. But if we use a one-to-one relationship, we can match 1 and 1, 2 and ½ and so on. Thus the sets are the same size.

If we look at the idea of comparing infinite sets a little bit further, we arrive at the question about whether two infinite sets can possibly have different sizes. To examine this question, we compare the real numbers between 0 and 1 to the natural numbers, leading to the following proof.

Proof

Why is the set $(0, 1) = \{x: 0<x<1\}$ not the same size as {1, 2, 3…}?

If they were the same size we would be able to have a 1-1 correspondence between the two sets. Something would be matched with 1 - lets call it x_1, something with 2 - call it x_2, something with 3 – call it x_3, and so on. Every number would be matched with a different integer, so the interval of real numbers between 0 and 1 would be listed $x_1, x_2, x_3…$

Claim: The list has a missing number.

Think of writing each number in expanded decimal form. We can then think of them going on forever, ½ = 0.50000…. and 1/3= 0.33333….. We note that some care is necessary though since 0.499… = 0.500….

So, let

$x_1 = 0.x_{11}x_{12}x_{13}x_{14}\ldots$

$x_2 = 0.x_{21}x_{22}x_{23}x_{24}\ldots$

$x_3 = 0.x_{31}x_{32}x_{33}x_{34}\ldots$

$x_4 = 0.x_{41}x_{42}x_{43}x_{44}\ldots$

The first subscript indicates which number in the list to refer to and the second indicates which decimal place.

Now think about the number $y = 0.y_1y_2y_3y_4\ldots$where $y_1 \neq x_{11}$, $y_2 \neq x_{22}$, $y_3 \neq x_{33}$ … and so in general $y_n \neq x_{nn}$ for n = 1, 2, 3...

Then y is between 0 and 1 but is not in the list because it differs from each number in the list at some point in its expansion. So no list contains all the numbers from 0 to 1. Therefore there is not a 1-1 correspondence and there must be more numbers from 0 to 1 than in the set of natural numbers.

Thus, we are able to conclude that not all infinite sets are the same size.

This makes comparing infinite sets more interesting because when we conclude two sets have the same number of elements; it is not because they are both infinite but because they truly have the same number of elements. This example shows why there can be an extended period of disequilibrium even through the basic idea is an extension of an idea that children start to understand at early ages.

V. Weaknesses

While many would feel that Jean Piaget's cognitive development theory has had many positive effects, others would point at the weaknesses in his theory. Some critics feel that his contribution to cognitive development relies too heavily on observations without scientific support [6]. Criticism of Jean Piaget's developmental theory also focuses on his failure to offer a complete description of a child's stages and why [7]. Also, since Piaget labels each stage of development with a general age group, educators and other individuals can make assumptions about what a child can understand. While this is not a weakness of Piaget's theory, these age groups do create a situation where individuals misapply the stages and make assumptions from his theory. For example, a study was conducted on a group of college freshman where it was expected that these students were into the Formal Operations Stage, but this was not always true [8].

A. *Philippine Study*

The study on college freshman was conducted on 59 Philippine students investigating their mathematical achievement [8]. The college freshman were asked questions about arranging numbers from lowest to highest, finding the next number in a sequence, and determining a missing angle in a triangle [8]. Scores were determined based on completeness and understanding [8]. The results determined that 61.02% of students were in the Concrete Operations Stage and 38.98% of students were in the Formal Operations Stages of development [8]. Therefore while it was assumed that these students would be well into the last stage of development, the assumption was clearly misguided.

This leads to the importance of not making assumptions about what a child is capable of understanding without assessing their cognitive ability first. Thus, while Piaget tries to give a generalization about age groups and the developmental stage a child would be in, people misapply his theory and make incorrect assumptions.

VI. Conclusion

Overall Jean Piaget's Developmental theory allows us to see how a concept develops and outlines how a child progresses through this developmental process of learning. We can also see how topics are related and build off of each other, especially one-to-one correspondence and comparisons. Examining Piaget's theory begs the question of whether we are pushing students in our educational system through this developmental process too quickly and asking them to learn concepts before they are capable of fully understanding these ideas. This leads to the importance of determining where a child is before presenting new information to fully address whether a child is capable of understanding the new material based on their current developmental stage. So, understanding how a child moves through this developmental process can enhance our understanding of how children learn and therefore increase the chances of understanding new complex ideas.

Acknowledgment

I would like to thank my group members, Tyler Ruppert and Sarah Weyer for their support and suggestions enhancing this project. I would also like to thank my advisor, Professor David Scott for his assistance on this project and help developing the proof within this paper.

References

[1] Atherton, James. "Piaget." 04 Nov 2009. Web. 24 Jan 2010. <http://www.learningandteaching.info/learning/piaget.htm>.

[2] Blake, Barbara, and Tambra Pope. "Developmental Psychology: Incorporating Piaget's and Vygotsky's Theories in Classrooms." May 2008. Journal of Cross-Disciplinary Perspectives in Education, Web. 12 Jan 2010. <http://74.125.155.132/scholar?q=cache:GG_o4VWwZOMJ:scholar.google.com/+Developmental+Psychology:++Incorporating+Piaget%27s+and+Vygotsky%27s+Theories+in+Classrooms&hl=en&as_sdt=2000>.

[3] Bodner, George. "Constructivism: A Theory of Knowledge." *Journal of Chemical Education* 63. 1986. 873-78.

[4] Case, Robbie. "Theories of Learning and Theories of Development." *Educational Psychologist* 28.3 1993. 219-33.

[5] Davis, G.A., and J.D. Keller. "Mathematics Development." *Education.com*. Exploring Science and Mathematics in a Child's World, Web. 12 Jan 2010.

<http://www.education.com/reference/article/mathematics-development/>.
[6] Elkind, David. *Children and Adolescents: Interpretive Essays on Jean Piaget*. New York: Oxford University Press, 1970. Print.
[7] Flavell, John H. *The Developmental Psychology of Jean Piaget*. Princeton, NJ: D. Van Nostrand Company, Inc., 1963. Print
[8] Leongson, Jaime A., and Auxencia A. Limjap. "Assessing the Mathematical Achievement of College Freshman Using Piaget's Logical Operations." N.p., n.d. Web. 17 Mar 2010. <http://www.cimt.plymouth.ac.uk/journal/limjap.pdf>.
[9] Ojose, Bobby. "Applying Piaget's Theory of Cognitive Development to Mathematics Instruction." *Mathematics Educator* 18.1 2008. 26-30.
[10] Richmond, P.G. *An Introduction to Piaget*. New York: Basic Books, Inc., Publishers, 1970. Print.
[11] Sousa, David A. *How the Brain Learns Mathematics*. Thousand Oaks, CA: Corwin Press, 2008. Print.

Kristin E. Reedal was born in Park Ridge, Illinois on July 31, 1988. She graduated from William Fremd High School in Palatine, Illinois and is currently pursuing a Bachelors degree in mathematics with an education minor from Ripon College in Ripon, Wisconsin.

She is currently employed by the Mathematics and Computer Science Department and the Educational Studies Department at Ripon College as a Departmental Assistant. She has worked the past few summers as an employee of Aerex Pest Control in Rolling Meadows, Illinois. She will be graduating January 2010 after completing student teaching in Chicago, Illinois.

Ms. Reedal is a member of the Laurel Society of Ripon College and Pi Lambda Theta. In her free time, she enjoys reading, spending time with family and friends, and being outdoors.

Summation
May 2010, pp. 21-27
http://ripon.edu/macs/summation

Cognitively Guided Instruction: An Alternative Way to Teach Mathematics

Tyler J. Ruppert
Department of Mathematics and Computer Science – Ripon College

***Abstract*— Cognitively Guided Instruction is a style of teaching based on years of research showing that people learn beginning mathematical concepts linearly. That is, there are clear stages that are passed through in a particular order. It leads to student centered learning and stimulates discussion about multiple approaches to solving the same problem.**

***Key Words*— Cognitively Guided Instruction, Counting Strategies, Direct Modeling, Inclusive Classroom, and Invented Algorithms.**

I. Introduction

"School curricula […] have too often consisted of discrete facts and skills removed from any meaningful context or sense of purpose. Students spend inordinate hours memorizing and practicing letter-sound correspondences, rules for subject-verb agreement, or computational algorithms, with few opportunities to read, write, or solve mathematical problems as complete and meaningful tasks" [2]. This thought leads directly into how many people begin to learn mathematics at an early age. Most mathematics today continues to be taught with lecture, followed by a textbook homework assignment. This contributes to students' inabilities to understand what math concepts are really occurring during real world situations and/or doesn't lead to good problem solving skills in the latter grades. With this system, students are only trying to get the correct final answer, and often must do it in the way that the teacher has instructed, or it is considered incorrect. Ultimately, for most students, this may lead to poor problem solving skills in the future, and may turn many students away from enjoying math.

This is simply unfair for students who like to look at math from different perspectives. This is why an alternative method to teaching mathematics, which would begin in kindergarten and end with Algebra I, might be considered. This alternative method is known as Cognitively Guided Instruction. Cognitively Guided Instruction (CGI) was developed in 1985 by Elizabeth Fennema, Thomas Carpenter, and Penelope Peterson [5]. CGI is a style of teaching based on years of research showing that people learn beginning mathematical concepts linearly. That is, there are clear stages that are passed through in a particular order. This can be seen in the education system today when starting with concrete math facts in the early grades and then moving to more abstract math ideas in geometry or trigonometry towards the end of high school. With CGI, "Instruction [builds] on children's existing knowledge and […] teachers should help students to construct mathematical knowledge rather than passively absorb it" [2]. As stated, CGI is an alternative way to teach and learn mathematics from an early age where students start with concrete demonstrations of what story problems are asking and eventually work towards abstract representations by inventing of their own algorithms to solve story problems. This is not a flashcard type of learning, which is popular today, especially when considering the early ages. Instead of memorizing number facts, students construct their knowledge in any way possible because all methods of finding solutions are accepted and then critiqued until the desired final answers are correct. Essentially, "children are not shown how to solve problems. Instead, each child solves them in any way that he or she can, and then shows how the problem was solved with peers and the teacher" [8].

II. Organizing a CGI classroom

Imagine an inclusive classroom in which all genders, ethnicities, minor language differences, and disabilities can learn math together. Students first explore individually by being presented a word problem to solve. They use any available materials in the classroom to solve the problem such as: counting blocks, tally marks, paper, markers, etc. The teacher walks around and asks students questions to understand what level the students are at cognitively while solving the problem. Then, the students share their solutions with the entire class and have to communicate their thought processes. After one student presents, another student presents a method. During this time, all peers must listen to the presentations and politely critique the student presenters if there is an error or something unclear in the presentations. The teacher helps facilitate discussion between students, distinguishing the similarities and differences in the presented methods. Students may be at varying cognitive levels, so each student may not present a solution daily, but are encouraged to share whenever they have valid solutions. Also, the class

Work presented to the college March 11, 2010 as part of a three part series in Cognitive Development and The Learning Of Mathematics.

T. Ruppert hopes to work as a secondary mathematics teacher starting in 2011. He will be student teaching in the fall of 2010. (phone: 715-853-4996; e-mail: ruppertt@ripon.edu).

doesn't move on until all the students are ready. This may mean that students are at different levels such as Direct Modeling (concrete stage) or Mental Counting (closer to final abstract stage), but still ready to move on [5].

