An Adaptive Educational Game To Help Students Learn How To Solve Systems Of Linear Equations

Wang Zhang

AN ADAPTIVE EDUCATIONAL GAME TO HELP STUDENTS LEARN HOW TO SOLVE SYSTEMS OF LINEAR EQUATIONS

Wang Zhang

AN ADAPTIVE EDUCATIONAL GAME TO HELP STUDENTS LEARN HOW TO SOLVE SYSTEMS OF LINEAR EQUATIONS

Wang Zhang

Columbus State University

The College of Business and Computer Science

The Graduate Program in Applied Computer Science

An Adaptive Educational Game to Help Students Learn How to Solve Systems of Linear Equations

A Thesis in

Applied Computer Science

by

Wang Zhang

Submitted in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

August 2014

I have submitted this thesis in partial fulfillment of the requirements for the degree of Master of Science

8/29/2014 — Date

Wang Zhang — Wang Zhang

We approve the thesis of Wang Zhang as presented here.

8/29/2014 — Date

[illegible] — Rania Hodhod, Ph.D.
Assistant Professor of Computer Science,
Thesis Advisor

8/29/2014 — Date

[illegible] — Shamim Khan, Ph.D.
Professor of Computer Science

8/29/2014 — Date

[illegible] — Rodrigo Obando, Ph.D.
Associate Professor of Computer Science

Abstract

Educational games have been proven to be effective in developing problem solving skills in well-defined domain, such as Math and Physics. In this thesis, an educational game called MatriX was developed to foster problem solving skills in the domain of linear algebra, particularly solving a system of linear equations. MatriX is an adaptive educational game that uses intelligent tutoring modules to guide the student's learning process and provide feedback based on the student's performance. These modules are domain module, student module, pedagogical module and presentation module. The domain module contains all the concepts the student needs to learn and an automated solver for linear equations that adopts the rules of Gaussian Elimination. The student module records the student's performance and provides the pedagogical module with the required information about the student's current skills. The pedagogical module uses the automated solver to assess the student's performance on the designated task and a neuro-fuzzy system to decide on the next proper game level for the student. MatriX has been evaluated by 13 students from the Columbus State University. The results show that MatriX was well perceived by the students and that they were able to transfer the skills learned in the game to real world problems on systems of linear equations.

Acknowledgements

I would like to thank everyone who has helped and supported me with this thesis. First of all, I would like to thank my supervisor, Prof. Rania Hodhod, for her professional and patient guidance. This thesis would not be, without her help. Particularly, I would like to thank all the students who participated in the evaluation study. And thank all the faculty and staff in the TSYS School of Computer Science, especially Dr. Wayne Summers, Dr. Shamim Khan, Dr. Rodrigo Obando and Ms. Dianne Phillips. Finally, I would like to acknowledge, I am so thankful and grateful to my family members, especially my parents, for supporting me all the time throughout my study in Columbus State University.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Introduction

Problem solving skills helps us throughout our lives; it is something we use every day. We encounter problems, no matter how big or small, which require problem solving skills to deal with them (Steve K. Robbins, 2013). In fact mathematics can be seen as a useful tool to solve daily problems. We can convert our daily problems into math problems and also convert math problems into what we see in daily life, such as puzzles and games (Bonnie Averbach, 2012).

No matter whether it is a math problem or other kind of problem, we improve our problem solving skills through education. Education is a form of propagation and learning, through which, knowledge, skills, and life experience are learned or transferred. With today's technology we can make the process of learning more fun. Problem solving skill is as any other skill that requires practice in order to be developed and used efficiently. Educational games provide a good platform to improve problem solving skills.

For the purpose of this thesis an adaptive educational game called MatriX was developed. MatriX aims to help students to improve their skills using the rules of the elementary matrix operations to solve systems of linear equations. MatriX consists of 4 modules – a domain module, a student module, a pedagogical module and a presentation module. The domain module contains the rules and the problem-solving strategies that the students need to learn. The student module tracks students solving steps and reflects the students' skills. The pedagogical module provides smart hints that tell the student what the next step is when the student does not advance and also determines the next level the student should go to after they finish a level. The presentation module provides the GUI for the students. MatriX was implemented using Microsoft XNA Game Studio 4.0 and programmed in the C# language.

1.2 Game and Education

People have learned from games since ancient times. When we play a game, we must learn and adapt to the rules of the game in order to get to the challenging goal, which is the process by which we learn new knowledge and skills. Before the invention of computers, we played card games to learn Arithmetic and played chess to learn strategies. After the invention of computer games, the variety of educational games became significantly rich. We can find a list of hundreds or thousands of new educational games in every decade. The tremendous number of games makes some people fear that games are bad educators, which have actually been proven false by successful educational games.

Intelligent educational games allow a personalized learning experience to each individual student. Those games can provide instant and individualized feedback by using Artificial Intelligence techniques that allow the game to reason, plan, and adapt to each student (Rania Hodhod, 2010).

1.3 Domain of Interest

When I was learning linear algebra I spent a lot of time practicing the Gaussian elimination, which is the most widely used method to solve systems of linear equations, which is one of the most basic and most used methods in linear algebra and is also used to find the rank, the determinant and inverse matrix of a matrix. Accordingly Gaussian

Elimination is considered an important technique in linear algebra.

Gaussian Elimination uses three types of elementary matrices row operations: Row switching, row multiplication and row addition. Row switching is exchanging a row in the matrix with another one, row multiplication is multiplying each element in a row in the matrix by a non-zero constant and row addition is to replace all elements in a row with the sum of this row with another row in the matrix. Gaussian Elimination uses these operations to transform the coefficient matrix to be a triangular matrix while the constant vector is applied by those transformations as well (Robert J. Lopez, 2010). MatriX aims to develop problem solving skills in students by providing tasks that help them exercise the elementary matrix row operations which are the core of the Gaussian Elimination technique.

Chapter 2 Related Work

2.1 Problem Solving Skills

Jennifer Krawec et al. (2012) published a paper that addresses the effects of cognitive strategies titled "Instruction on Knowledge of Math Problem-Solving Process of Middle School Students with Learning Disabilities". This study investigated the effectiveness of a cognitive strategy intervention called Solve It! on students' knowledge of math problem-solving strategies. Solve It! was designed to improve the math problem solving skills of middle school students who had learning disabilities. This study followed the Mayer's (1985) model of the problem-solving process that identifies four sequential phases: Problem translation, problem integration, solution planning and solution execution. The researchers collected data over the course of two years with two separate samples from 7th and 8th grade students. They composed a Math Problem-Solving Assessment to measure the students' skills that contains a structured interview consisting of two word problems and thirty four items selected from a longer version developed for research purpose. The results indicated that the students who got trained on Solve it! were able to use more strategies to solve mathematical word problems than those students who didn't.

Robert W. Maloy et al. (2010) published an article that describes the study of a web-based mathematics tutoring systems, called 4MALITY, with one hundred and twenty five fourth grade students and their teachers. The 4Mality is an online tutoring system that was used to promote inquiry learning and problem solving among elementary and middle school students. This system uses a hint model to organize suggestions and strategies along two axes – problem solving steps and learning style preferences. The researchers introduced five steps in the problem-solving axis, which was originally drawn from the work of George Polya (1973): Hint Level 1) What kind of question is this?; Hint Level 2) What is the question asking for?; Hint Level 3) What do I already know that will help solve the problem?; Hint Level 4) What is my plan for solving the problem?; And Hint Level 5) How do I know I have solved the problem? They found a mean gain of 25.51% in test scores from pre-test to post-test among all students.

2.2 Serious Games

Damien Djaouti et al. (2011) stated, in their article titled Origins of Serious Games, that “Serious Game” was not a new phenomenon. They believe that the very first video games were not designed purely for entertainment. The first serious games were not necessarily based on a digital support. While there were many games not labelled as serious games, they are the closest ancestor to today’s serious games. In this work the researchers compared the numbers of serious games released each year, and found out that the first high peak occurred in 2000’s and 2002 was the starting point of the current wave of serious games. The researchers pointed out that the number of serious games released in 2007 was 230, which was twice the number in 2003 that was the highest number before 2007. They stated that before 2002 education had the highest percentage, 65.8% partition of serious games; however this number decreased to 25.7% after 2002 and advertising games reached the top with 30.6% of the games.

Irene Polycarpou et al. (2010) developed an educational game called Math-City, which was a simulation-based game for K-12 students to improve their achievement in Mathematics. In the game, students can create and maintain their own city. They can add their own residential, commercial and industrial buildings in the game. The goal of the game is to create a city that has the maximum happiness of the residents, which includes five factors – pollution, police, fire, health and big building. In the game, the students need to earn their money to develop the city by answering mathematical problems. The teachers who participated in the game filled out a survey, and gave mostly positive feedback.

2.3 Fuzzy Systems

Regina Stathacopoulou (2006) did a study in her thesis about a neural network-based fuzzy modeling approach to assess student's learning characteristics and update the student module in Intelligent Learning Environments. She designed a three-stage diagnostic model, in which the first state is the fuzzification state, the second is the inference stage and the third is the defuzzification state. The fuzzification stage represents teachers' subjective linguistic description of a students' behavior. The inference state represents teachers' reasoning in categorizing students qualitatively according to their learning characteristics. And the deffuzification state represents teachers' final decision in classifying a student in one of the predefined linguistic values of the characteristic. This system has 3 fuzzy inputs and they are the student's total time on the scenario, the number of attempts to find the correct forces and the number of random mouse moves. And the output is how much the student is interested in the scenario. The experimental test result shows that the proposed model accurately evaluated students.

Shahriar Husainy (2013) did a study concerning the development of a Fuzzy Inference System for identifying likely student dropouts at Columbus State University. This system was developed and evaluated by utilizing historical students' Retention, Progression and Graduation (PRG) data from Columbus State University Information and Technology

Services. He used a top-down and a bottom-up approaches to perform the knowledge extraction for the system. The top-down approach was used to extract data from the knowledge gained from interviewing domain experts for forming the rules. And the bottom-up approach was used to analyze the weights of an ANN and derive additional rules for the system.

Chapter 3 Design of MatriX

3.1 Overview

This chapter focuses on the educational game MatriX developed for the purpose of this thesis. The rules of the game are simply the elementary matrix operations, which are row addition, row multiplication and row swapping. The goal of the game is to use those operations to get an identity matrix. MatriX is an intelligent educational game in the sense that it can track the student's performance and provide individualized feedback. MatriX uses four modules to achieve this: a domain module, a student module, a pedagogical module and a presentation module. The domain module is the main component of the game and is where the game rules are applied. The student module records the student's performance and helps providing a personalized learning process. In this project, the student module captures the students' actions, cognitive processes and provides that information to the pedagogical module. The pedagogical module uses a neuro-fuzzy system to provide adaptation to individual students playing the game. Adaptation allows the presentation of a sequence of game levels that fits the student's skills, i.e. the student doesn't need to play all the game levels in the game. In addition the pedagogical module provides the students with smart hints that help the student to proceed with his learning activity when he gets stuck.

3.2 Game Design

MatriX is a puzzle game, which applies the rules from the Gaussian Elimination technique, performing a sequence of elementary operations on the associated matrix of coefficients. For example we can derive the system of linear equation like

$\begin{cases} x + 3y = 8 \\ s + 4y + 2z = 3 \\ x + 2y + 2z = 5 \end{cases}$ to be like $\begin{pmatrix} 1 & 3 & 0 \\ 1 & 4 & 2 \\ 1 & 2 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 8 \\ 3 \\ 5 \end{pmatrix}$, and then we associate the coefficient matrix with the constant vector and get an augmented matrix like $\left(\begin{array}{ccc|c} 1 & 3 & 0 & 8 \\ 1 & 4 & 2 & 3 \\ 1 & 2 & 2 & 5 \end{array}\right)$. This augmented matrix can be easily represented in a game where the students can practice the different operations of the Gaussian Elimination technique as shown in figure 1.