A. *Identifying Strategies*

In order for teachers to distinguish what level each student is currently working at, problems are classified into different categories. The students start out with a concrete exploration of problems and eventually progress to the stage where they understand abstract ideas such as addition, subtraction, multiplication, and division. The concrete exploration stage in CGI is called Direct Modeling. Direct Modeling is when the student literally does what the problem is asking with manipulating materials (see Table 1, Direct Modeling Strategies). For example, in Table 1, with a *Join (Result Unknown)* problem, students will use a Joining All strategy where they create two sets of objects, such as counting blocks to represent the objects in the problem and then complete the problem by counting all the elements of the two sets to arrive at their final solution. To most people, this would seem like a simple addition problem. When students are at the Direct Modeling stage of learning they are usually in kindergarten, hence the abstract concept of addition is not fully developed yet. The students simply understand they are counting two sets in order to arrive at the total amount.

The next stage of learning is developing counting strategies and moving away from requiring concrete materials such as counting blocks to solve a problem. Examples of these counting strategies can be seen in Table 2. From Table 2, examining a *Join (Result Unknown)* problem again, it can be seen that the student no longer needs counting blocks, but instead counts in their mind or out loud without the use of manipulative materials (See Table 2, Counting Strategies for +/- Problems). Here, students use what is called a Counting On From First or Counting On From Larger strategy. The Counting On From First strategy is done by a counting sequence beginning at the number of the first object and continues counting for the number of the second object to arrive at the final number in the counting sequence as the total. This total is the students' answer. At this stage, the students are starting to explore the abstract. Specifically, in this example the concept of addition is abstract.

Finally, the ultimate goal is to have the students invent their own algorithms from derived number facts to complete problems. Using the same *Join (Result Unknown)* problem, the students could possibly say "I know that 3 and 3 is 6, but there are 2 more, so the answer is 8." More examples of invented algorithms can be seen in Table 3, Invented Algorithms. When students begin inventing their own algorithms, they usually understand base 10 concepts and that is why the examples in Table 3 include base 10 concepts. Once the student is able to invent their own algorithms, they understand the abstract concepts, in this case addition.

TABLE 1
DIRECT MODELING STRATEGIES

Problem	Strategy Description
Join (Result Unknown) Ellen had 3 tomatoes. She picked 5 more tomatoes. How many tomatoes does Ellen have now?	Joining All A set of 3 objects and a set of 5 objects are constructed. The sets are joined and the union of the two sets is counted.
Join (Change Unknown) Chuck had 3 peanuts. Clara gave him some more peanuts. Now Chuck has 8 peanuts. How many peanuts did Clara give him?	Joining To A set of 3 objects is constructed. Objects are added to this set until there is a total of 8 objects. The answer is found by counting the number of objects added.
Separate (Result Unknown) There were 8 seals playing. 3 seals swam away. How many seals were still playing?	Separating From A set of 8 objects is constructed. 3 objects are removed. The answer is the number of remaining objects.
Separate (Change Unknown) There were 8 people on the bus. Some people got off. Now there are 3 people on the bus. How many people got off the bus?	Separating To A set of 8 objects is counted out. Objects are removed from it until the number of objects remaining is equal to 3. The answer is the number of objects removed.
Compare (Difference Unknown) Megan has 3 stickers. Randy has 8 stickers. How many more stickers does Randy have than Megan?	Matching A set of 3 objects and a set of 8 objects are matched 1-to-1 until one set us used up. The answer is the number of unmatched objects remaining in the larger set.
Join (Start Unknown) Deborah had some books. She went to the library and got 3 more books. Now she has 8 books altogether. How many books did she have to start with?	Trial and Error A set of objects is constructed. A set of 3 objects is added to the set, and the resulting set is counted. If the final count is 8, then the number of objects in the initial set is the answer. If it is not 8, a different initial set is tried.

Direct Modeling Strategies. Page 19 [5].

Note: Tables 1, 2, and 3 do not provide all the types of problems. The appendix has more classifications of problems and more examples of other types of problems.

By selecting problems that elicit specific problem solving skills, the teacher is able to see which strategies students are using and what level of understanding students have. The interesting idea of CGI is that some students may be at the Direct Modeling level, some students may be using Counting Strategies, and other students may be using Invented Algorithms, yet all are incorporated within one classroom at the same time. With CGI, all strategies are encouraged and there is no wrong way to solve a problem correctly. The strategy used simply allows the teacher to identify what level each student is currently at and what the individual student is mentally capable of solving currently.

B. Selecting Problems

That is how different students' abilities are classified, but how can this teaching method be implemented into the classroom? How do teachers organize CGI classrooms? The first step is to select problems that match the daily objectives. If the students are going to work in small groups, then select more challenging problems. If the students are going to work individually, then select more straight-forward problems. Yet, sometimes the more advanced students will finish these straight-forward problems quickly, so then encourage them to use multiple strategies to solve these problems. On the other hand, there are the lower level students that may not be able to do the problem yet, so the teacher can reword the problem to make it clearer. For example:

1. Sunny had 7 pennies. His dad gave him some more, and now he has 11 pennies. How many did his dad give him?
2. Ashley has 9 toy rockets. How many more rockets does she need so that she will have 17 toy rockets all together?

Both problems are classified as *Join (Change Unknown)* problems. Yet, the second example is worded to elicit the Direct Modeling strategy. The first example is worded to elicit Counting Strategies or Invented Algorithms. This is where the role of the teacher becomes very important. The teacher must choose problems that are challenging, yet have a focus. Some ideas to focus on are new problem solving strategies, refining reporting skills, or just presenting interesting problems that relate to things in school like a field trip or another subject.

C. Reporting Strategies

The second important aspect of organizing a CGI classroom is reporting strategies. Since it is common only to complete two to four problems in a 50 minute class period, there are not many opportunities for every student to report to the entire class what their solutions were. Therefore, another responsibility of teachers is to question the individual students throughout the process of solving a problem or after they have completed a solution to a problem. These questions could simply be: "How did you solve that problem?" "Tell me how you got the answer." or "Can you tell me what you were thinking?" Once the student responds to these questions, the teacher can ask probing questions to really determine what strategies were used and what level the students are at in understanding the concepts. Some examples of these probing questions are: "Can you tell me why you separated those blocks?" "Why did you start with that number when you counted?" or "Can you tell me how you counted?" The students' answer to these probing questions should be a detailed description of the procedure they used to solve the problem.

D. Manipulative Materials and Problem Solving Tools

Teacher questioning and the student reporting strategies leads to the third important aspect of organizing a CGI classroom: manipulative materials and other problem solving tools. It is very important that at any stage of a students' learning in a CGI setting they feel comfortable to use necessary materials to solve problems. This includes having materials such as counters, paper, pencils, base-ten blocks and/or linking cubes always available for students to use. These materials are not used to demonstrate strategies, but are there for the students to use as tools to solve problems to aid the students in reporting the procedure they used. Once the students gain confidence in what they are doing, they start using the counters less and begin to write number sentences or use tally marks to keep track of numbers counted.

TABLE 2
COUNTING STRATEGIES FOR +/- PROBLEMS

Problem	Strategy Description
Join (Result Unknown) Ellen had 3 tomatoes. She picked 5 more tomatoes. How many tomatoes does Ellen have now?	Counting On From First The counting sequence begins with 3 and continues on 5 more counts. The answer is the last number in the counting sequence.
Join (Result Unknown) Ellen had 3 tomatoes. She picked 5 more tomatoes. How many tomatoes does Ellen have now?	Counting On From Larger The counting sequence begins with 5 and continues on 3 more counts. The answer is the last number in the counting sequence.
Join (Change Unknown) Chuck had 3 peanuts. Clara gave him some more peanuts. Now Chuck has 8 peanuts. How many peanuts did Clara give him?	Counting On To A forward counting sequence starts from 3 and continues until 8 is reached. The answer is the number of counting words in the sequence.
Separate (Result Unknown) There were 8 seals playing. 3 seals swam away. How many seals were still playing?	Counting Down A backward counting sequence is initiated from 8. The sequence continues for 3 more counts. The last number in the counting sequence is the answer.
Separate (Change Unknown) There were 8 people on the bus. Some people got off. Now there are 3 people on the bus. How many people got off the bus?	Counting Down To A backward counting sequence starts from 8 and continues until 3 is reached. The answer is the number of words in the counting sequence.

Counting Strategies for +/- Problems. Page 23 [5].

E. Learning About The Students and Identifying Change

Through seeing students use manipulative materials and growing more and more comfortable with abstract concepts, the teacher is learning about the students and can see changes in the students' abilities. The fourth essential aspect of organizing a CGI classroom is learning about the students and identifying their development as they pass through learning stages. To begin learning about students, teachers must pre-test or interview the students at the beginning of the year. This will give the teacher a baseline for what level each student is currently working at and what levels are left to reach. This interviewing process is an ongoing affair. The teacher must constantly be testing, interviewing, observing, listening, and questioning the students to fully understand what level the student is currently working at. Then, the teacher must be able to realize the levels of learning are constantly changing. Therefore, some days a student may take a step back in the strategies they are using, while another day they are at the highest level. Hence, it is important for the teacher to know what types of problems they are giving so they know what strategies to look for in order to identify the different levels of learning. The tables in the appendix are very useful tools for identifying problem types.

Once the teacher understands what level each student is at, they have to be able to identify the changes in levels. Many teachers will find students are able to solve more difficult problems than they anticipated. Also, the students may have an array of different strategies or a combination of strategies they used to solve a problem. Thus, there are many ways of getting the same solution and in CGI all strategies are accepted. Further, the large group presentations of solutions approach allows students to learn from each other, and the teacher can really see when some students begin to use strategies they have never seen before, after listening to peer's techniques.

III. Weaknesses

Like any alternative learning strategy, there is still research being conducted on the effectiveness of CGI versus the traditional way mathematics is being taught now. However, there appear to be few weaknesses to CGI due to "the NCTM (National Council for Teaching Mathematics) standards being grounded on the process standards of CGI: problem solving, reasoning, connections, modeling, and communication" [6]. Yet, there was one study that looked at the Iowa Tests of Basic Skills to see if CGI students achieved higher test scores than regular math students. This study was conducted by the Promising Practices Network and they concluded that CGI students scored higher on complex addition and subtraction problems, but lower level addition and subtraction problems showed no significant difference between the scores of CGI students and other math students [7]. This is the only study found that analyzed test scores to see if CGI is really beneficial and it seems that at higher levels CGI is more beneficial than traditional mathematics teaching strategies [7]. Further, the same study expressed the concern that to implement the CGI program effectively into a school requires a lot of supervision and training. As of 2007, there has been no development for a systematic evaluation of the effectiveness of this program in terms of the children's' mathematics performance [7]. This leads to principals and superintendents not wanting to spend time and money to have a teacher or multiple teachers learn how to implement CGI into their classrooms. One final weakness that has been suggested, in discussion with other future educators and

TABLE 3
Invented Algorithms

Problem	Incrementing	Combining Tens and Ones	Compensating
Join (Result Unknown) Paul had 28 strawberries in his basket. He picked 35 more strawberries. How many strawberries did he have then?	"20 and 30 is 50, and 8 more is 58. 2 more is 60, and 3 more than that is 63."	"20 and 30 is 50. 8 plus 5 is like 8 plus 2 and 3 more, so it's 13. 50 and 13 is 63."	"30 and 35 would be 65. But it's 28, so it's 2 less. It's 63."
Separate (Result Unknown) Paul had 75 strawberries in his basket. He ate 26. How many did he have left?	"70 take away 20 is 50, and take away 6 more is 44. But you have to put back the 5 from the 75. That's 49."	"70 take away 20 is 50. 5 take away 6, that makes 1 more to take away from the 50. That's 49."	"If it was 75 take away 25, it would be 50. But it's 26, so you have to take one more away. 49."
Join (Change Unknown) Paul has 47 strawberries in his basket. How many more strawberries does he have to pick to have 75 all together?	"47 and 3 is 50 and 20 more is 70. So that's 23, but I need 5 more, so it's 28." "47, 57, 67. That's 20. 67 and 3 is 70, and 5 more is 75. So 8 and the 20, 28."	Combining Tens and Ones is not commonly used for Join (Change Unknown).	"If it were 45, it would be 30. But it's 47, so it's 2 less. 28."