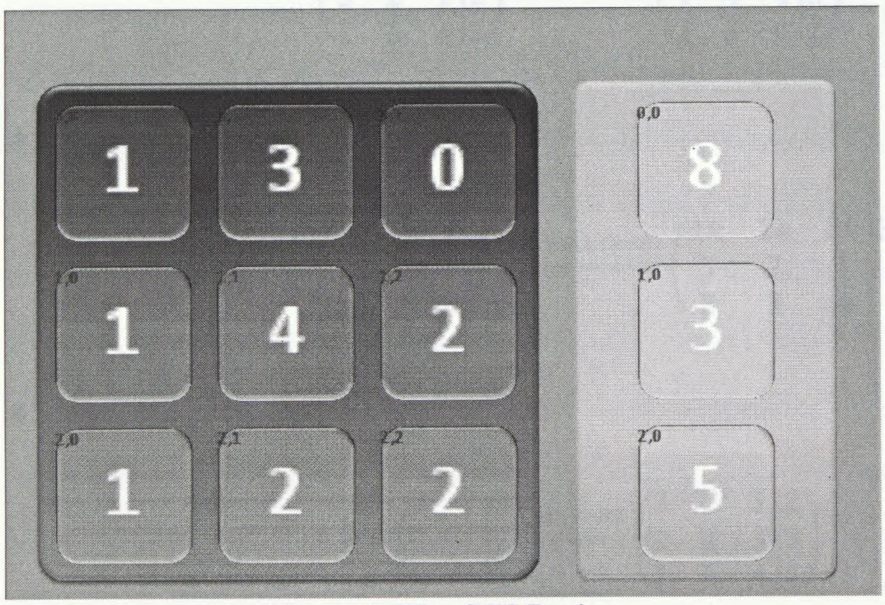


Figure 1. The GUI Design

The players just need to drag and drop or click on the rows to perform the different operations to reach the identity matrix.

3.3 Domain Module

3.3.1 Concepts

The main concept in the domain module is row reduction in a matrix using elementary operations. The domain contains all the problems the student needs to solve and all the concepts the student needs to learn about. There are three main types of operations the student needs to understand – row addition, row multiplication and row swapping. The row addition is to add one row onto another, the row multiplication is to multiply a row by a non-zero constant, and the row switching is to swap two rows, for example:

- Row addition

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 3 \\ 3 & 2 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{array}\right) \xrightarrow{\text{R1 add to R2}} \left(\begin{array}{ccc|c} 1 & 2 & 3 & 3 \\ 4 & 4 & 4 & 5 \\ 2 & 3 & 4 & 1 \end{array}\right)$$

- Row Multiplication

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 3 \\ 3 & 2 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{array}\right) \xrightarrow{\text{k* R1} \rightarrow \text{R1 (k}\neq\text{0)}} \left(\begin{array}{ccc|c} 1k & 2k & 3k & 3k \\ 3 & 2 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{array}\right)$$

- Row Switching

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 3 \\ 2 & 2 & 1 & 2 \\ 2 & 3 & 4 & 1 \end{array}\right) \xrightarrow{\text{R1} \leftrightarrow \text{R2}} \left(\begin{array}{ccc|c} 3 & 2 & 1 & 2 \\ 1 & 2 & 3 & 3 \\ 2 & 3 & 4 & 1 \end{array}\right)$$

The goal of the Gaussian elimination is to transform a matrix to be in an echelon form, which means that all entries below the main diagonal entries are 0. For example:

$$\begin{pmatrix} d & a & b \\ 0 & e & c \\ 0 & 0 & f \end{pmatrix}$$

However, in this game the goal is to transform the coefficient matrix to be an identity matrix so that the student can directly see the roots of the system of linear equations by looking at the constant vector at the right side of the augmented matrix. For example, the form shown below is a solved matrix, through which we can see the roots are 11, -1 and -2.

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 11 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{array}\right)$$

3.3.2 Elimination with matrices

An automated solver has been implemented to complement the domain module. This solver can provide the solution to any solvable 2d, 3d or 4d system of linear equation using matrix operations and return the list of steps required to solve this particular problem. The solver uses an algorithm derived from Gaussian Elimination. The solver doesn't come up with the best solution, as this is an NP problem, which might use even 1 hour to solve a 3 by 3 matrix, not to mention the 4 by 4 matrix problems. However, the solver aims to provide the next step from any current step, which proved to be ab excellent way to avoid storing solutions to each problem in the game and, most importantly provide high flexibility as the student does not have to follow certain sequence of rules to solve the problem in hand.

For simplification purposes, MatriX does not allow the use of fractions; therefore all numbers in the solution must be integers. For this reason the solver cannot directly use the algorithm of the Gaussian Elimination, which uses a lot of fractions. Also note that the base case of the adapted algorithm is the coefficient matrix reduced to an identity matrix rather than a reduced row echelon form.

3.3.3 Level Difficulty and Level Design

Problems with different difficulty levels were designed for the purpose of this game. In the meantime, MatriX is not set to generate new problems but this is easy to be incorporated in MatriX. All what it needs to achieve this is to allow the automated solver to use down-top problem solving approach instead of using the top-down approach. This part is left as future work as it was not required to fulfill the requirements of this thesis.

There are different ways that can be approached in order to design game levels (problems) with different difficulties. One way is to randomly choose a couple of numbers as the coefficient of the systems of linear equations. However, using this approach makes it hard to manage the level difficulties and requires us to check the rank of each matrix to make sure it is solvable. Another approach, which is adopted in this work, applies elementary matrix operations on the identity matrix to generate a new problem. This way we could easily control the difficulty level by controlling the number of steps of elementary operation on the matrix to generate the new problem. This is a process similar to mixing up a Rubik Cube; the more times we turn any side of the cube, the more difficult it would be to solve it.

For example, assume we have the two transformations below,

$$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\rightarrow\begin{pmatrix}1&0&0\\1&1&0\\0&0&1\end{pmatrix}$$ Transformation 1

$$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}\rightarrow\begin{pmatrix}1&0&0\\1&1&0\\0&0&1\end{pmatrix}\rightarrow\begin{pmatrix}1&0&0\\1&1&0\\1&1&1\end{pmatrix}$$ Transformation 2

It can be seen that the result of Transformation 1 is easier than the result of Transformation 2, because Transformation 2 uses more steps than Transformation 1. The dimension of the matrix is also a great factor to define the difficulty level. Solving a 3-D matrix seems to be more difficult than solving a 2-D one, even if both problems require the same number of steps in the solution.

The pedagogical module is responsible for providing adaptation that helps the student to experience an appropriate learning curve. A reasonable difficulty curve can keep the learning experience balanced to the student. For that purpose, we designed 12 easy levels, 12 medium levels and 8 hard levels and used numbers from 0 to 100 to represent the difficulty level (this will be converted to 0~1 to be used as a membership function in the student module). The following table demonstrates the distribution of the difficulty level of the problems.

Table 1. Level Difficulty

Level No	Difficulty Degree	Difficulty
0~11	$(0, 25]$	Easy
12~23	$(25, 75]$	Medium
24~31	$(75, 100]$	Hard

The following formula is used to determine the difficulty level of each problem,

$$\left(\sqrt[n]{m - m_0}\right)^{l-l_0} + m_0$$

In this formula, m_0 represents the lower bound of the difficulty level of the current problem; m represents the upper bound of the difficulty level of the current problem; n represents the number of levels in the current level difficulty; l represents the current level number and l_0 represents the initial level number of the current difficulty level.

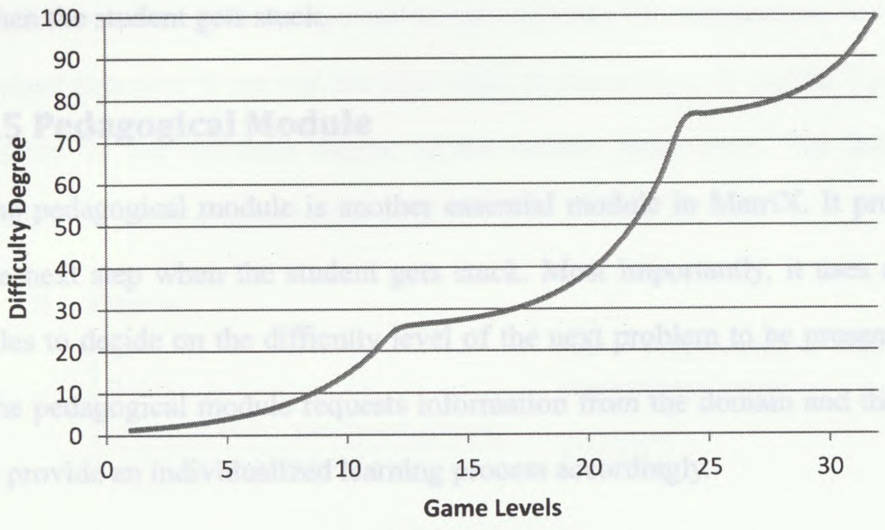


Figure 2. The Difficulty Curve

Figure 2 shows the difficulty curve obtained from the above formula. The x-axis represents the level number of each level, and the y-axis represents the level difficulty degree. As we can see every time we enter a new level difficulty, such as from easy to medium or from medium to hard, the slope is increasing gently which allows the student to fit into the new difficulty level smoothly when advancing from one level to another.

3.4 Student Module

The student module is a crucial part of MatriX. The student module records the performance and learning ability of the student, which will be used by the pedagogical module in MatirX. Accordingly, the pedagogical module would know what steps did the student take to solve the problems, how much time the student used to solve the problems and the current difficulty level the student is on, so that the pedagogical module can automatically choose the next appropriate level or problem that fits the student most. Moreover, the pedagogical module can give the right level of smart hints to the student when the student gets stuck.

3.5 Pedagogical Module

The pedagogical module is another essential module in MatriX. It provides hints about the next step when the student gets stuck. Most importantly, it uses a fuzzy system of rules to decide on the difficulty level of the next problem to be presented to the student. The pedagogical module requests information from the domain and the student modules to provide an individualized learning process accordingly.

3.5.1 Fuzzy System Design

In order to select the right difficulty level for the student, a system of fuzzy rules was developed that evaluates the student's performance. The information collected from the student module represents the different premises of the rules and the output represents the difficulty level of the next problem. The rules were designed to be used in a neuro-fuzzy system that has functions make it be able to learn from the student. Unfortunately the learning functions were turned off in this work because of the abundance of training data. The following section describes the neuro-fuzzy system in detail.

- **Input/Outputs**

The neuro-fuzzy system has 3 input layers and 1 output layer. The inputs include: 1- Step: the value of Step is defined by the difference between the number of steps taken by the automated solver to solve the problem and the number of steps taken by the student to solve the same problem divided by the number of steps taken by the automated solver to solve the problem. If we have a problem with expected step count to be 3 and the student used 4 steps then the value of Step would be 1/3. 2- Time: The value of time is defined to be the time the student expected to use dividing the comparison between the time the student expected to use and the time taken by the student. 3- Level: The value of Level is defined by the difficulty degree of the current game level. The following shows the formulas used to compute each input

- $Step =$

$$\begin{cases} 0 & (userStepCount < expectedStepCount) \\ 1 & (userStepCount > 2 * expectedStepCount) \\ \frac{userStepCount - expectedStepCount}{expectedStepCount} & (Other) \end{cases}$$

- $$Time = \begin{cases} 0 & (userTime < expectedTime) \\ 1 & (userTime > 2 * expectedTime) \\ \frac{userTime - expectedTime}{expectedTime} & (Other) \end{cases}$$

- Level = difficulty degree of the current game level

The output of the neuro-fuzzy system is the difficulty degree of the next problem. All the values mentioned above are clamped in the range from 0 to 1. For example, if the student used three times of the expected time, the value of Time is one. This can be attributed to the fact that the expected performance varies from one problem to another.

- **Membership Functions**

Each of the input variables along with the output variable is represented with a membership function as seen in Figure 3, 4, 5 and 6. The Step variable has three fuzzy values: fast, medium and slow. A triangular and trapezoidal membership functions are used to represent those values as seen in Figure 3.

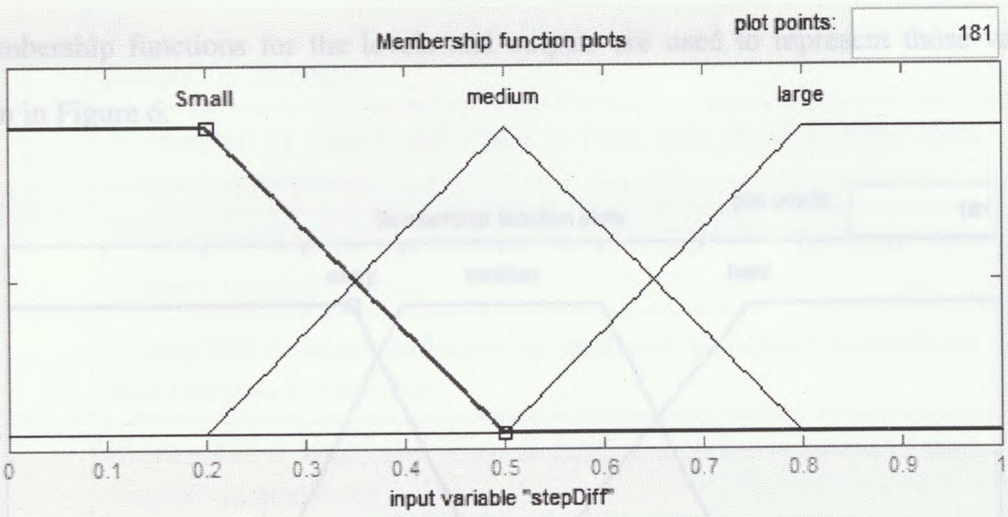


Figure 3. Step Membership Functions

The Time variable has three fuzzy values: fast, medium and slow. A triangular and trapezoidal membership functions are used to represent those values as seen in Figure 4.

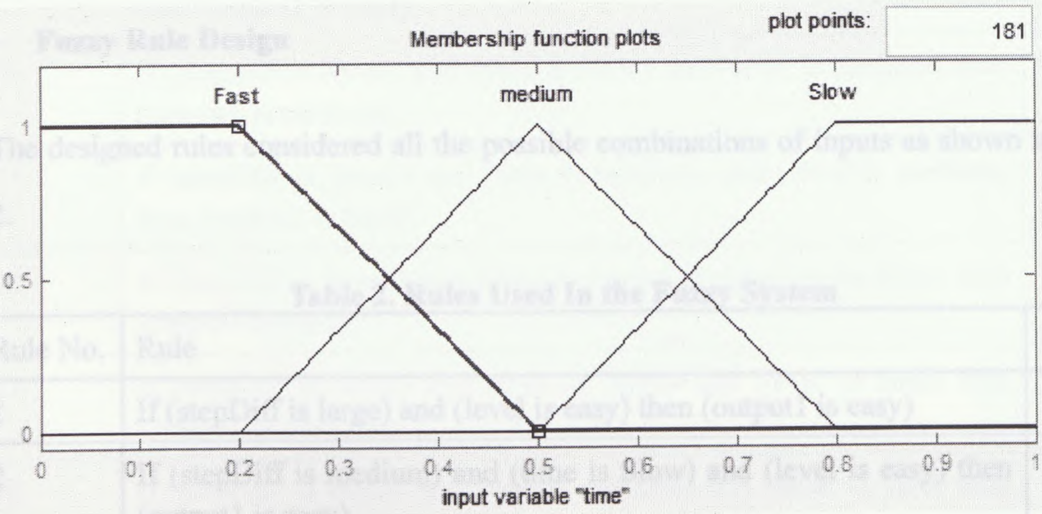


Figure 4. Time Membership Functions

For the level there are also 3 membership functions, easy, medium and hard. Their ranges are as in the following figure. These membership functions are used as input membership function for level input and as output membership function for level output as well. The membership functions for the levels and outputs are used to represent those values as seen in Figure 6.

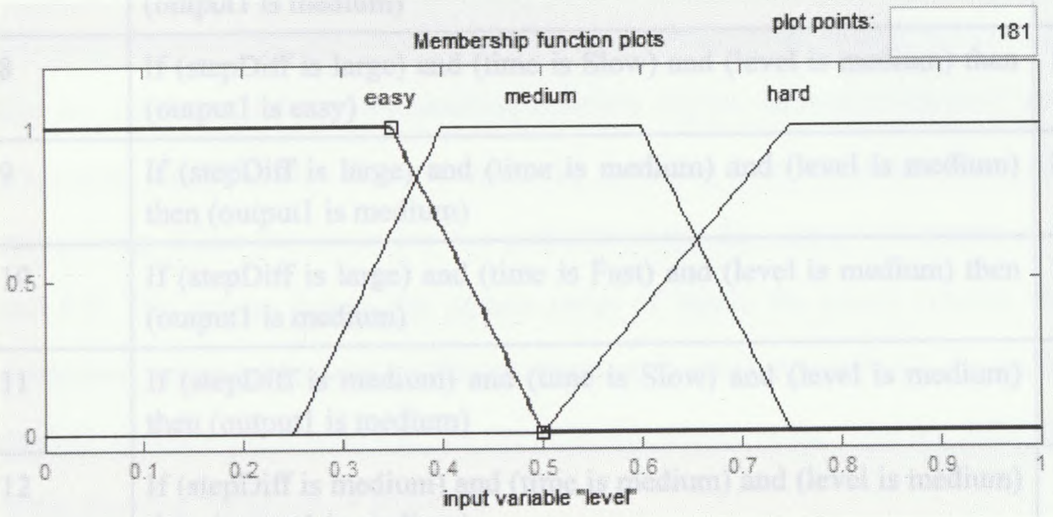


Figure 5. Level and Output Membership Functions

- **Fuzzy Rule Design**

The designed rules considered all the possible combinations of inputs as shown in Table 2.

Table 2. Rules Used In the Fuzzy System

Rule No.	Rule	Weight
1	If (stepDiff is large) and (level is easy) then (output1 is easy)	1
2	If (stepDiff is medium) and (time is Slow) and (level is easy) then (output1 is easy)	1
3	If (stepDiff is medium) and (time is medium) and (level is easy) then (output1 is easy)	1
4	If (stepDiff is medium) and (time is Fast) and (level is easy) then (output1 is medium)	1
5	If (stepDiff is Small) and (time is Slow) and (level is easy) then (output1 is easy)	1

6	If (stepDiff is Small) and (time is medium) and (level is easy) then (output1 is medium)	1
7	If (stepDiff is Small) and (time is Fast) and (level is easy) then (output1 is medium)	1
8	If (stepDiff is large) and (time is Slow) and (level is medium) then (output1 is easy)	1
9	If (stepDiff is large) and (time is medium) and (level is medium) then (output1 is medium)	1
10	If (stepDiff is large) and (time is Fast) and (level is medium) then (output1 is medium)	1
11	If (stepDiff is medium) and (time is Slow) and (level is medium) then (output1 is medium)	1
12	If (stepDiff is medium) and (time is medium) and (level is medium) then (output1 is medium)	1
13	If (stepDiff is medium) and (time is Fast) and (level is medium) then (output1 is hard)	1
14	If (stepDiff is Small) and (time is Slow) and (level is medium) then (output1 is medium)	1
15	If (stepDiff is Small) and (time is medium) and (level is medium) then (output1 is hard)	1
16	If (stepDiff is Small) and (time is Fast) and (level is medium) then (output1 is hard)	1
17	If (stepDiff is large) and (time is Slow) and (level is hard) then (output1 is medium)	1
18	If (stepDiff is large) and (time is medium) and (level is hard) then (output1 is hard)	1
19	If (stepDiff is large) and (time is Fast) and (level is hard) then (output1 is hard)	1
20	If (stepDiff is medium) and (level is hard) then (output1 is hard)	1
21	If (stepDiff is Small) and (level is hard) then (output1 is hard)	1

- **Level Selection**

The fuzzy system uses the 3 inputs (Step, Time, and Level) to figure out the appropriate difficulty degree of the next problem. And then the pedagogical module selects a problem that has the closest value to the resulting difficulty degree. As seen in Figure 6, the z-axis represents the difficulty of the output, the x and y axes represent Time and Step respectively. When the inputs are in a particular range the output will remain at 0.1, 0.5 and 0.8. This means that in this certain range of inputs the output remains the same (undesired output as we need a different output each time). Accordingly, the pedagogical module selects a game level that has the difficulty degree in the resulting difficulty level while allowing variations of problems.

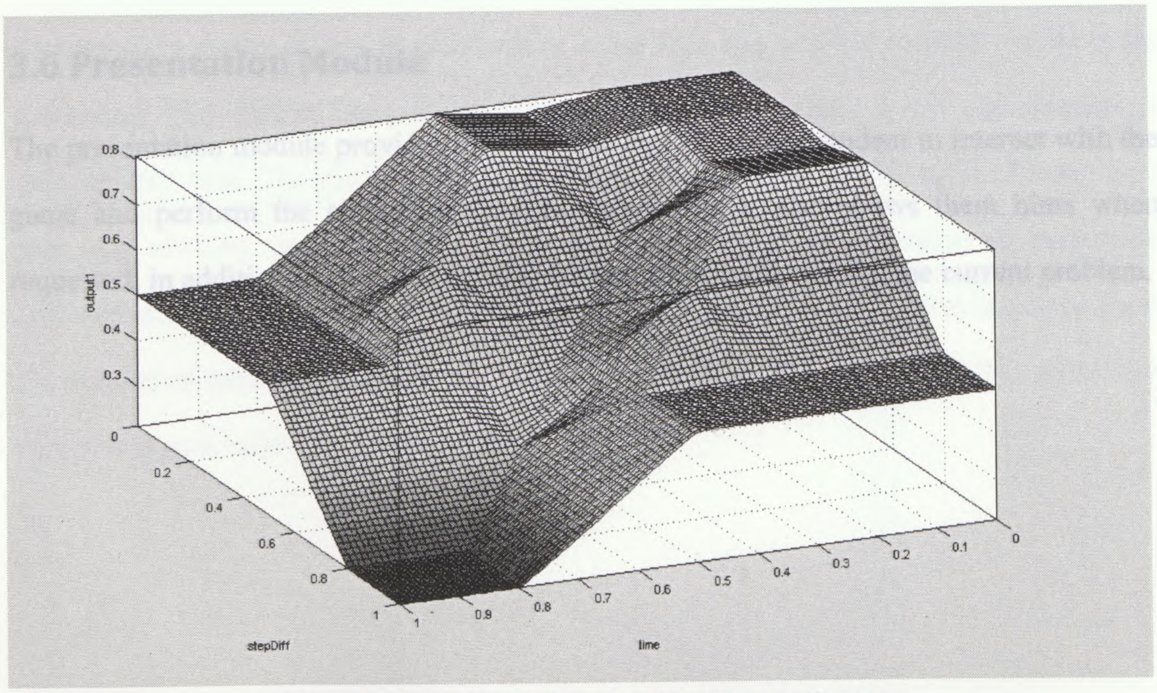


Figure 6. 3D Output Graph of the Fuzzy System

3.5.2 Smart Hint

A smart hint in this game is the next step that the pedagogical module gives to the student when the student gets stuck, which is the responsibility of the pedagogical module. The pedagogical module uses the automated solver to provide the student with the next step. Whenever the student takes a new step in the solution, the game records the step and compares it along with all previous steps taken by the student to the solution generated by the automated solver. If the steps taken by the student did not match the steps generated by the automated solver the pedagogical module uses the current student's step as a new problem and work on solving it using the automated solver so that it can provide the student with the next step. The student is free to apply that proposed operation or work his own.