Invented Algorithms. Page 74 [5].

Professor David Scott, is the comfort level of elementary teachers with mathematics. Today's teaching requirement to become an elementary teacher is to major in education, but not any particular subject. Thus, many elementary teachers know enough about a lot of subjects to be certified, but do not know a specific subject in depth, especially mathematics. This leads to teachers not being comfortable with the concept of understanding the underlying processes that are going on, because teachers may be mathematically inept themselves. In conclusion, more extensive research is needed to get a better examination of CGI's capabilities.

IV. Intended Audience

This paper is directed to future and current educators. The main points of CGI are to realize that many different strategies to get to the same answer are acceptable and people learn in specific identifiable stages at a young age. I have attempted to summarize the book Children's Mathematics Cognitively Guided Instruction, combined with other sources, to give an overview and introduction of CGI to the intended audience. I recommend this book for further details about higher level topics such as multiplication and division. It also has video examples of individual students solving problems and also students solving problems in a CGI classroom setting.

V. Conclusion

Cognitively Guided Instruction can be used all the way up to basic algebra to solve for missing variables. CGI leads to great problem solving skills that can benefit any student, regardless of what level of math they are learning. CGI is a style of teaching based on years of research showing that people learn beginning mathematical concepts linearly. That is, there are clear stages that are passed through in a particular order. Fundamentally, the problem solving process comes first and number sentences are a consequence of this process used to represent work more efficiently. CGI is a great teaching strategy because "the NCTM standards were grounded on the process standards of CGI: problem solving, reasoning, connections, modeling, and communication" [6]. CGI is student focused; not lecture and booked based with daily homework. CGI focuses on the learning process and teaching great problem solving skills, instead of trying to memorize facts. This is not a very common strategy for teaching mathematics, but it is spreading in popularity among educators. Further research is necessary before more conclusions can be drawn on the effectiveness of CGI compared to other ways of teaching mathematics.

Acknowledgment

I would like to thank the Ripon College Math Department for supporting my personal growth in mathematics and mathematics education, especially professor David Scott (my academic and group advisor). I would also like to thank professor Judith Hankes, a professor of mathematics education at the University of Wisconsin – Oshkosh, who directed me in my final direction in researching CGI. Finally, I would like to thank my senior seminar peer group, Kristin Reedal and Sarah Weyer, for critiquing how I expressed my presentation.

References

[1] *Beyond Classical Pedagogy Teaching Elementary School Mathematics (Studies in Mathematical Thinking and Learning).* Mahwah: Lawrence Erlbaum, 2001.

[2] Borko, H., and R. Putnam. "Learning to Teach." *Handbook of Educational Psychology* (1996): 673-707. Macmillan. The Gale Group.

[3] Carpenter, Thomas P. "Using Knowledge of Children's Mathematics Thinking in Classroom Teaching: An Experimental Study." *American Educational Research* (1988): 1-64.

[4] Carpenter, Thomas P., and Elizabeth Fennema. "Cognitively guided instruction: A knowledge base for reform in primary mathematics instruction." *Elementary School Journal* 97.1 (1996): 3.

[5] Children's mathematics cognitively guided instruction. Portsmouth, NH: Heinemann, 1999.

[6] Hankes, Judith. "Research Question." Message to the author. 25 Jan. 2010.

[7] "Promising Practices Network | Programs that Work | Cognitively Guided Instruction (CGI)." *Promising Practices Network on Children, Families and Communities | Home*. Web. 28 Jan. 2010. <http://www.promisingpractices.net/program.asp?programid=114>.

[8] Secada, Walter G., Elizabeth Fennema, and Lisa B. Adajian. *New directions for equity in mathematics education*. New York: Cambridge UP, 1995.

[9] Silver, Edward A. *Teaching and Learning Mathematical Problem Solving Multiple Research Perspectives*. Mahwah: Lawrence Erlbaum, 1985.

Tyler Ruppert was born and raised in Clintonville, WI on March 13, 1988. He graduated from Clintonville High School and then attended Ripon College in Ripon, WI. At Ripon College he achieved a mathematics major and a secondary education minor in January 2011.

He has worked at Torborg's Lumber for eight consecutive summers as a Yard Worker. Upon graduation, he plans to teach secondary mathematics in northern Wisconsin.

Mr. Ruppert is interested in sports, especially football and rugby. He also enjoys hunting, fishing, snowboarding, and cutting firewood.

Appendix

TABLE 4
CLASSIFICATION OF WORD PROBLEMS

Problem Type:	Join	Separate	Part-Part-Whole	Compare
	(Result Unknown) Connie had 5 marbles. Juan gave her 8 more marbles. How many marbles does Connie have altogether?	(Result Unknown) Connie had 13 marbles. She gave 5 to Juan. How many marbles does Connie have left?	(Whole Unknown) Connie has 5 red marbles and 8 blue marbles. How many marbles does she have?	(Difference Unknown) Connie has 13 marbles. Juan has 5 marbles. How many more marbles does Connie have than Juan?
	(Change Unknown) Connie has 5 marbles. How many more marbles does she need to have 13 marbles altogether?	(Change Unknown) Connie had 13 marbles. She gave some to Juan. Now she has 5 marbles left. How many marbles did Connie give to Juan?	(Part Unknown) Connie has 13 marbles. 5 are red and the rest are blue. How many blue marbles does Connie have?	(Compare Quantity Unknown) Juan has 5 marbles. Connie has 8 more than Juan. How many marbles does Connie have?
	(Start Unknown) Connie had some marbles. Juan gave her 5 more marbles. Now she has 13 marbles. How many marbles did Connie have to start with?	(Start Unknown) Connie had some marbles. She gave 5 to Juan. Now she has 8 marbles left. How many marbles did Connie have to start with?		(Referent Unknown) Connie has 13 marbles. She has 5 more marbles than Juan. How many marbles does Juan have?

Classification of Word Problems. Page 12 [5].

TABLE 5
PROBLEMS THAT ELICIT GROUPING BY TEN STRATEGIES

Problem	Strategies		
	Counting by Ones	Counting by Tens	Direct Place Value
Multiplication John has 6 pages of stickers. There are 10 stickers on each page. He also has 4 more stickers. How many stickers does he have in all?	Makes 6 groups of counters with 10 counters in each group. Adds 4 additional counters and counts the set by ones.	Counts, "10, 20, 30, 40, 50, 60, 61, 62, 63, 64," keeping track on fingers.	Says, "64. 6 tens is 60 and 4 more is 64."
Measurement Division Mary has 64 stickers. She pastes them in her sticker book so that there are 10 stickers on each page. How many pages can she fill?	Counts out 64 counters and puts them into groups with 10 in each group. Counts the number of groups.	Counts, "10, 20, 30, 40, 50, 60," putting up a finger with each count. Counts fingers to get answer 6.	Says, "6. There are 6 tens in 60."

Problems That Elicit Grouping By Ten Strategies. Page 63 [5].

TABLE 6
GROUPING/PARTITIONING, RATE, PRICE, MULTIPLICATIVE COMPARISON PROBLEMS

Problem Type:	Grouping/Partitioning	Rate	Price	Multiplicative Comparison
Multiplication	Gene has 4 tomato plants. There are 6 tomatoes on each plant. How many tomatoes are there all together?	Ellen walks 3 miles an hour. How many miles does she walk in 5 hours?	Pies cost 4 dollars each. How much do 7 pies cost?	The giraffe in the zoo is 3 times as tall as the kangaroo. The kangaroo is 6 feet tall. How tall is the giraffe?
Measurement Division	Gene has some tomato plants. There are 6 tomatoes on each plant. All together there are 24 tomatoes. How many tomato plants does Gene have?	Ellen walks 3 miles an hour. How many hours will it take here to walk 15 miles?	Pies cost 4 dollars each. How many pies can you buy for $28?	The giraffe is 18 feet tall. The kangaroo is 6 feet tall. The giraffe is how many times taller than the kangaroo?
Partitive Division	Gene has 4 tomato plants. There are the same number of tomatoes on each plant. All together there are 20 tomatoes. How many tomatoes are there on each tomato plant?	Ellen walked 15 miles. It took her 5 hours. If she walked the same speed the whole way, how far did she walk in one hour?	Jan bought 7 pies. He spent a total of $28. If each pie cost the same amount, how much did one pie cost?	The giraffe is 18 feet tall. She is 3 times as tall as the kangaroo. How tall is the kangaroo?

Grouping/Partitioning, Rate, Price, Multiplicative Comparison Problems. Page 48 [5].

Summation
May 2010, pp. 28-35
http://ripon.edu/macs/summation

The Shapes of Fibonacci and the Golden Ratio

Brian J. Fatla
Department of Mathematics and Computer Science – Ripon College

***Abstract*—The Fibonacci sequence is used to find the Golden Ratio, represented by the Greek letter phi (ϕ). Both the Fibonacci sequence and Golden Ratio can be used to make certain geometric shapes. These shapes and other relationships are found in many places outside of mathematics, such as art, architecture, and nature. This paper will describe those shapes and show some instances where they are found in nature.**

***Key Words*—Fibonacci sequence, golden ratio, nature, phi**

I. The Fibonacci Numbers

The Fibonacci sequence is a sequence of numbers made famous by Leonardo of Pisa, known simply as Fibonacci. He presented a rather unrealistic rabbit population problem, and in his answer, he discovered the Fibonacci numbers.

A. *The Rabbit Problem*

The problem that Fibonacci posed is as follows: A pair of newly born rabbits is left alone to reproduce. They are unable to reproduce after one month as they still need time to mature, but after the second month, the two have a pair of babies, one male and one female. They continue having babies every month, always with one male and one female. Like their parents, the offspring cannot reproduce for the first month, but in the second month, they will also have a pair of babies, one male and one female. We want to know how many pairs there will be after one year.

This problem assumes that the rabbits never die, there is an infinite amount of food and space, the parents will always produce one male and one female, and those offspring will start reproducing after two months.

At month 0, the starting month, we have 1 pair of rabbits. At month one, we still have that 1 pair as they need to mature before reproducing. At month two, the first pair would reproduce and we would have 2 pairs. At month three, the first pair would reproduce again while the new pair would need to mature, leaving us with 3 pairs. We can continue this process and see that for the first twelve months, starting at month 0, we'd have the following number of pairs: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, and 233. These are the first thirteen Fibonacci numbers. Figure 1 helps visualize this scenario.

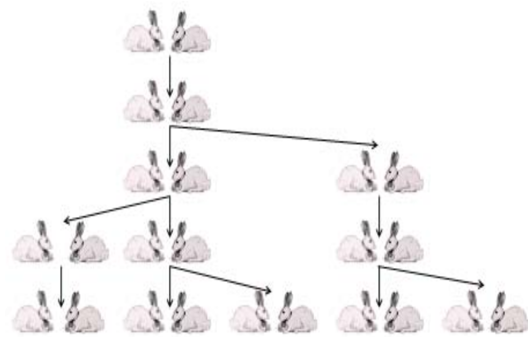

Figure 1. Here is the rabbit problem Fibonacci presented. As you can see, the total number of pairs each month follows the Fibonacci sequence [1].

These numbers result from a recursive relationship. As the problem shows, we can select any two consecutive numbers in the relationship, add them, and it will result in the number immediately following the two numbers. This relationship can be expressed as $F_n = F_{n-1} + F_{n-2}$, where $F_0 = 1$ and $F_1 = 1$. For example, to find F_6, we add F_4 and F_5, which are 5 and 8, respectively. Therefore, $F_6 = F_4 + F_5 = 5 + 8 = 13$. It's important to note that this relationship can be started at $F_0 = 0$ and $F_1 = 1$ or $F_0 = 1$ and $F_1 = 1$, it simply depends on the situation. In the rabbit problem, it would not make sense to begin with 0 pairs, so we start at $F_0=1$.

B. *Binet's Formula*

Suppose we want to find the 20th Fibonacci number, or even higher, say the 100th Fibonacci number. It would be extremely time consuming and inefficient to have to go through all the previous Fibonacci numbers to find F_{100}. To avoid this, we can use what is known as Binet's Formula. In 1843, Jacques Binet was able to express the Fibonacci numbers in a closed form, allowing us to find any Fibonacci number we wanted simply by plugging the nth position value into a formula. The formula he derived was

$$F_n = \frac{\phi^n - \left(-\frac{1}{\phi}\right)^n}{\sqrt{5}} \tag{1}$$

where ϕ (phi) is the golden ratio, which is equal to $(1+\sqrt{5})/2$. So, if we wanted to find F_{20}, we simply let n=20. This gives

Work presented to the college March 26, 2010.
B. J. Fatla is currently a student at Ripon College. He will be graduating on May 16, 2010 (phone: 715-889-18311 e-mail: fatlab@ripon.edu).

$$F_{20} = \frac{\phi^{20} - \left(-\frac{1}{\phi}\right)^{20}}{\sqrt{5}} \tag{2}$$

Remember that $\phi \approx 1.618$, so we have

$$F_{20} = \frac{1.618^{20} - \left(-\frac{1}{1.618}\right)^{20}}{\sqrt{5}} \tag{3}$$

This gives us $F_{20}=15127/\sqrt{5}$, which means that $F_{20}=6765$. To appreciate why this formula is so useful, consider F_{50} and F_{100}. Plugging in the appropriate *n* values, we see that $F_{50}=12586269025$ and $F_{100}\approx 3.54*10^{20}$. Binet's Formula makes it much easier to find any Fibonacci number [2].

II. The Golden Ratio

A direct consequence of the Fibonacci sequence is the golden ratio. If we divide consecutive Fibonacci numbers and consider the results as we move through the sequence, the quotients seem to converge to some value as we approach infinity,

$$\lim_{n \to \infty} \frac{F_{n+1}}{F_n} = \phi \tag{4}$$

as seen in Table 1.

As the Fibonacci numbers get bigger and bigger, it becomes clear that the ratios approach some number. This value happens to be the golden ratio ϕ. Decimally, the golden ratio is approximately 1.6180339887498948482045868343656…. The golden ratio appears in many places, including the Fibonacci sequence, a variety of shapes, and many instances in nature, art, and architecture [2].

TABLE 1
RATIO OF CONSECUTIVE FIBONACCI NUMBERS

Fibonacci Number	1	1	2	3	5	8	13	21
Quotient		1	2	1.5	1.667	1.6	1.625	1.615

III. Golden Shapes

Golden shapes are shapes that have the golden ratio in them, typically in ratios of lengths and sides. Here we will discuss some of the most common ones.

A. *Golden Section*

The simplest and perhaps most observed golden shape is the golden section. The golden section can be seen in Figure 2. The line segment is divided in such a way that the segment as a whole compared to the larger segment is equal to the larger segment compared to the shorter segment. This equation expresses the relationship,

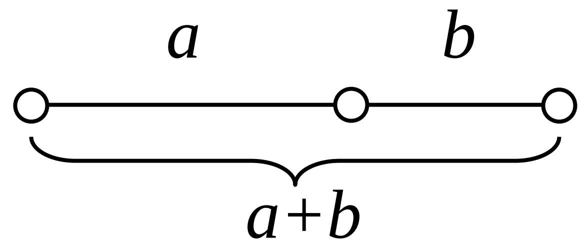

Figure 2. The golden section is the simplest golden shape. The relationship is *a* + *b* is to *a* as *a* is to *b* [1].

$$\frac{a+b}{a} = \frac{a}{b} \tag{5}$$

where *a* is the longer of the two lengths, as shown in Figure 2.

To see how the golden section relates to the golden ratio, we will let *b* equal 1 and solve for *a*. This gives

$$\frac{a+1}{a} = \frac{a}{1} \tag{6}$$

or $a^2 - a - 1 = 0$. We can use the quadratic formula to get the two solutions of $a_1=(1-\sqrt{5})/2$ and $a_2=(1+\sqrt{5})/2$. Because we are looking at a length, a negative number would not make sense, so we want to use $a_2=(1+\sqrt{5})/2$, which is the golden ratio.

B. *Golden Rectangle*

Another common golden shape is the golden rectangle, a rectangle where the ratio of the long side to the short side is the golden ratio. A distinctive feature of the golden rectangle is when we take a square out of the golden rectangle, the remaining part will again be a golden rectangle, as seen in Figure 3. This process can be continued infinitely many times and is used to create the golden spiral, which will be discussed later.

The golden rectangle has piqued the interest of many psychologists and other non-mathematicians as it is believed that the golden rectangle holds a special aesthetic appeal. The earliest interest in the psychological appeal of the golden

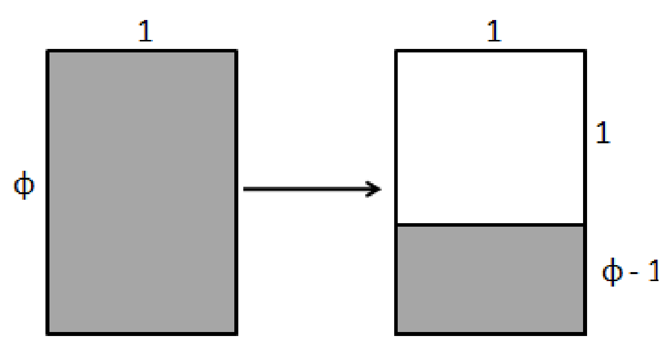

Figure 3. The golden rectangle is shown on the left. On the right, a square was taken out (white). When we take out a square, we are left with yet another golden rectangle (the shaded section on the right) [3].

rectangle came in 1876 from a German experimental

TABLE 2
FECHNER'S SURVEY RESULTS

Ratio of Sides on Rectangle	Percent for Best Rectangle	Percent for Worst Rectangle
1 to 1	3	27.8
5 to 6	0.2	19.7
4 to 5	2	9.4
3 to 4	2.5	2.5
20 to 29	7.7	1.2
2 to 3	20.6	0.4
21 to 34 (1.619)	35	0
13 to 23	20	0.8
1 to 2	7.5	2.5
2 to 5	1.5	35.7

psychologist named Gustav Fechner. Fechner made thousands of measurements of commonly seen rectangles, such as playing cards, writing pads, books, windows, etc. He found that most had the ratio of length to width that was close to the golden ratio.

In addition to these measurements, Fechner also conducted a survey of 228 men and 119 women where he had them choose the most and least aesthetically pleasing rectangle out of a group of 10 rectangles. As Table 2 shows, the rectangle closest to the golden rectangle received the highest number of votes for most pleasing rectangle and the lowest amount of votes for the least pleasing [2].

It is surprisingly simple to construct a golden rectangle. Begin with a square. Then extend any two parallel sides a little past the square. Next, on one of the extended sides, bisect the length of the square. Then use a compass to draw an arc from the corner opposite the bisected point to an extended side. Now draw a line connecting the two extended lengths at the point where the arc and the extended side intersect,

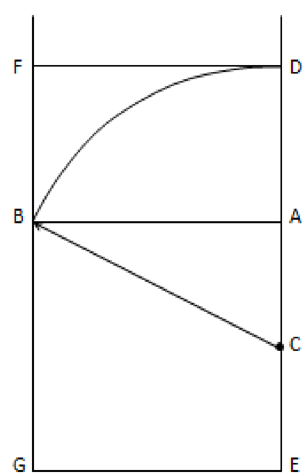

Figure 4. Despite its aesthetic appeal, it is relatively simple to draw a golden rectangle [3].

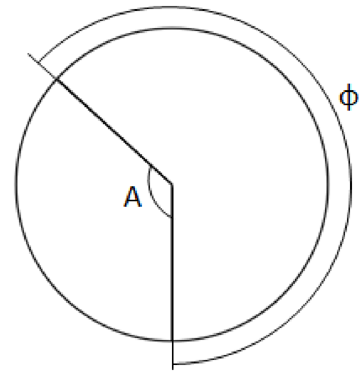

Figure 5. The golden angle is approximately 137.5°, represented here by A.

making sure the connecting line is perpendicular to the extended sides. Figure 4 provides a good example of these steps [4].

To prove this is the golden rectangle, let the lengths of the square equal 1. Now find BC by using the Pythagorean Theorem, which states that $c^2 = a^2 + b^2$. In this case, c=BC, a=AB, and b=AC. We know AB=1, and because C marks the point of bisection, we know that AC=1/2. This gives $c^2=1^2+(1/2)^2=1+(1/4)=5/4$. Now we have $c^2=5/4$, and by taking the square root of both sides, we get $c=\sqrt{5}/2$. By our construction, we also know that $BC=CD=\sqrt{5}/2$, and we also know that AC=CE=1/2. Now we can simply add CD and CE to find DE: $\sqrt{5}/2+(1/2) =(1+\sqrt{5})/2=DE$. This is the golden ratio, thus proving EDFG is a golden rectangle.

C. *Golden Angle*

The golden angle is a relatively easy shape to describe. Think of curling the golden line segment into a circle. Now we can easily find the golden angle.

Because we know the line segment has a ratio of ϕ to 1, the longer arc length is ϕ and the shorter length is 1. Looking at Figure 5, to find A, we first have to find the other angle. We simply divide 360° by ϕ. This is approximately equal to 222.49223559…°. Now to get the golden angle, A in figure 5, we simply subtract 222.49223559…° from 360°. This gives us approximately 137.5077640…°, which is the golden angle.