3.6 Presentation Module

The presentation module provides the interface that allows the student to interact with the game and perform the elementary matrix operations. It also shows them hints when requested, in addition to time and number of steps they took to solve the current problem.

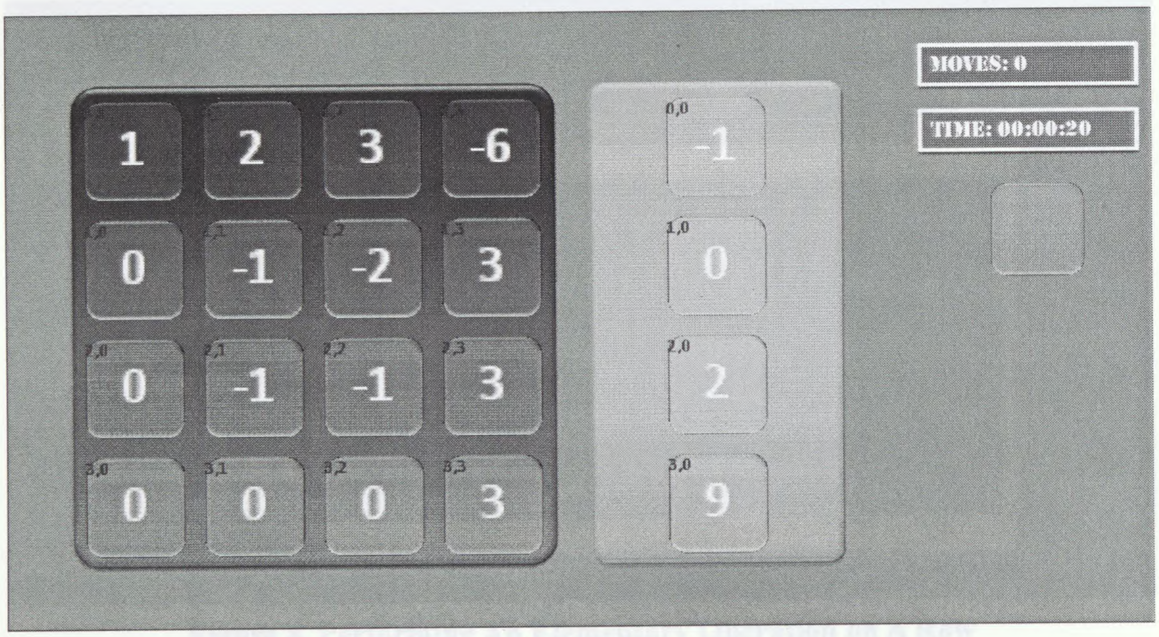


Figure 7. The Screenshot of The Game Level Screen

The interface shows the coefficient matrix in the blue box and the constant vector in the green box. The student can perform a row addition by dragging a row and dropping it onto another row. The student can perform a row multiplication by right-clicking on a row and clicking on the number pad. The row switching operation is performed automatically by the game. On the right side of the screen there are two labels to show the number of moves (steps) and the time elapsed for the current problem. The smart hint button is shown below these two labels.

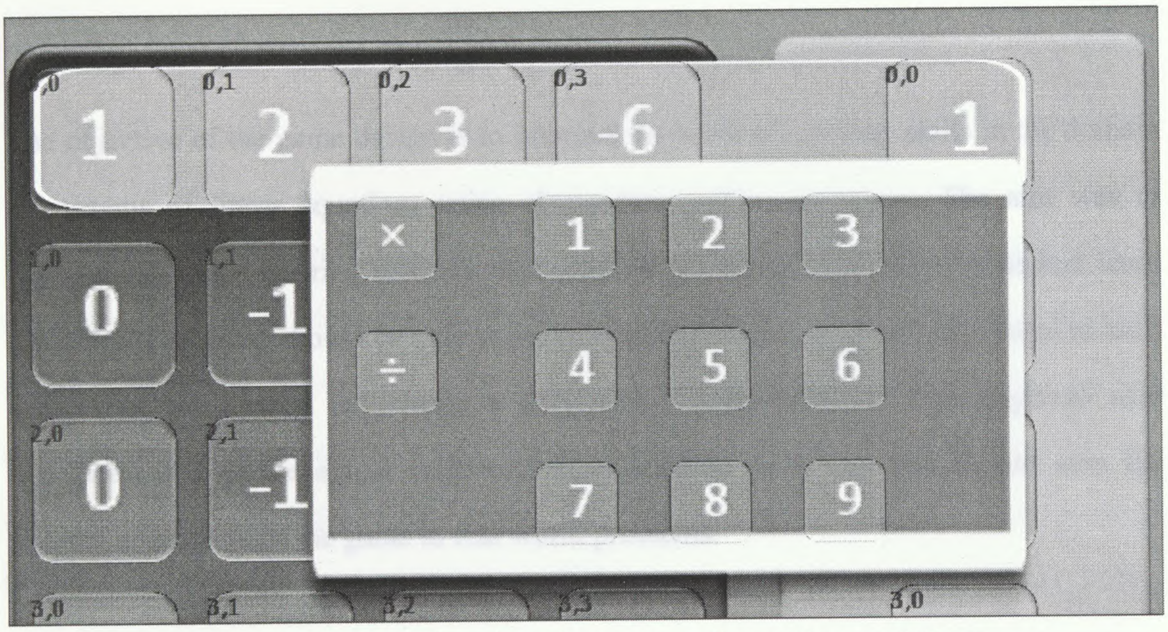
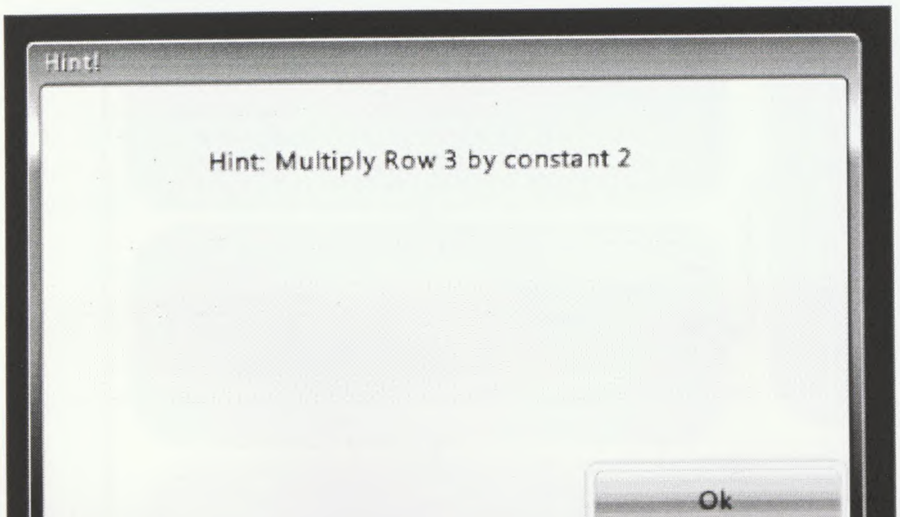


Figure 8. Performing An Elementary Operation on A Row

If the student clicks on the hint button a message box will shows up telling the student what the next step should be. The student can choose to follow the hint or not.



Hint: Multiply Row 3 by constant 2

Figure 9. A Screenshot of the Hint Message Box

3.7 Objective of Game Design

The objective of the game design is to improve the student's solving skills in the domain of systems of linear equations using elementary matrix operations. The aim was to develop intelligent tutoring modules that adapt to the student and help the student learn the skills. The game should be able to provide an individualized learning process to each student in which they experience a personalized learning path. The graphical user interface maps to the actual problems' representation in a way that should ease the transfer of skills from the game to real world problems.

Chapter 4 Implementation of MatriX

4.1 Overview

This Game is implemented using Microsoft XNA Game Studio 4.0 and programmed in the C-Sharp language. Microsoft XNA Game Studio is a video game developing platform based on the .Net Framework 4.0, which enables developers to develop video games for the Windows PC, Xbox 360 and Windows Phone (Aaron Reed, 2010). All the implemented modules in MatriX were implemented without using any third-party libraries, so that MatriX can run as a standalone application. To run MatriX, users only require the .Net Framework 4.0, XNA Framework 4.0, and Games for Windows Live installed on their PC.

4.1.1 Architecture

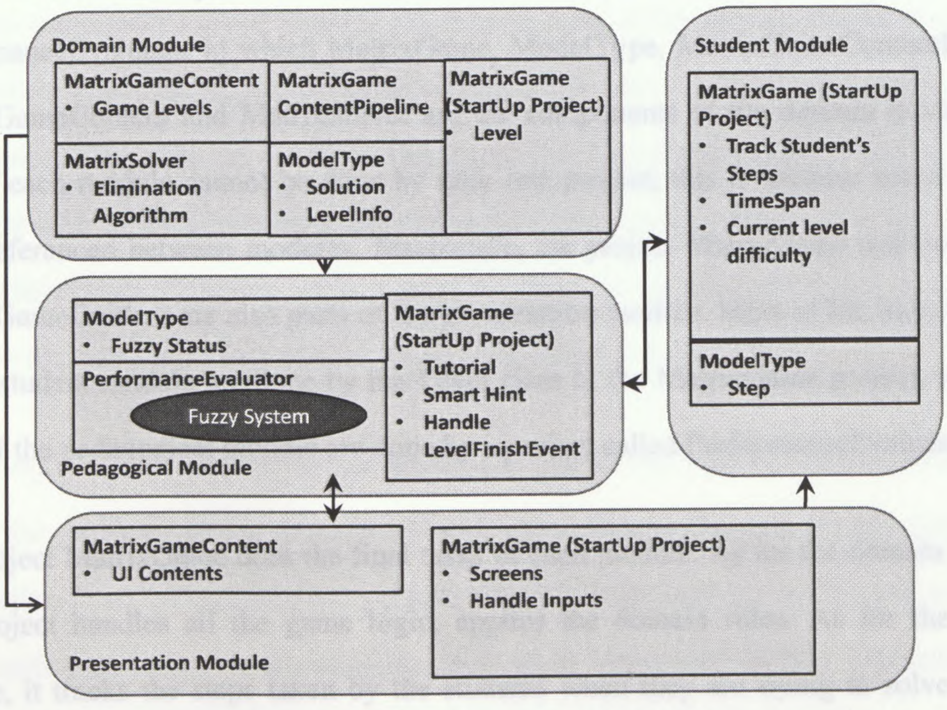


Figure 10. Architecture of MatriX

Figure 10 shows the architecture of MatriX. The way the different modules interact with one another is as follows: The domain module stores the problems as game levels and also calculates and stores the solutions of these problems. The presentation module reads the information of current level from the domain module and displays it on the screen and handles the inputs of the students. The student module gets the students inputs from the presentation module and tracks the students' performance. The pedagogical module requests the student's information from the student module and content information from the domain module upon which it determines the proper game level the student needs to play next, finally the pedagogical module informs the presentation module to present that level to the student.

The MatriX application consists of 6 projects – MatrixGame, ModelType, MatrixGameContentPipeline, MatrixGameContent, MatrixSolver and PerformanceEvaluator in which MatrixGame, ModelType, MatrixGameContentPipeline, MatrixGameContent and MatrixSolver are the components of the domain module. The jobs of each module cannot be done by only one project; this is because some data are cross-referenced between modules. Meanwhile, the project MatrixGame and the project MatrixGameContent are also parts of the presentation module. Most of the tasks required by the student module are done by the Level class in the MatrixGame project, while the tasks of the pedagogical module are done by a project called PerformanceEvaluator.

The project MatrixGame does the final tasks of each module. As for the domain module, this project handles all the game logic, applies the domain rules. As for the student module, it tracks the steps taken by the students when they are trying to solve a level, record the time span. And as for the pedagogical module, it comprehends information from the domain and student modules, gives smart hint and selects game levels. As for the presentation module, this project draws all the contents on the screen and manages screens used for different purposes. This project is also the application entry point and is built as an executable file.

The project ModelType is a part of the domain module, which contains the data structures that are used by the modules, such as the problem the student is going to solve, and stores the problem-solving solutions. As the domain module needs to communicate with other modules, this project is also referred to by other projects (modules), such as student and pedagogical modules.

The MatrixGameContent stores all the game assets such as textures and scripts. The game needs to read all the data it needs from this Content project through the content pipeline. The most commonly used types of assets are already supported by the XNA Game Studio, in which the default content pipeline can process most of the game contents stored in the hard drive. However, some of those contents are new to XNA, for example, level information. Therefore we needed to implement our own content pipeline to import those levels (system of linear equations problems) from the content project to the game. So the MatrixGameContentPipeline project imports those level asset files, processes them, converts them to a level information object, and passes it to the game.

The pedagogical module for this game needs to answer the student's request at any point during the problem solving process and tells them what to do next, so this module gets the solution from the MatrixSolver project, which is in the domain module and solves any given system of linear equations in the form of an augmented matrix. The pedagogical module is responsible for choosing the next game level for the student. The PerformanceEvaluator is implemented for this purpose, in which there is a fuzzy system that is in essence a neuro-fuzzy network. This module receives the student's performance from the student module and gives the difficulty degree of the next level, and then selects the next game level (problem) from the levels defined in the domain module. The neuro-fuzzy network in this module does fuzzy reasoning and is capable of doing back-propagation learning. However, we turned off the learning functions as we did not have enough data to train this neuro-fuzzy system, which might cause enormous inaccuracy if we leave the learning functions on.

4.2 Domain Module

As mentioned above, the domain module stores the problems that students need to solve, and applies the game rules, logic and the problem-solving skill that students need to learn. Those jobs are done by five projects – MatrixGame, MatrixSolver, MatrixGameContent, MatrixGameContentPipeline and ModelType.

4.2.1 Structure

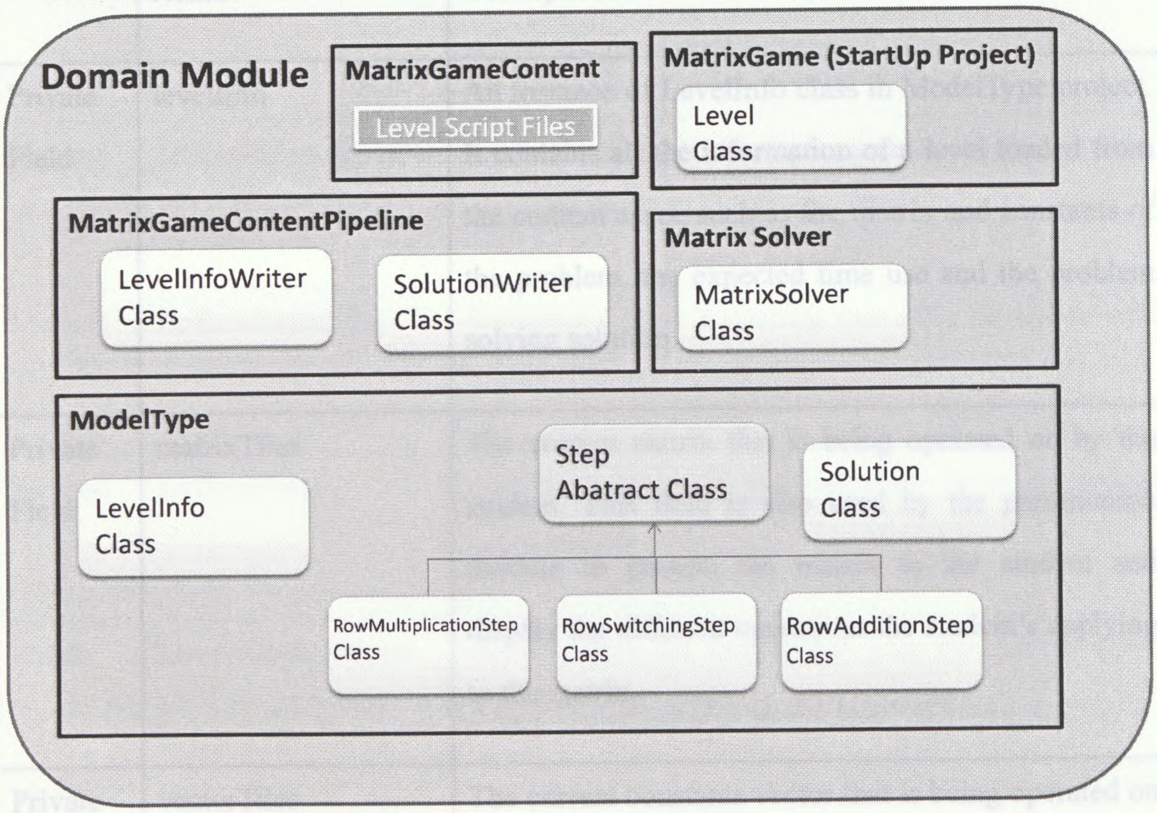


Figure 11. Domain Module Structure

As seen in Figure 11 the jobs of the domain module are allocated to these projects. The Level class represents a problem the student needs to solve, which in this domain module communicates with both the student and pedagogical modules.

4.2.2 Game Level

The most important class is the Level class, which represents a game level. It refers to all other projects, and manages the game logic and rules in addition to fulfilling the functions of the game. The level has the following members that works for the domain module.

Table 3. Members of Level for Domain Module

	Name	Description
Private Field	levelInfo	An instance of LevelInfo class in ModelType project. It contains all the information of a level loaded from the content asset, such as the matrix and constants of the problem, the expected time use and the problem solving solution.
Private Field	matrixTiles	The current matrix that is being operated on by the student. This field is also used by the presentation module to present the matrix to the student and display the different operations the student's applying to the matrix.
Private Field	vectorTiles	The current constants vector that is being operated on by the student, and is the extension part of matrix. This field is also used by the presentation module to present any constants to the student.
Property	ExpectedStepCount	Gets the expected step count of the current level from the levelInfo instance.

Property	ExpectedTimeUse	Gets the expected time of the current level from the levelInfo instance.
Property	IsWon	Gets the winning state of the current level; determines if the player has won the current level (solved the problem).
Method	DoRowAddition	Performs a Row Addition operation on the current matrixTiles and vectorTiles.
Method	DoRowMultiplication	Performs a Row Multiplication operation on the current matrixTiles and vectorTiles.
Method	DoRowSwitching	Performs a Row Switching operation on the current matrixTiles and vectorTiles.
Method	GetRestSteps	Gets the solution for the rest of the steps the student needs to take.
Event	LevelFinishedEvent	Occurs when the student has won the current level. This event is handled by the MatrixGame instance.

4.2.3 Matrix Solver

For the purpose of this thesis, we implemented an algorithm that performs elementary matrix operations to transform a matrix to an identity matrix. None of the existing Gaussian Elimination algorithms such as partial pivoting elimination algorithms seem to work for us. That was because, as discussed earlier, some new rules were added to the game as follows;

- No fractions
- Rule Swapping is operated only when at least one row is reduced and only by the computer automatically.
- The coefficient matrix should be transformed to an identity matrix instead of a row echelon form.

The approach is to eliminate the maximum eliminable element in the row with maximum number of non-zero entries all the time until the matrix is identity. The first step in the process is to check if the given matrix is an identity matrix. If it is then return finish computing. The next step is to check if we need to switch the rows. In this step, we apply the following transformation.

$$\left(\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & 0 & 0 & c_2 \\ a_{31} & a_{32} & 0 & c_3 \end{array}\right) \rightarrow \left(\begin{array}{ccc|c} a_{21} & 0 & 0 & c_2 \\ a_{11} & a_{12} & a_{13} & c_1 \\ a_{31} & a_{32} & 0 & c_3 \end{array}\right)$$

In the example above the second row is switched with the first row as the second row has the entry in the first column that is 1 and rest are 0s. This process makes sure the following form will never appear in the game.

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Then we go to the first step of elimination, which is to find the row that has the maximum number of non-zeroes. The row with the maximum number of non-zeroes will be the row that is added on by another row. For example in the following matrix, row no.2 will be selected.

$$\begin{pmatrix} d & e & 0 \\ a & b & c \\ 0 & f & g \end{pmatrix}$$

However, in some cases we do not want to select the row that has the maximum number of non-zeroes. For example in the matrix below, although the first row has the maximum number of non-zeroes, it will not be chosen to be operated on by other rows. If we add any of the rest of the rows on to the first row, the second entry on this row would not be equal to 0 anymore. And that is want we need to avoid.

$$\begin{pmatrix} a_{11} & 0 & a_{13} & a_{14} \\ 0 & a_{22} & 0 & 0 \\ 0 & a_{32} & a_{33} & 0 \\ 0 & a_{42} & 0 & a_{44} \end{pmatrix}$$

For the following matrix, the first row can be chosen, because the fourth row can be added to it and would not change the value of the 0 entry.

$$\begin{pmatrix} a_{11} & 0 & a_{13} & a_{14} \\ 0 & a_{22} & 0 & 0 \\ 0 & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{pmatrix}$$

However in some cases you might not find any row that can be selected, such as the following matrix:

$$\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ a_{31} & a_{32} & 0 & a_{34} \\ a_{41} & a_{42} & a_{43} & 0 \end{pmatrix}$$

In this case, we can select any row and check the count of non-zero numbers in the selected row. If the count number equals1 then it means the rest of the entries in this row are 0s. In this case we divide the selected row by the value of the only entry that is not zero. If the count number is greater than 1 then we find another row to eliminate the entry in the row. Therefore we need to decide which entry we need to eliminate. We first exclude those entries equal to 0 and then exclude the ones that cannot be eliminated by any row.

For example:

$$\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ 0 & 0 & a_{33} & 0 \\ a_{41} & 0 & a_{43} & a_{44} \end{pmatrix}$$

In the matrix above, we would not select the first entry in the first row as it is 0. We would not select the second one either, because it is not eliminable. So then we need to choose from the third and fourth ones. Normally we select the one with the maximum absolute value, but when we are selecting which column in the row should be eliminated, we need to avoid selecting the ones that cause the zeroes to be added to other non-zero values in the row we selected. For example in the above matrix, we won't select the fourth entry in the first row to eliminate. In this case we exclude those rows that has non-zero at the column that we have 0 in the row we selected earlier. Then we select the entry with the maximum absolute value from those entries which can be eliminated by the rows that are not excluded.

After we select the cell we want to eliminate, we identify its row and work on selecting another row to add it to the identified row. If there is only one row available we should select that row, otherwise we choose the one that has the maximum number of zeroes.