D. *Golden Pentagon and Pentagram*

The Pythagoreans of the fifth century B.C. had a special symbol to distinguish who was a member of the Pythagoreans. The symbol was the pentagram, a star-like shape, seen in Figure 6. Pythagoras found that the ratio of a diagonal of a regular pentagon to any side could not be expressed as a rational number. In fact, the irrational number he found was the golden ratio [2]. To prove that a diagonal compared to a side has the ratio of ϕ, look at Figure 6 and assume that the sides of the pentagon are equal to 1. We know that every angle is 108°. First consider triangle *ABC*. Because we have two sides and one angle, we can use the Law of Cosines ($c^2=a^2+b^2-2ab\cos\Theta$) to find the length of the diagonal. Thus we get $c^2=1^2+1^2-2(1)(1)\cos 108°$. Doing some algebra and using the fact that $\cos 108°=-(\sqrt{5}-1)/4$, we find that c=2*(-(√5-

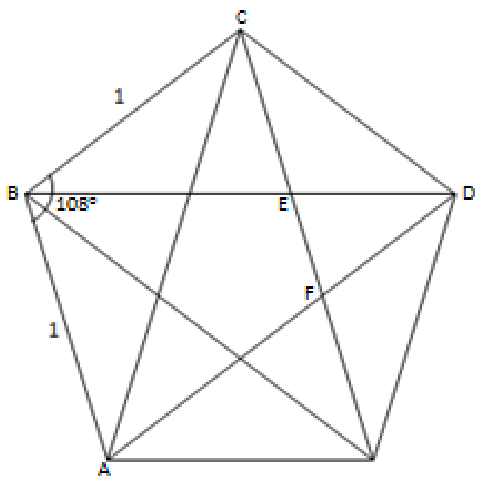

Figure 6. When all the diagonals of a regular pentagon are connected, the result is a golden pentagram. Within the figures are numerous examples of the golden ratio, including the ones described here [5].

1)/4), which equals (1+√5)/2, which is the golden ratio.

Now let's look at the triangle *DEF*. First, we can let length *EF* equal 1, however, we have to find two more angles so we can use the Law of Sines (sinD/d=sinE/e=sinF/f) to find *DF*. We can use triangle *ABC* to determine some missing angles.

Because we have a regular pentagon, we know that the diagonals are parallel to the sides opposite them. This means that triangle *ABC* is going to be isosceles, and by the definition of a regular pentagon, we know that the angle *ABC* is 108^o, so the smaller angles are going to equal 36^o. Now, out of the three angles formed by connecting the diagonals, we only have angle *EDF* remaining. We know that the three angles added together have to equal 108^o, so we subtract the two 36^o angles we found from 108^o and find that the middle angle is also 36^o. We now know that angle *EDF* is 36^o but we still need at least one more angle before using the Law of Sines. Again using the fact that a diagonal is parallel to the side opposite it, we know that we're dealing with another isosceles triangle, so we know that the remaining two angles must be 72^o.

Finally, we have two angles and a side so we can use the Law of Sines, where D is angle *EDF*, d is length *EF*, E is angle *DEF*, and e is length *DF*. Substituting appropriately, we get sin72^o/e= sin36^o/1. Solving for e, we get (1+√5)/2, which is the golden ratio.

One last triangle we can look at is triangle *CDE*. This time we will let *DE* equal 1, and by now we know that angles *ECD* and *CDE* are equal to 36^o and the angle *CED* equals 108^o. Because of these facts, we have two sides and three angles, so we can use the Law of Sines or the Law of Cosines. We'll just use the Law of Cosines in this example. Substituting in correctly, we see that E^2=1^2+1^2-2(1)(1)cos108^o, and when we solve for *E*, we get *E*=(1+√5)/2, which again is the golden ratio.

E. Golden Spiral

The golden spiral is a logarithmic spiral having the equation

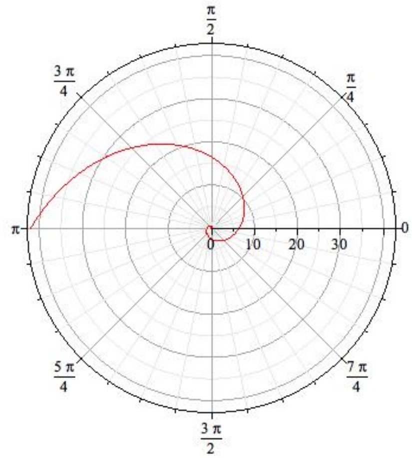

Figure 7. The golden spiral is a logarithmic spiral that grows at a constant rate equal to the golden ratio [2].

$$r = a \cdot \phi^{\frac{2\theta}{\phi}} \quad (7)$$

where *a* is a constant that alters the way the spiral looks [6]. Figure 7 shows a logarithmic golden spiral with ϴ ranging from -2π to 2π and a=-1. If *a* equaled 1, the spiral would be reflected across the *x* and *y* axes.

There are two ways that the golden spiral can be drawn approximately. These approximations are used more often than the actual golden spiral due to the ease with which they can be created and the fact that the approximations are generally good enough for the desired applications.

Fibonacci Spiral

The first approximation uses the Fibonacci sequence and is

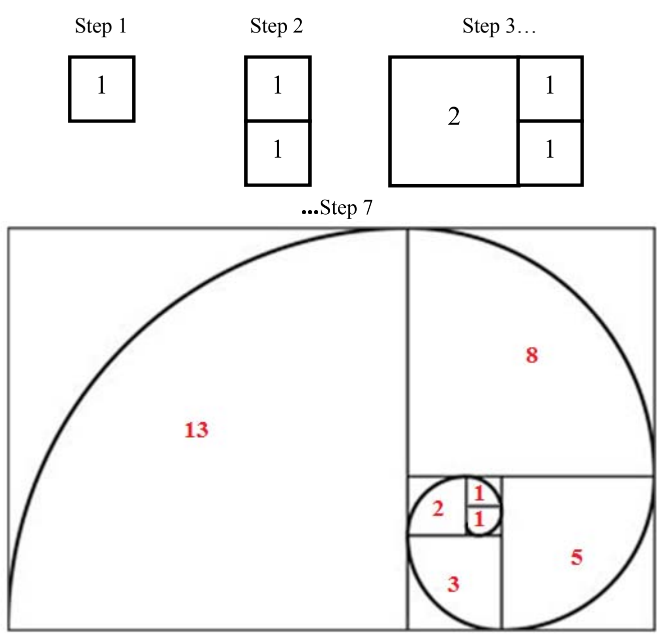

Figure 8. Golden spirals are easily approximated by using squares with lengths equal to the Fibonacci numbers and drawing quarter circles in each one [2].

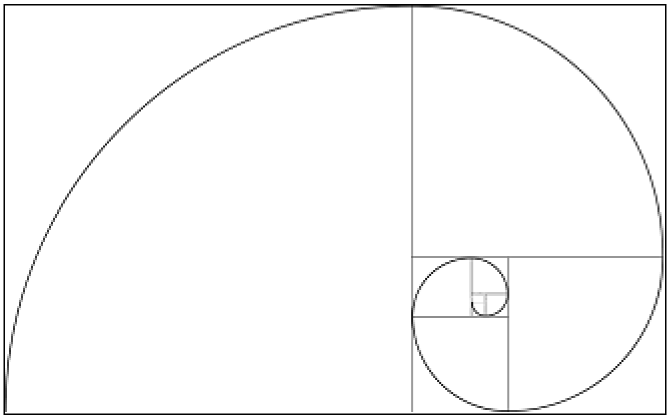

Figure 9. The golden spiral can also be approximated by taking squares out of multiple golden rectangles. This method provides the best approximation [2].

therefore called the Fibonacci spiral. To make a Fibonacci spiral, we start out by drawing a unit square. Draw a square identical and adjacent to that square. Now we can draw a square with length 2 next to those. Adjacent to the 2 and 1 length squares, we can draw a square with length 3. We can continue this for as long as we want, and when we have a rectangle big enough for our purpose, we simply draw quarter circles in each square to get a good approximation of the golden spiral, as shown in Figure 8 [2].

With the Fibonacci spiral, the approximation becomes better as the squares become larger. This can be explained by using the limit of consecutive quotients. As was stated earlier, the golden spiral grows at a rate of ϕ, and the Limit of Consecutive Quotients says that the quotient of two consecutive Fibonacci numbers gets closer to ϕ as we approach infinity. This means that our squares will become better approximations as we approach infinity. For example, looking at Figure 7, we have squares of length 2 and 1 early in the spiral. The quotient of those is 2. The last squares we drew were of length 13 and 8. The quotient of those is 1.625, which is clearly much closer to ϕ [2].

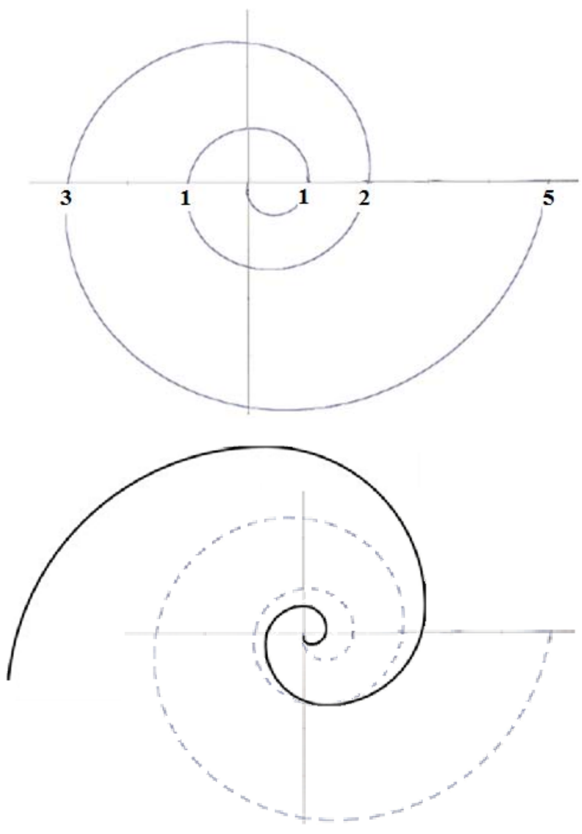

Figure 10. Top: If one were to trace the spiral of the nautilus shell and place it on a graph, the points where the spiral crosses the x-axis follows the Fibonacci sequence [1].
Bottom: Many claim that the spiral inside the nautilus shell is the golden spiral, when it's quite clear that this isn't the case [8].

Golden Spiral Approximation

Another way to approximate the golden spiral is to use golden rectangles instead of Fibonacci squares. As shown earlier, removing a square from a golden rectangle creates another smaller golden rectangle. We can continue doing this numerous times and we will get a group squares as shown in Figure 9. We simply draw the quarter circles again and we have a golden spiral approximation. This approximation is used most often because it provides a better approximation for most purposes [2].

IV. Nature

These shapes are mathematically interesting, but they are also seen in everyday life, most notably in nature, art and architecture. Here we will observe some of their occurrences in nature.

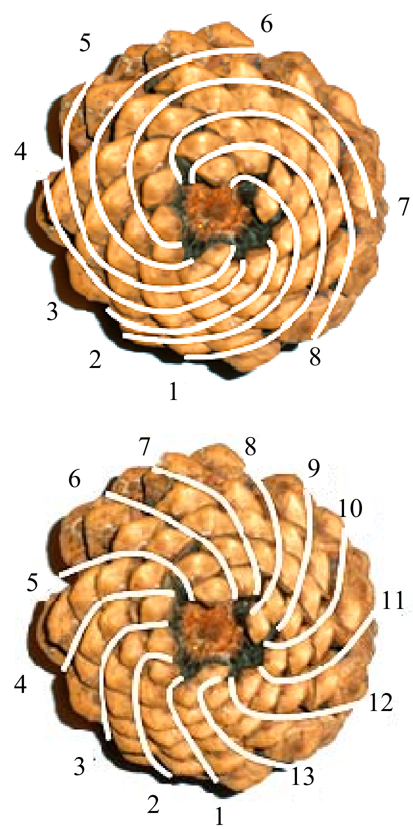

Figure 11. The pinecone has two distinct spirals, one going clockwise and one counterclockwise. If you count the number in each direction, both will be numbers in the Fibonacci sequence [1].

A. Nautilus Shell

Perhaps the most well known example of the Fibonacci sequence in nature is also the most misunderstood. While the nautilus shell does have the Fibonacci sequence in it, many people misinterpret where it's actually found.

When looking at a nautilus shell, we can see a distinct spiral. If we traced that spiral and placed it on a graph so that the first time it crosses the x-axis is at 1, as shown in Figure 10, we can see a pattern start to form for each time the spiral crosses the x-axis. If we take the absolute value of the x coordinates, we get 1, 1, 2, 3, and 5, or the first five Fibonacci numbers [2].

While this certainly is a Fibonacci relationship, this is not the way many people relate the nautilus shell to the Fibonacci sequence. Instead, many people claim that the actual spiral of the shell *is* the golden spiral.

If we look at Figure 10, we can clearly see that two spirals are not the same. While the golden spiral grows at a rate of approximately .618, the nautilus shell spiral grows at an average rate of 1.33 [8]. While the Fibonacci sequence is found in many aspects of nature, it's important to beware of false claims.

TABLE 3
VARIOUS PINECONE SPIRALS

Type of Pinecone	Spirals (counterclockwise to clockwise)
Whitebark pine	5-3
Limber pine	8-5
Sugar pine	8-5
Sliver pine	3-5
One-leaved pinon	3-5
Pinus edulis	5-3
Four-leaved pinon	5-3
Bristlecone pine	8-5
Foxtail pine	8-5
Bishop pine	8-13
Santa Cruz Island pine	5-8
Beach pine	8-13
Lodgepole pine	8-5
Torrey pine	8-5
Yellow pine	13-8
Jeffrey pine	13-5
Monterey pine	13-8
Knobcone pine	8-5
Digger pine	13-8
Coulter pine	13-8

B. Pinecone

Pinecones also have a Fibonacci relationship in them. If we look at the bottom of a pinecone, we can see that the scales form spirals, some in a clockwise direction and some in a counterclockwise direction. If we count the number of spiralsin each direction, we find that they equal numbers in the Fibonacci sequence, as Figure 11 shows. In the clockwise direction, there are 8 spirals, in the counterclockwise direction, there are 13, both numbers in the Fibonacci sequence [1].

Not all conifers have identical cones, and Table 3 shows numerous species and the number of spirals in the cones in each one. Notice that all the pinecones have spiral numbers in the Fibonacci sequence [2].

C. Flowers

In many cases, the head of a flower is made up of small seeds that are produced at the center and migrate outward to fill the space. Each new seed appears at a certain angle in relation to the preceding one. For example, if we look at Figure 12, the angle of the seeds growing in Flower 1 is 90°. Obviously this is not the most efficient way of filling space. In fact, if the angle between the appearances of each new seed is a portion of a turn which corresponds to a simple fraction, i.e. a rational number such as 1/4, 1/3, 3/4, etc., then the seeds will always grow in a straight line. To fill space more efficiently, seeds must grow in a way which results in a portion of a turn that is an irrational number. However, even

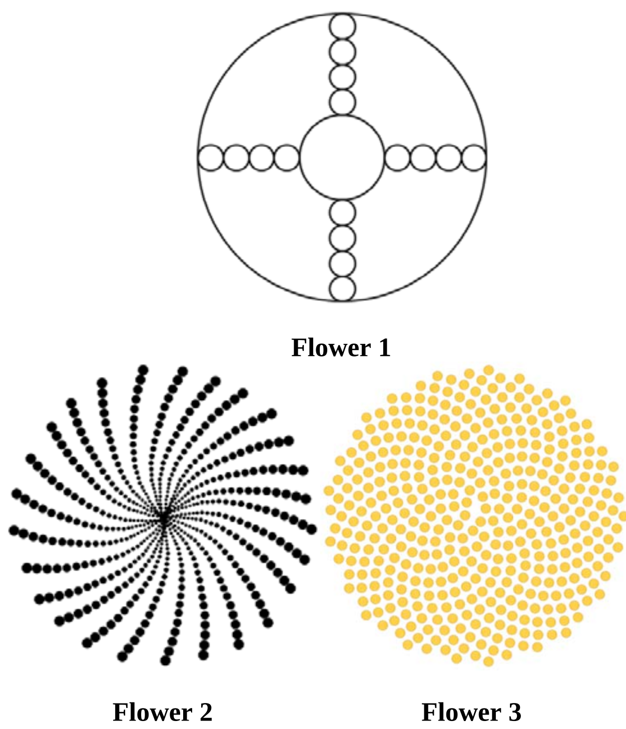

Figure 12. The way seeds grow is a very precise process. Flower 1 has seeds growing 90° apart, Flower 2 has seeds growing 135.6° apart (just 0.1° different from the golden angle), and Flower 3 has seeds growing approximately 137.5° apart, or the golden angle. When the seeds grow in an pattern using the golden angle, we see two distinct spirals [1].

with a simple irrational number, an arrangement like that shown on Flower 2 in Figure 12 is likely to occur. In order to optimize filling, seeds must grow in a way that is not well approximated by a fraction. This number is the golden angle of approximately 137.5^o. The angle has to be very precise as the slightest variations completely destroy the pattern. For example, in Flower 2 of Figure 12, the angle is just one tenth of a degree different than the golden angle at 137.6^o. It's only when the angle is the golden angle that seeds will optimize space and the two spirals that we see in Flower 3 develop [2].

The best example of this arrangement of seeds can be found on the sunflower. In addition to having seeds in this arrangement, we can add up the number of clockwise and counterclockwise spirals in a sunflower and they will generally be numbers in the Fibonacci sequence. As they mature,
sunflowers develop more spirals. About 82% of the time they take on the pairs of 21:34, 34:55, 55:89, or 89:144. Another 14% of the time the spirals will be numbers of the Lucas sequence, another recursive relationship that follows the same rules as the Fibonacci numbers. And yet another 2% of the time the number of spirals will be neither Fibonacci nor Lucas but will still follow the same recursive relationship. Only 2% of the time are the seeds not related to a Fibonacci-like sequence [2].

Flowers also have one other relation to the Fibonacci sequence; the number of petals they grow. Nearly 92% of the time, the number of petals on a flower will be equal to numbers in the Fibonacci sequence. Some examples include irises and lilies (3 petals), buttercups and wild roses (5), ragwort and certain daisies (13), and other daisies (34). In some cases, even when the number of petals isn't a Fibonacci number, they grow in sets that are numbers in the Fibonacci sequence. For example, some lilies have six petals, but they grow in two sets of three [2].

D. *Pineapple*

The hexagonal bracts on a pineapple form three different directions of rows, as shown in Figure 13. The number of rows formed in each direction is a Fibonacci number. In Figure 13, A has 5 rows, B has 8 rows, and C has 13 rows [2].

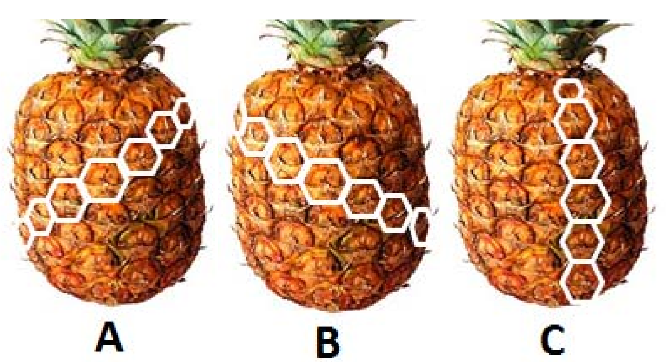

Figure 13. The number of rows for each different type of arrangement in a pineapple is Fibonacci numbers [1].

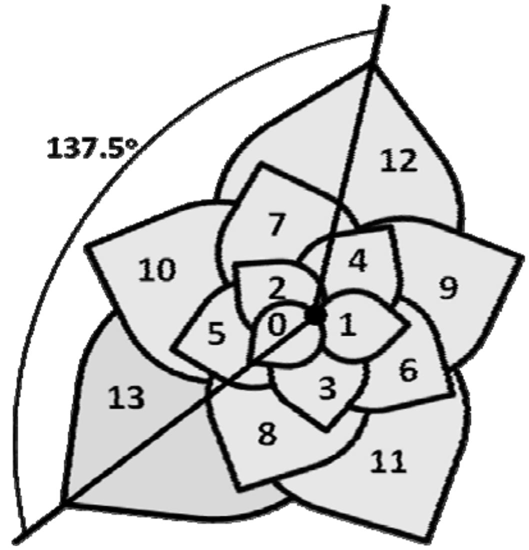

Figure 14. Many plants' leaves grow at an angle equal to the golden angle. This allows the maximum amount of sunlight to hit each leaf. Additionally, the number of leaves on a fully grown plant will typically be a Fibonacci number and the number of rotations it takes to get to the top will also many times be a Fibonacci number

E. *Phyllotaxis*

Finally, we have phyllotaxis, the arrangement of leaves on a stem. As Figure 14 shows, leaves on a plant generally grow in a way such that two consecutive leaves are approximately 137.5^o apart, which is the golden angle. By growing a distance apart that is equal to the golden angle, plants allow the maximum amount of sunlight to hit each of their leaves. Looking at Figure 14, we can see 13 leaves. With any other angle, the leaves would start to cover each other up much sooner [9].

Another way phyllotaxis is related to the Fibonacci sequence can be seen by looking at a fully grown plant and counting the number of leaves and the number of rotations it takes to get to the top. Many times the numbers will be Fibonacci numbers. Looking at Figure 14, we can see that, starting at 0 and going out to 13, it takes 5 spirals to get to the 13th leaf, and both 5 and 13 are Fibonacci numbers. Table 4 shows some plants and the number of leaves and rotations they have [9].

V. Conclusion

Binet's Formula provides the means to find any Fibonacci number we want without relying on recursion. The golden ratio is also derived from Binet's Formula, which adds even more uses to the Fibonacci sequence.

The Fibonacci numbers and golden ratio are seen in a variety of shapes, known as golden shapes. These include the golden rectangle, angle, pentagram, and spiral.

Golden shapes can be found in many places in nature, such as nautilus shells, pinecones, flowers, pineapples, and plant structures.

There are even more applications of the Fibonacci numbers, including uses in financial analysis, Pascal's triangle, and observations in art and architecture. However, not all claims the Fibonacci sequence are valid. For example, here are two websites that claim that nautilus shells grow at a rate equal to the golden spiral:

http://www.sciencemall-usa.com/nautilus1.html

http://www.world-mysteries.com/sci_17.htm

While a simple rabbit problem gave birth to the Fibonacci sequence, its uses have expanded within and beyond the realm of mathematics.