Once we decide on the cell to be eliminated and the row that can eliminate this cell, we first calculate the least common multiple of the number we are going to eliminate and the number that eliminates it. Then we do row multiplication to those two rows by the quotient of the least common multiple and these two numbers. And then do the row addition to eliminate the cell. The process would look like this:

In $\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ 0 & 0 & a_{33} & 0 \\ a_{41} & 0 & a_{43} & a_{44} \end{pmatrix}$, $\mathrm{n} = \mathrm{LCM}(\mathrm{a}_{13}, \mathrm{a}_{33})$, therefore,

$$\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{21} & 0 & a_{23} & a_{24} \\ 0 & 0 & a_{33} & 0 \\ a_{41} & 0 & a_{43} & a_{44} \end{pmatrix} \rightarrow \begin{pmatrix} 0 & a_{12} * \left(\frac{n}{a_{13}}\right) & a_{13} * \left(\frac{n}{a_{13}}\right) & a_{14} * \left(\frac{n}{a_{13}}\right) \\ a_{21} & 0 & a_{23} & a_{24} \\ 0 & 0 & a_{33} * \left(\frac{n}{a_{33}}\right) & 0 \\ a_{41} & 0 & a_{43} & a_{44} \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 0 & a_{12} * \left(\frac{n}{a_{33}}\right) & 0 & a_{14} * \left(\frac{n}{a_{33}}\right) \\ a_{21} & 0 & a_{23} & a_{24} \\ 0 & 0 & a_{33} * \left(\frac{n}{a_{13}}\right) & 0 \\ a_{41} & 0 & a_{43} & a_{44} \end{pmatrix}, \mathrm{n} = \mathrm{LCM}(\mathrm{a}_{13}, \mathrm{a}_{33})$$

Then we apply the same process recursively until we have an identity matrix. To avoid an over-flow exception, before each recursive step we included a process that reduces each row of the matrix by the greatest common divisor of each row. However, this doesn't solve the problem permanently.

To test the stability of the developed algorithm, a tool was implemented to test all the possible 2 by 2 matrices, 10,000 random 3 by 3 matrices and 10,000 random 4 by 4 matrices, in which all entries were ranged from 0 to 9. The results are shown in the Table 4.

Table 4. Failure Rate of the Algorithm

Dimension	Failure Rate
2 by 2	0%
3 by 3	0%
4 by 4	3.128%

As we can see from Table 4, 3.128% of the 4 by 4 matrix problems cannot be solved using this algorithm, but this is good enough for this game. To avoid the problems that occur in 4 by 4 matrices, it becomes important to limit the number of steps of the recursive call. On the other side, no problems were encountered with all matrix problems of size less than 4x4.

4.3 Student Module

The student module tracks the student's moves and reflects on the student's performance to the pedagogical module, which it is implemented in the Level class mentioned earlier. The Level class does not only represent a problem the student needs to solve, but also gives the stage for the student to perform so that the game can track the student's moves. The following table shows the members that works for the student module in the level class.

Table 5. Members of Level for Domain Module

	Name	Description

Private Field	stepsRecord	A list of instances of the ModelType.Step abstract class that is used to records the students' steps.
Method	DoRowAddition	This method records one step into the stepsRecord after an operation is performed successfully.
Method	DoRowMultiplication	This method records one step into the stepsRecord after an operation is performed successfully.
Method	DoRowSwitching	This method records one step into the stepsRecord after an operation performed successfully.
Event	LevelFinishedEvent	Occurs when the student has won this level. This event passes the LevelFinishedEventArgs to the pedagogical module, in which the time span, step count and current level difficulty will be used by the pedagogical module to evaluate the student performance.

4.4 Pedagogical Module

The pedagogical module allows MatriX to adapt to individual students using a neuro-fuzzy system. The pedagogical module receives information on the student's performance from the student module and content information from the domain module to help it make strategic decisions about the student's learning process.

4.4.1 Fuzzy system

The structure of the neuro-fuzzy system is a neural network with nodes acting as neurons connected together forming links. Each node gathers input values from all previous nodes linked to it (Michael Negnevitsky, 2011).

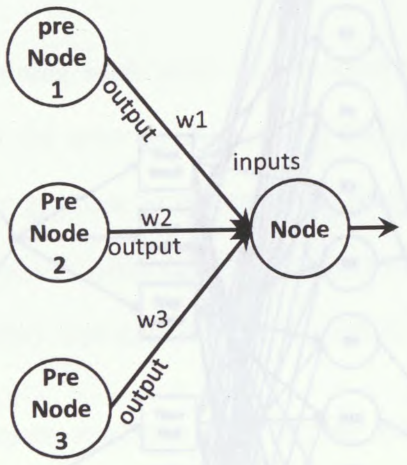


Figure 12. Getting Inputs from Previous Nodes

A method called UpdateOutput is used by the nodes to generate an input array, in which each entry equals the output value from previous node times the weight and minus the threshold as the formula below.

$$input_n = \sum output_m * weight_m - threshold$$

However, in this program, each layer has a switch to indicate if the training and weights are enabled in the current layer. If the switch is turned off (Boolean value equals false) then each value in the input array would simply be equal to the output value from the previous node.

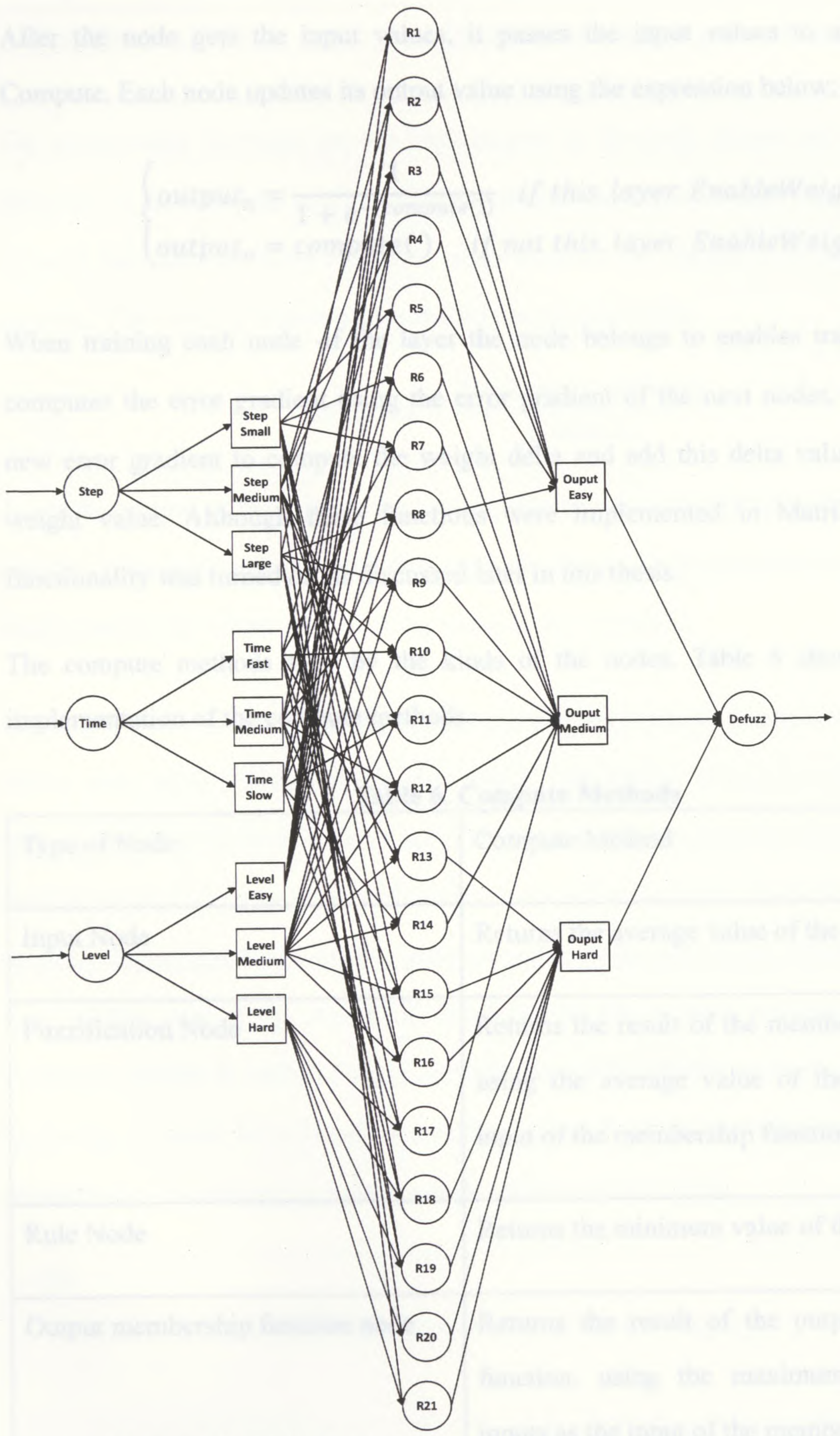


Figure 13. The Structure of the Neuro-fuzzy System

After the node gets the input values, it passes the input values to a method called Compute. Each node updates its output value using the expression below;

$$\begin{cases} output_n = \dfrac{1}{1+e^{-(compute())}} & if\ this.layer.EnableWeights \\ output_n = compute() & if\ not\ this.layer.EnableWeights \end{cases}$$

When training each node -if the layer the node belongs to enables training- the node computes the error gradient using the error gradient of the next nodes, then it uses the new error gradient to compute the weight delta and add this delta value to the current weight value. Although these functions were implemented in MatriX, the learning functionality was turned off as discussed later in this thesis.

The compute methods vary by the kinds of the nodes. Table 6 shows the different implementation of the compute methods.

Table 6. Compute Methods

Type of Node	Compute Method
Input Node	Returns the average value of the inputs
Fuzzification Node	Returns the result of the membership function, using the average value of the inputs as the input of the membership function
Rule Node	Returns the minimum value of the inputs
Output membership function node	Returns the result of the output membership function, using the maximum value of the inputs as the input of the membership function

Defuzzification Node	Returns the result of centroid function

The membership functions are not implemented in the node classes and are invoked as delegates in the compute methods. These membership functions are implemented by the instance that uses the fuzzy system in order to allow the system to be as generalized as possible. The centroid function uses the following expression:

$$centriod = \frac{\sum output_i * centers_i}{\sum ouput_i}$$

To get the output of the system, the system searches the network layer by layer and then searches each layer node by node then calls all update output methods for each node, and finally returns the output values of the nodes in the last layer.

To explain how to use this fuzzy system, here we have a simple example. First we need to define your input nodes, fuzzification nodes, rule nodes, output-membership-function nodes and defuzzification nodes. Each node has a member called PreviousNode, which is used to connect the new node with nodes in the previous layer. There is also a member property called NextNode used to connect nodes in the next layer. However we do not need to take care of the NextNode properties as this is done automatically when the EndBuild method is called. Finally BeginBuild and EndBuild methods are called at the beginning to build the network and at the end to finish the building process building, respectively. The following code is used to define an input node.