Acknowledgment

I would like to thank Professors Karl and Diane Beres for all their help in this research, Professors Scott and Hess for their insightful teaching throughout college, Mr. Gretzinger for first getting me interested in mathematics, Mrs. Bredel for some of the best teaching I've ever had, and all the students of Ripon College who have helped me when I needed it.

References

[1] Knott, R. (2008, December 16). *Fibonacci Numbers and Nature*. Retrieved from http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html. Retrieved March 17, 2001.

[2] Posamentier, A.S., & Lehmann, I. (2007). *The (Fabulous) Fibonacci Numbers*. Amherst, NY: Prometheus Books. Retrieved March 17, 2001.

[3] Ross, K. (1999). *The Golden Ratio and the Fibonacci Numbers*. Retrieved from http://www.friesian.com/golden.htm. Retrieved March 17, 2001.

[4] Anneberg Media. (2010). *Golden Rectangles*. Retrieved from http://www.learner.org/workshops/math/golden.html. Retrieved March 17, 2001.

[5] Serras, H. (2009, November 2). *The Golden Section The Golden Triangle The Regular Pentagon and Pentagram The Dodecahedron*. Retrieved from http://cage.ugent.be/~hs/polyhedra/dodeca.html. Retrieved March 17, 2001.

[6] Barrus, D. (1998, March 31). *Golden Spiral*. Retrieved from http://mathforum.org/library/drmath/view/51760.html. Retrieved March 17, 2001.

[7] Kursat, A. (1999, December 4). *Foundations of Geometry*. Retrieved from http://jwilson.coe.uga.edu/EMT668/EMAT6680.F99/Erbas/KURSATgeometrypro/golden%20spiral/logspiral-history.html. Retrieved March 17, 2001.

[8] Akkana, M. (2007, February 8). *The Fibonacci spiral and the nautilus*. Retrieved from http://www.shallowsky.com/blog/science/fibonautilus.html. Retrieved March 17, 2001.

[9] Atela, P, & Christope, G. (2003, March). *Phyllotaxis*. Retrieved from http://www.math.smith.edu/phyllo/. Retrieved March 17, 2001.

TABLE 4
PLANT LEAVES AND ROTATIONS

Rotation to Number of Leaves	Plants
1:2	elm, linden, lime, grasses
3:8	asters, cabbages, poplar, pear, hawkweed, roses
1:3	alders, birches, sedges, beech, hazel, blackberry, grasses
2:5	roses, oak, apricot, cherry, apple, holly, plum, groundsel
8:21	fir trees, spruce
5:13	pussy willow, almond
13:34	pine trees

Brian J. Fatla was born in Crandon, WI on December 25, 1987. He will be receiving a Bachelor of Arts degree in mathematics with a minor in economics from Ripon College, Ripon, WI 54971 in May of 2010.

He has held various summer jobs, including a lab worker at Northern Lakes Service in Crandon, WI, a merchandiser for Pepsi in Rhinelander, WI, to currently being a manager at Little Caesars in Ripon, WI. Upon graduating, he hopes to work at a business firm in finance or marketing.

Mr. Fatla is currently a member of the National Honor Society and Spanish Honor Society.

Summation
May 2010, pp. 36-40
http://ripon.edu/macs/summation

Lights Out: Solutions Using Linear Algebra

Matthew A. Madsen
Department of Mathematics and Computer Science – Ripon College

***Abstract*—Microcontrollers are seen everywhere in everyday life. Items such as cell phones, remotes, and electronic games use them. This paper investigates the electronic game Lights Out that uses a microcontroller and explains how one could compute a winning strategy for the game. It demonstrates a couple of different methods and works through the proofs of them.**

***Key Words*—Lights Out, Linear Algebra, Microcontroller**

I. Introduction

MICROCONTROLLERS are small computers on a single integrated circuit consisting internally of a relatively simple CPU, clock, timers, I/O ports, and memory. They are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, power tools, and toys. The device that this paper is on is the electronic game Lights Out.

In this paper, I will go over several different methods that can be used to find a solution for the game Lights Out. I will first go through a mathematical method that involves the uses of linear algebra to find a winning strategy using the fewest number of moves. Then I'll talk about a method that doesn't involve the use of an electronic device and one that was described to me by one of my colleagues.

II. Lights Out Rules and Observations

A. Basic Rules

Lights Out is an electronic game manufactured by Tiger Toys in 1995. It consists of 25 buttons, each of which can be lit up, which form a 5 by 5 array. Each button can be in one of two states (on or off). At the start of play, a pattern of lit buttons is chosen by the microcontroller. By pressing the buttons, the player can alter the state of some subset of the buttons according to the rule. The effect of pressing a button is to toggle the state of that button, and its immediate vertical and horizontal neighbors. The goal of the game is to turn all the buttons to the off state.

The game has three modes. In the first mode, you are presented with 50 increasingly difficult puzzles. You have to solve one puzzle before you can move on to the next. The goal of this game is to solve each initial set-up in the minimum possible number of moves; however you are allowed to go over this minimum by ten. The second mode gives you another 1000 puzzles to solve. Mode three allows the player to enter your own puzzles, however not all the possible configurations of states are solvable. This will be demonstrated later in the paper.

B. Observations

The first initial observation about the game is that each button needs to be pressed no more than once. This observation comes from the fact that pressing a button twice is like not pressing it at all. Since pressing a button toggles the state of that button and of its immediate vertical and horizontal neighbors, pressing that same button again will reverse the states and toggle the buttons back to their original states. The key consequence of this observation is that all the calculations can be done in modulo 2. That is, pressing a button an even number of times is like not pressing it at all and pressing it an odd number of times is the same as pressing it only once. This consequence will be very important in calculating the solutions mathematically.

The second initial observation is that the state of each button only depends on how many times it and its immediate vertical and horizontal neighbors are pressed. This observation indicates is that the order in which you press the buttons is irrelevant to the resulting configuration.

A third observation comes directly from the first observation. If we start with the board completely off and press a sequence of buttons to get a configuration, then starting with that configuration and pressing the same sequence of buttons will result in all the lights turning off.

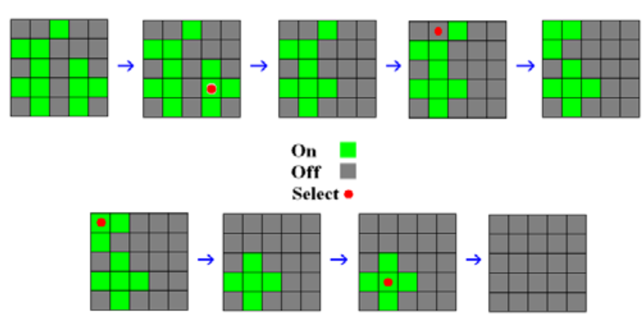

Figure 1. An example of how to play the game. The green squares represent lights that are on. The red dots are the selection of the play. This starting configuration can be solved with only pressing four buttons

Work presented to the college March 23, 2010 as the second talk in a series of three talks about microcontrollers

Matthew A. Madsen currently pursuing his baccalaureate degree in mathematics at Ripon College in Ripon, WI 54971 USA (e-mail:madsenm@ripon.edu).

III. Linear Algebra Solution

One way to guarantee a solution in the fewest number of buttons pressed is to use linear algebra. This can be done with basic matrix operations, some Gauss-Jordan elimination, and an understanding of the column and null space of a matrix.

Since there are two states to the buttons, on and off, we can do all of our calculations in modulo 2 by letting one represent on and zero represent off. We can then think of the 5 by 5 array as a 25 by 1 vector, as in

$$\vec{b} = (b_1, b_2, b_3, \cdots, b_{24}, b_{25})^T \tag{1}$$

where b_1, b_2, b_3, … ,b_{24}, b_{25} will represent the state of each button related to Fig. (2). This vector will be referred to as the configuration of the array. A configuration $\vec{b}$ is obtained by pressing a sequence of buttons, which we will denote as

$$\vec{x} = (x_1, x_2, x_3, \cdots, x_{24}, x_{25})^T \tag{2}$$

where x_1, x_2, x_3, … ,x_{24}, x_{25} will represent whether that particular button is pressed. If we start with all the lights out and the configuration $\vec{b}$ is obtained by strategy $\vec{x}$, then

$$\begin{aligned}
b_1 &= x_1 + x_2 + x_6 \\
b_2 &= x_1 + x_2 + x_3 + x_7 \\
b_3 &= x_2 + x_3 + x_4 + x_8 \\
\vdots &= \qquad \vdots \\
b_{13} &= x_8 + x_{12} + x_{13} + x_{14} + x_{18} \\
\vdots &= \qquad \vdots \\
b_{24} &= x_{19} + x_{23} + x_{24} + x_{25} \\
b_{25} &= x_{20} + x_{24} + x_{25}.
\end{aligned} \tag{3}$$

It is now straightforward to rewrite this system of linear equations as the matrix product $A\vec{x} = \vec{b}$, where A is the 25 by 25 matrix

$$A = \begin{bmatrix} Z & I & O & O & O \\ I & Z & I & O & O \\ O & I & Z & I & O \\ O & O & I & Z & I \\ O & O & O & I & Z \end{bmatrix}. \tag{4}$$

In this matrix, I is the 5 by 5 identity matrix, O is the 5 by 5 matrix of all zeros, and Z is the matrix

$$Z = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{5}$$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Figure 2. How the buttons on a 5 by 5 are viewed as a 25 by 1 vector

A key observation to make about Z is that it is symmetric, and therefore A is also symmetric. To find a solution for the A given configuration $\vec{b}$, we must find the strategy $\vec{x}$ that satisfies $A\vec{x} = \vec{b}$. To do this, we will prove two theorems.

A. Theorem 1

Given a starting configuration, we first need to check to see if the configuration is winnable. We say that configuration $\vec{b}$ is winnable if there is a strategy $\vec{x}$ that satisfies $A\vec{x} = \vec{b}$.[1] Therefore, Theorem 1 states a configuration $\vec{b}$ is winnable if and only if it is orthogonal to the two vectors $\vec{n}_1$ and $\vec{n}_2$, where

$$\vec{n}_1 = (0,1,1,1,0,1,0,1,0,1,1,1,0,1,1,1,0,1,0,1,0,1,1,0,0)^T \tag{6}$$

and

$$\vec{n}_2 = (1,0,1,0,1,1,0,1,0,1,0,0,0,0,0,1,0,1,0,1,1,0,1,0,1)^T. \tag{7}$$

To prove this theorem, we first need to understand some definitions and theorems about the column and null space of a matrix. For configuration $\vec{b}$ to be winnable, it must have a strategy $\vec{x}$ that satisfies $A\vec{x} = \vec{b}$, meaning that configuration $\vec{b}$ is winnable if and only if it belongs to the column space of A, denoted as Col (A).

Since A is symmetrical, the row space of A, denoted as Row (A), is equal to Col (A). From the definitions of the row space and the null space, we know that Row (A) is equal to the orthogonal complements of the null space of A, denoted Null (A). Since the null space of a matrix is not affected by the Gauss-Jordan elementary row operations, let matrix E be the reduced row echelon form of matrix A. We can then see that Null (A) = Null (E).

It then follows that $\vec{b}$ is winnable if and only if it belongs to the orthogonal complements of Null (E). Since we are looking for the orthogonal complements of Null (E), we need to find an orthogonal basis for the Null (E). To find a basis for the Null (E), we will interpret matrix E as a system of equations and then solve for the dependent variables. We can see in the equations

$$\begin{array}{ccc} & & \\ 0 & = & e_1 + e_{25} \\ 0 & = & e_2 + e_{24} \\ 0 & = & e_3 + e_{24} + e_{25} \\ \vdots & = & \vdots \\ 0 & = & e_{23} + e_{24} + e_{25} \\ & & \end{array} \Rightarrow \begin{array}{ccc} e_1 & = & e_{25} \\ e_2 & = & e_{24} \\ e_3 & = & e_{24} + e_{25} \\ \vdots & = & \vdots \\ e_{23} & = & e_{24} + e_{25} \\ e_{24} & = & e_{24} \\ e_{25} & = & e_{25} \end{array}, \tag{8}$$

that by doing the calculations in modulo 2, that the negatives become positive. By writing the independent variables in terms of the dependent variables we get

$$\begin{array}{lcl} e_1 & = & \\ e_2 & = & \\ e_3 & = & \\ \vdots & = e_{24} & \\ e_{23} & = & \\ e_{24} & = & \\ e_{25} & = & \end{array} \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ 1 \\ 1 \\ 0 \end{bmatrix} + e_{25} \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{9}$$

The resulting collection of vectors will be the basis for the Null (E). So a basis for the Null (E) is $\vec{n}_1$ and $\vec{n}_2$, the same as (6) and (7).

To check to see if these two vectors form an orthogonal basis, we must simply show the dot product of the two vectors is equal to zero. We find out that $\vec{n}_1 \cdot \vec{n}_2 = 8$, but since we are doing our calculations in modulo 2, we see that $\vec{n}_1 \cdot \vec{n}_2 = 0$, and that $\vec{n}_1$ and $\vec{n}_2$ are an orthogonal basis for the Null (E).

Since $\vec{b}$ is winnable if and only if it belongs to the orthogonal complements of Null (E), then $\vec{b}$ is winnable if and only if it is orthogonal to the two vectors $\vec{n}_1$ and $\vec{n}_2$, thus proving Theorem 1.

B. *Theorem 2*

Theorem 2 states that if $\vec{b}$ is winnable, then the four winning strategies are

$$\begin{array}{rc} \vec{x} = & R\vec{b} \\ & R\vec{b} + n_1 \\ & R\vec{b} + n_2 \\ & R\vec{b} + n_1 + n_2 \end{array} \tag{10}$$

where $\vec{n}_1$ and $\vec{n}_2$ are the same as (6) and (7) and *R* is the product of the elementary matrices which perform the reducing row operation, RA=E .[1]

Since *E* has two independent variables, x_{24} and x_{25}, then the four configurations we can choose for them are

$$\begin{array}{c|c} x_{24} & x_{25} \\ \hline 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{array} \tag{11}$$

Let's choose $x_{24} = 0$ and $x_{25} = 0$. We can then see the product $E\,\vec{x} = \vec{x}$. We can then substitute in RA=E and $A\vec{x} = \vec{b}$ to get

$$\vec{x} = E\ \vec{x} = R\ A\,\vec{x} = R\,\vec{b}\ . \tag{13}$$

Therefore, one solution is $\vec{x} = R\,\vec{b}$. By looking at the other configurations for x_{24} and x_{25}, we can let x_{24} and x_{25} be as shown in (11). As a result, we can see that the four winning strategies are the four in (10).

C. *Graphical User Interface*

Using Theorems 1 and 2, we now have a way to mathematically check to see if a starting configuration is winnable. If it is winnable, we now have four equations that will find for a strategy that will turn all the lights out. To find the strategy with the fewest number of buttons pressed, you simply need to count the number of buttons that need to be pressed for each of the four strategies.

Since we are dealing with large matrices, 25 by 25, doing this computation by hand would be long and tedious. With the help of a computer algebra system or programming language capable of handling matrices like *Maple*, *Mathematica*, or *R*, one could compute the winning strategies. Due to my current knowledge, I did my computation in *R* and created a graphical user interface where one could enter a starting configuration and the GUI would return "No Solution" if there was no winning strategy or it would return the winning strategy with the fewest number of moves.

IV. Solving Using Light Chasing

A way that one could solve Lights Out without using a computer algebra system was first introduced to me by one of my colleagues [9]. Upon listening to him explain this method to me, which he called the "Binary Method," I did a little research and found that the method he was explaining was the Light Chasing Method.

The idea of the Light Chasing Method is to separate the 5 by 5 array into a 1 by 5 and a 4 by 5 array and then work on turning all the lights out in the 1 by 5 array. Then you would continue to separate the remaining array until you got down to a 1 by 5 array. An easy way to think about this is by turning out all the lights on the top row. One achieves this by simply pressing the buttons on the second row that are directly underneath a lit buttons on the top row. The top row will then have all its lights off. By repeating this step for the second, third, and fourth rows (i.e. chase the lights all the way down to the bottom row), you may have solved the puzzle, but it is more likely that there will now be some lights left on in the bottom row. If so, there are only seven possible configurations. Depending on which configuration you are left with, you will need to press some buttons in the top row. You can determine which buttons you need to press from the Figure 3.

The problem with the Light Chasing Method is that you will most likely end up pressing some buttons more that once, which means you won't solve the game in the fewest number of buttons pressed and will end up replaying the board. A simple solution to this is once you get down to your bottom row, restart the board and start by pressing the corresponding buttons in the top row to clear the board.

If you don't have the Figure 3 with you, my colleague that introduced me to this method explained that one should think of the five buttons in the top row as a five digit binary number staring at zero and increasing by one each time. If you press the buttons in the top row and it doesn't clear the board then continue on to the next binary number. For example, first try 00000, 10000, 01000, and 00100 where a one represents pressing the corresponding button. If you continue this, you will find that there are only 32 different ways to press the buttons in the top row. We proved earlier in this paper that there are 4 winnable solutions, so an eighth of the possible ways to press the buttons in the top row will clear the board. Using a geometric distribution and the "Binary Method", you

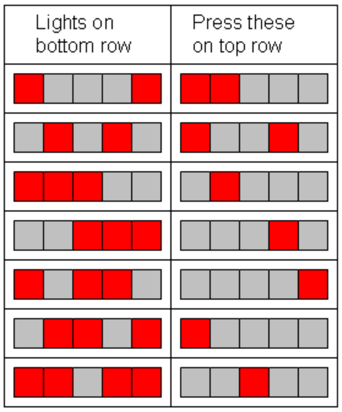

Figure 3. After chasing the lights down to the bottom row, you will have turned the lights out or have one of seven remaining configuration as shown in the left column. By pressing the buttons in the top row that are in the right column of the row of the remaining configuration and repeat the Chasing Lights Method you will turn all the lights out.

have a 0.5512 probability of clearing the board within the first six attempts, with a mean of eight attempts.

The other problem with solving lights out using Light Chasing is there is no way of checking to see if a board is winnable unless you go through all 32 different combinations.

V. Changes to the Game

So far this paper has just talked about the original 5 by 5 Lights Out, but since its release in 1995, Tiger Toys have created several other variations to the game.

The game Lights Out Cube consists of a 3 by 3 by 3 cube of lights and is played just like ordinary lights out. When you press a button at the edge of a 3x3 square, the affected lights wrap around onto the adjacent faces.

The Lights Out Keychain, also referred to as Lights Out Mini, is a 4 by 4 array of lights that plays on a torus. Playing on a 4 by 4 torus there is a unique board because every possible starting configuration is solvable and each solution is unique.

Lights Out Deluxe is a version of Lights Out played on a 6 by 6 array and offers some different game variations. These variations are slight changes in the rules. One variation is called Lit Only where you are only allowed to press buttons that are currently on.

Tiger Toys' latest Lights Out is called Lights Out 2000. This game plays 3-state Lights Out. The lights cycle through red, green, and off. This creates a significantly different and challenging game.

There are also hundreds of different variations that could be made to the game. First of all, the board could take on almost any shape it wants. Whether it is an *n* by *n* board, a circle, or a randomly scattered board, as long as you know which buttons toggle when you press a given button, you can play Lights Out.

VI. Conclusion

This paper has shown a variety of different methods that can be used to solve the electronic game Lights Out. One could use a computer algebra system capable of handling matrices where you know you can get an answer right away or use a guessing method like the "Binary Method" where you keep trying different sequences until onefinds a solution.

Having read this paper, I hope that I have inspired you to search the internet for an online version of Lights Out so you can practice the methods that I have talked about or a method of your own. If reading this paper has made you ambitious enough, go buy a microcontroller and try to recreate the game Lights Out.

Acknowledgment

I would like to acknowledge and thank the others members of my group, Jon Palecek and Prasoon Saurabh, for their input and help on my research. I would also like to thank our faculty advisor Professor Timothy Hess for his input and help with programming. I would also like to thank Ross Lewis for a great conversation about the game and introducing me to a different method for solving the game. Additionally, I would like to acknowledge Professor David Scott for helping me with various terms and calculations.

References

[1] Turning Lights Out with Linear Algebra, by Marlow Anderson and Todd Feil Mathematics Magazine, Vol. 71, No. 4 (Oct., 1998), pp. 300-303 Mathematical Association of America.
[2] An Easy Solution to Mini Lights Out, by Jennie Missigman and Richard Weida *Mathematics Magazine* © 2001 Mathematical Association of America.
[3] Two Reflected Analyses of Lights Out, by Óscar Martín-Sánchez and Cristóbal Pareja-Flores *Mathematics Magazine* © 2001 Mathematical Association of America.
[4] C. Haese, paper in the sci.nath newsgroup in 1998, available at http://www.math.niu.edu/-rusin/known- math/98/lights-out
[5] Hohn, Franz E. *Introduction to Linear Algebra*. New York: The Macmillian Company, 1972. 161-188. Print.
[6] K. Banr, Lights Out Fan Club, at http://gbs.mit.edu/-kbarr/lo/
[7] D.L. Stock, Merlin's magic square revisited, Amer Math. Monthly 96 (1989), 608-610.
[8] java version of the commercially marketed game *Lights Out* at http://www.whitman.edu/mathematics/lights_out/
[9] Barile, Margherita. "Lights Out Puzzle." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein. http://mathworld.wolfram.com/LightsOutPuzzle.html
[10] Lewis, Ross. Conversation with about alterative solution

Matthew A. Madsen was born in Reedsburg, Wisconsin on November 7, 1988. He is currently pursuing his Baccalaureate degree in Mathematics and Physics from Ripon College in Ripon, Wisconsin and will be graduating in May 2011 after completing his final semester.

He is currently employed by the Mathematics and Computer Science Department and the Physics Department as a Departmental Assistant while working part-time at Webster's Pick 'N Save in Ripon, Wisconsin as customer service and terminal operator. He has spent several years working as a lifeguard at the Wilderness Resort in Wisconsin Dells.

Mr. Madsen is a member of the Society of Physics Students. In his free time he enjoys playing video games, hunting, watching movies and sports, and spending time with family and friends.