C#

```
    InputNode inputNode = new InputNode() { Name = "input_1" };
```

After then the fuzzification nodes are defined. When building a new fuzzification node, we need to assign a property in the node called PreviousNodes and a suitable method to the membership function delegate. In the invoked membership function delegate, some other functions such as trimf and trapmf were invoked from the Math class, in which the trimf represents a triangular-shaped membership function and the trapmf represents a trapezoidal-shaped membership function. Sample code follows:

C#

```
FuzzificationNode fuzz1 = new FuzzificationNode()
{
    PreviousNodes = new List<Node>() { inputNode },
    Name = "fuzz1",
    MembershipFunction = (n) =>
    {
      return MathHelper.trimf(n.InputValues.Average(), 0, 0.2, 0.5);
    }
};

FuzzificationNode fuzz2 = new FuzzificationNode()
{
    PreviousNodes = new List<Node>() { inputNode },
    Name = "fuzz2",
    MembershipFunction = (n) =>
    {
      return MathHelper.trampf(n.InputValues.Average(), 0.3, 0.5,0.6, 1);
    }
};
```

After defining the fuzzification nodes, we define the rule nodes in which the previous nodes property was assigned by the fuzzification nodes defined earlier as shown below;

C#

```
RuleNode rule1 = new RuleNode()
{
    PreviousNodes = new List<Node>() { fuzz1 },
               Name = "R01"
};
RuleNode rule2 = new RuleNode()
{
    PreviousNodes = new List<Node>() { fuzz2 },
               Name = "R02"
};
```

Then we define the output membership function nodes. When we are building a new node, we need to assign a rule node or several to the previous node property. Also we need to assign a method to the output membership function delegate, for which we used lambda expression again in the sample code. In the method that we assigned to the output membership function delegate, we called some functions such as triArea and trapArea, in which the triArea calculates the area form by a triangular-shaped membership function and the trapArea calculates the area form by a trapezoidal-shaped membership function. Also we assign the center value of the membership functions, which would be used for centroid calculation in deffuzification. The sample codes follow:

C#

```
 OutputMembershipFunctionNode out1 = new OutputMembershipFunctionNode()
   {
     PreviousNodes = new List<Node>() { rule1 },
                  Name = "out1",
                  OutputMembershipFunction = (n) =>
                  {n.Center=0.5; return MathHelper.trapArea(n.InputValues.Max(), 0.25,
0.4, 0.6, 0.75);
 }
              };
OutputMembershipFunctionNode out2 = new OutputMembershipFunctionNode()
{
   PreviousNodes = new List<Node>() { rule2 },
                  Name = "out2",
   OutputMembershipFunction = (n) =>{n.Center=0.75;
   return MathHelper.triArea(n.InputValues.Max(), 0.5, 0.75, 1e+06);
 }};
```

Then we define the defuzzification node, and link it with all the output membership nodes by assigning the previous nodes' properties.

C#

```
DefuzzificationNode defuzziNode = new DefuzzificationNode() { PreviousNodes = new List<Node>()
{ out1, out2 }, Name = "DefuzzNode" };
```

Then the BeginBuild method is called to let the system know the network is built but not ready to use. And then we add those nodes defined earlier to relative layers, and indicate which layer would enable learning and weights (currently disabled in this work). After that the learning rate is declared if the learning method is enabled in any layer. Finally, we call the EndBuild method, which automatically assigns the nextNodes properties in each node that links one node with the next one.

C#

```
nfs.BeginBuild();
nfs.InputLayer.AddNodes(inputNode);
nfs.FuzzificationLayer.AddNodes(fuzz1, fuzz2);
nfs.FuzzificationLayer.EnableWeights=false;
nfs.RulesLayer.AddNodes(rule1, rule2);
nfs.RulesLayer.EnableWeights = false;
nfs.OutputMembershipFunctionLayer.AddNodes(out1,out2);
nfs.OutputMembershipFunctionLayer.EnableWeights = false;
nfs.DefuzzificationLayer.AddNodes(defuzziNode);
nfs.DefuzzificationLayer.EnableWeights = false;
nfs.EnableHiddenLayer = false;
nfs.LearningRate = 0.3;
nfs.EndBuild();
```

After the pedagogical module decides on the difficulty level of the next problem to present to the student, it selects a problem that has the closest difficulty degree to the one given by the fuzzy system from the domain module. The pedagogical module also checks if this problem has been solved earlier by the student, if it is the case it requests another problem from the domain module. The code is as follows:

C#

```
    float nextDiff = performanceEvaluator.GetNextLevelDifficulty(stepDiff, timeDiff,
e.LevelDifficulty,out fs);
    int preIndex = currentLevelIndex;
```

```
currentLevelIndex = levels.IndexOf(nextDiff);
if (preIndex < 2)
  currentLevelIndex = preIndex + 1;
else if (fs.StepCount == ModelType.StepCount.TooMany && (fs.TimeUsed ==
        ModelType.TimeUsed.Medium || fs.TimeUsed ==
        ModelType.TimeUsed.TooLong)&&(preIndex+1<currentLevelIndex))
            currentLevelIndex = preIndex + 1;
else if (currentLevelIndex == preIndex)
{
    switch (fs.NextLevelDifficulty)
    {
        case ModelType.LevelDifficulty.Easy:
            currentLevelIndex++;
            break;
        case ModelType.LevelDifficulty.Medium:
            currentLevelIndex++;
            break;
        case ModelType.LevelDifficulty.Hard:
            if ((currentLevelIndex + 1) >= levels.Count)
            {
                foreach (var in levels)
                {
                    if (l.Difficulty >=0.75)
                    {
                        currentLevelIndex = levels.IndexOf(l.Difficulty);
                        break;
                    }
            }
        }
        else
            currentLevelIndex = preIndex+1;
        break;
    default:
            break;}}
```

4.4.2 Smart Hint

The pedagogical module gives the student a smart hint on what to do next when the student is stuck while trying to solve the problem at hand. When the student clicks on the Smart Hint button a message box pops up and tells the students what to do next. While testing the smart hint in MatriX, a loop glitch happened in which the pedagogical module keeps repeating the same hint. This occurred when the student tries to eliminate an entry in the matrix. For example, let's consider to eliminate one row in the following matrix:

$$\left(\begin{array}{ccc|c} 3 & 2 & 4 & 1 \\ 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

The program believes we should eliminate the upper left cell in the first row (the digit 3). This can be achieved by multiplying the first row by 2 as follows:

$$\left(\begin{array}{ccc|c} 6 & 4 & 8 & 2 \\ 2 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

After the student followed this hint, MatriX would regenerate a new solution. As in the algorithm we mentioned earlier, the first step would be each row in the matrix, thus the hint would give the students the hint that they need to divide the first row by 2. But the correct hint should be multiply the second row by 3.

In order to avoid this issue, each step the student takes is tracked and compared to the generated solution. If the student's solution follows that solution generated by MatriX then the hints are provided directly from the generated solution. If it is not the case, then Matrix redefines the problem and uses the student's current step as the new problem and generates the solution for it. The code is as follows:

Pseudo

```
For n = 0...n
    If result matrix of step n in previousSolution == currentMatrix Then
            Add (n + 1) to startingIndexes list
    End if
Next
If count of startingIndexes != 0 Then
        result = Copy previousSolution from the max index of starting indexes
End if
```

Chapter 5 Evaluation

In order to evaluate this game, we recruited 13 students from Columbus State University to test the game, most of who were undergraduate students with couple of graduate students. All game testers were requested to finish a pre-test, play the game for 3 days and then take a post- test and a survey about their play experience.

5.1 Goal

The goal of the evaluation is to answer following questions,

- Can the game help the player learn how to solve linear equations by using Gaussian Elimination?
- Is the game's user interface friendly enough?
- Can the game adapt successfully to individual students?

5.2 Experimental Testing

In order to answer these questions, we invited 13 students to take a pre-test, play the game for 3 days and finally answer a post-test and a survey about their play experience. In the pre-test, there were some questions asking about their educational level, gender and algebra courses they had taken followed by a set of questions on systems of linear equations. In the post test similar math problems to those in the pre-test were introduced..

The following table shows the questions in the pre-test.

Table 7. Pre-test Interviews Questions

Question 0 Educational level and gender.
Question 1 Have you ever taken any algebra courses or related courses? If yes, please list the names of the courses.
Question 2 Have you learnt linear algebra before?
Question 3 Please try to solve the following equations or sets of equations. i. $3x + 5 = 4$ ii. $\begin{cases} 3x + 5y = 10 \\ x + 3 = 8 \end{cases}$

iii. $\begin{cases} x+y+z=12 \\ x+2y+5z=22 \\ x=4y \end{cases}$

iv. $\begin{cases} l+m=5 \\ l+m+p=5 \\ l+2m+n+p=8 \\ 2l+m+2p=8 \end{cases}$

v. $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & -1 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

5.2.1 Demographic Data

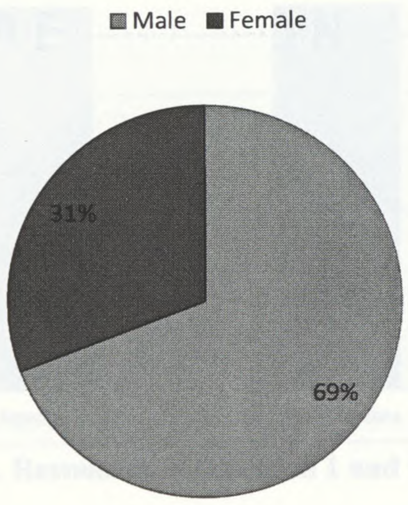


Figure 14. Gender Ratio

In this evaluation, 31% students are female and 69% are male (Figure 14). In addition, as we can see from the figure 15, we got 39% students were graduate students and 61% were undergraduate, in which 8% were freshmen, 15% were sophomore, 23% were junior and 15% were senior.

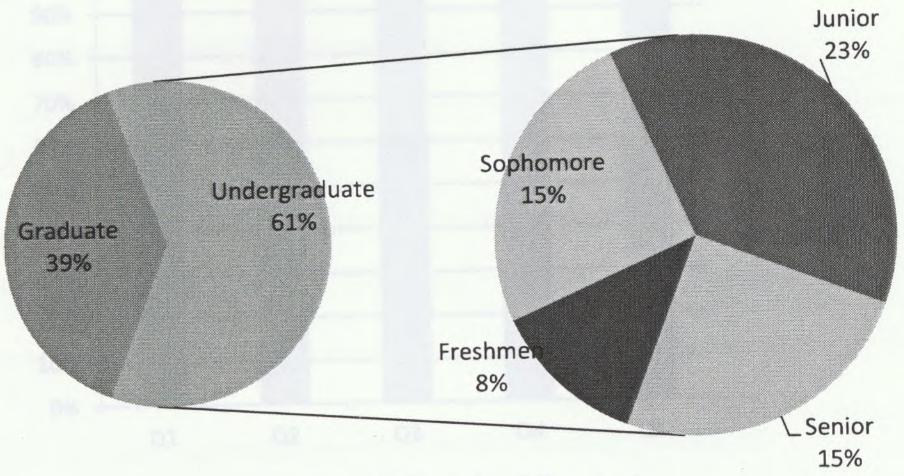


Figure 15. Students' Educational Levels

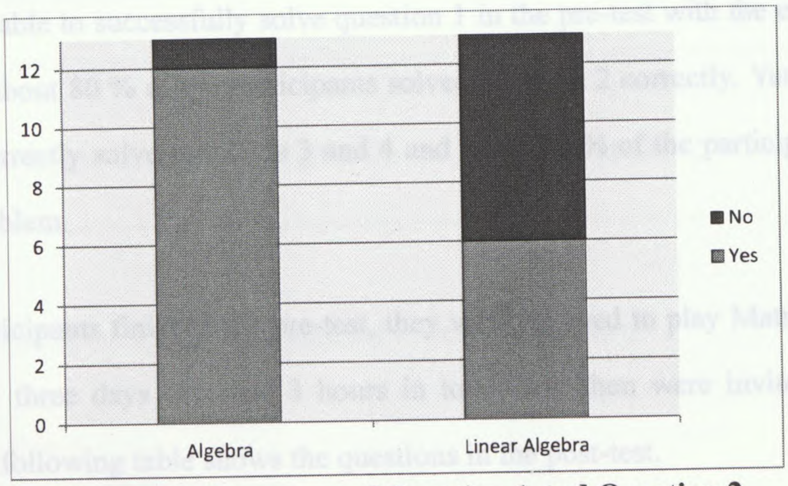


Figure 16. Responses of Question 1 and Question 2

Responses of questions 1 and 2 show that 6 students learned linear algebra and 7 students did not. However from the result of the pre-test, showed in Figure 17, only 1 student was able to correctly solve question no.5 in the pre-test, which indicates that even those who had linear algebra background do not have competent skills in this domain.

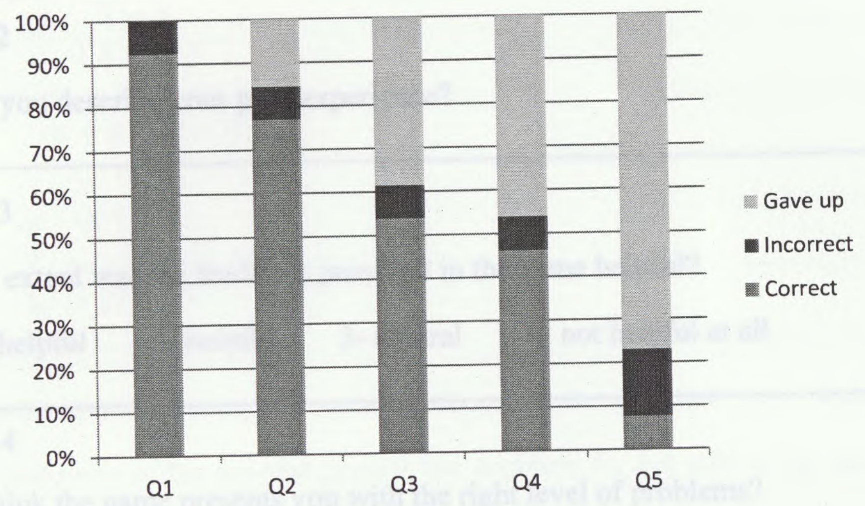


Figure 17. Result of Pre-test

Figure 17 shows that the participants gave up easily on the harder problems. Most students were able to successfully solve question 1 in the pre-test with the exception of 1 student. And about 80 % of the participants solved problem 2 correctly. Yet only 55% of them could correctly solve problems 3 and 4 and nearly 80% of the participants gave up on the last problem.

When the participants finished the pre-test, they were allowed to play MatriX for couple of hours over three days (average 3 hours in total) and then were invited back for a post-test. The following table shows the questions in the post-test.

Table 8. Post-test Interviews Questions

Question 1 How easy was it to figure out the rules of the game? 1- Very easy 2- easy 3- neutral 4-difficult 5- very difficult.

Question 2

How do you describe your play experience?

Question 3

To what extent was the feedback provided in the game helpful?

1- Very helpful 2- helpful 3- neutral 4- not helpful at all

Question 4

Do you think the game presents you with the right level of problems?

1- problems were at the right level

2- problems were too easy

3- problems were too hard

Question 5

What would be a good change to the game to make it more appealing to you?

Question 6

Please try to solve the following equations or sets of equations.

i. $3x - 5 = 4$

ii. $\begin{cases} 5x + 3y = 27 \\ 2x + y = 10 \end{cases}$

iii. $\begin{cases} 5x + 3y + 2z = 17 \\ 3x + 2y + 2z = 13 \\ 2x + y + 2z = 10 \end{cases}$

iv. $\begin{cases} l = 1 \\ l + n = 3 \\ l + 3m + 2n + 4p = 15 \\ m + n + p = 8 \end{cases}$

v. $\begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & -1 \\ 1 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

Besides some new questions, the math problems in the post-test were slightly different from the problems in the pre-test (except the problem 5), but still in the same difficulty levels. The problem 5 had no change as most of them gave it up in the pre-test.

The following figure shows the students' performance in the post-test.

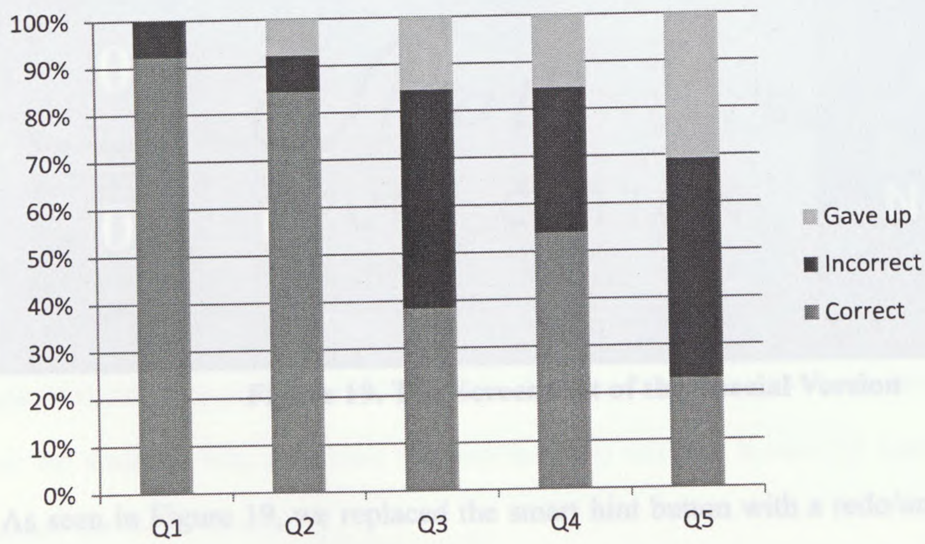


Figure 18. Result of Second Test

As seen in Figure 18, the correction rate of the problems 2 and 4 increased slightly, and the correction rate of problem 5 changed from 9% to 23%. And the gave-up rate decreased significantly for problems 3, 4 and 5, which is a good indication in itself. On the other hand, the correction rate for the problem 3 dropped by 15%. These results made us eager to look closer at the collected data which lead us to interesting results.

Looking at the solution steps, we found that although the game might have improved the desired operations, it did not help the students improve their calculation skills. To remove this interfering factor, we asked those students to take the post-test again while allowing them to use a special version of the game as a calculator.

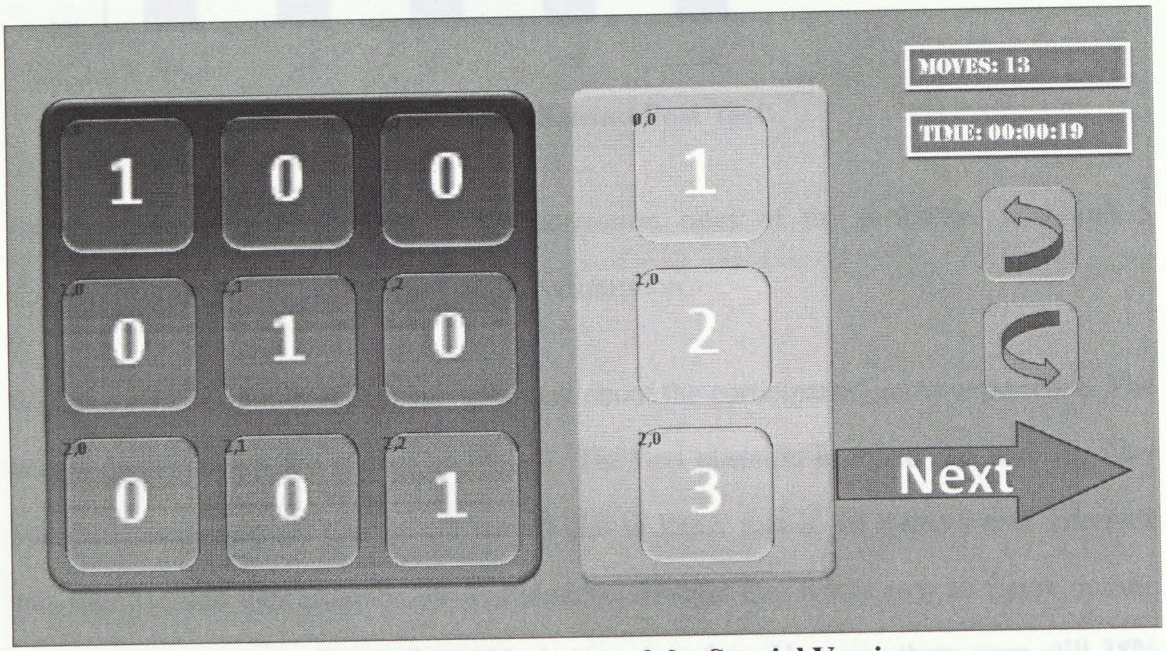


Figure 19. The Screenshot of the Special Version

As seen in Figure 19, we replaced the smart hint button with a redo/undo button, which allows the students to view all the steps they take. When the student finishes a level, the game would not go to the next level until the student clicks on the next button.

43% of the students agreed to retake the test. The new results are shown in Figure 20.

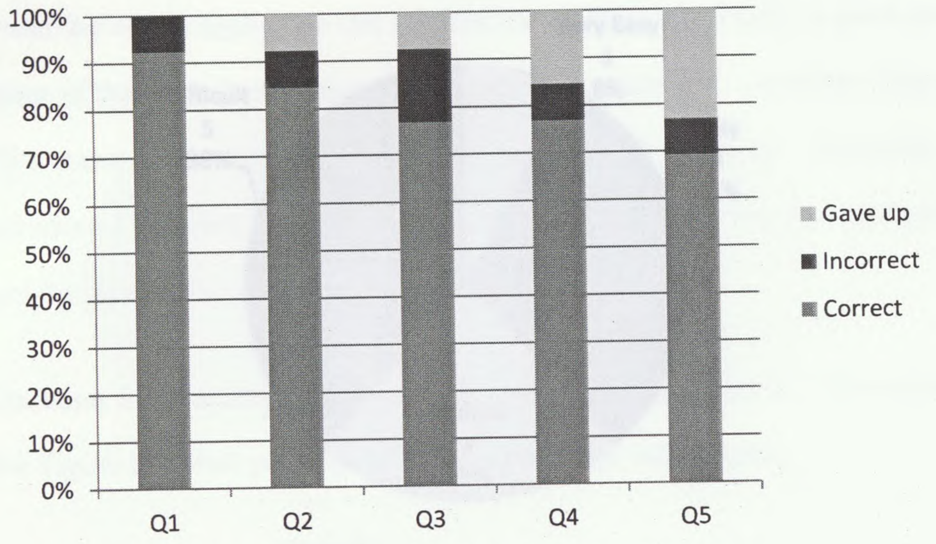


Figure 20. Results of the Second Post Test

As we can see from Figure 20, the correction rates of the problems 3, 4 and 5 significantly increased, which proves our assumption.

We have also developed a post survey to ask about the participants' game experience. The results of the survey are shown in Table 7. The first question is concerned with the user interface, as mentioned earlier; we would like to know that if the game's user interface was intuitive and user friendly. 39% of students thought that it was easy to figure out the game rules through the interaction with the user interface. However there were still 38% of the students who criticized the interface and felt that it was not intuitive enough and asked for more directions.

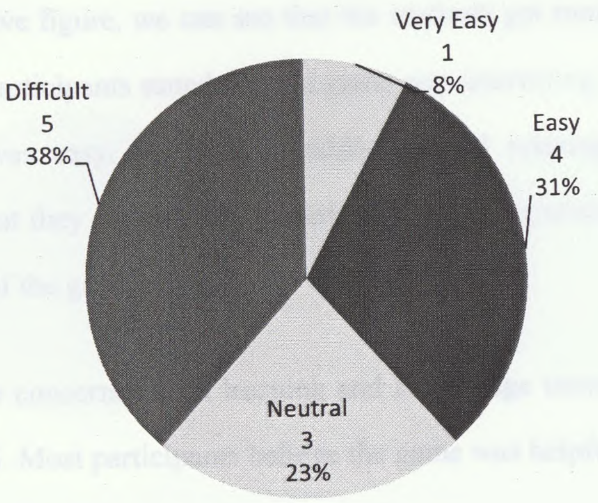


Figure 21. How Easy to Figure Out the Rules

The second question in the survey is concerned with the general game experience. The results are shown in Figure 22.

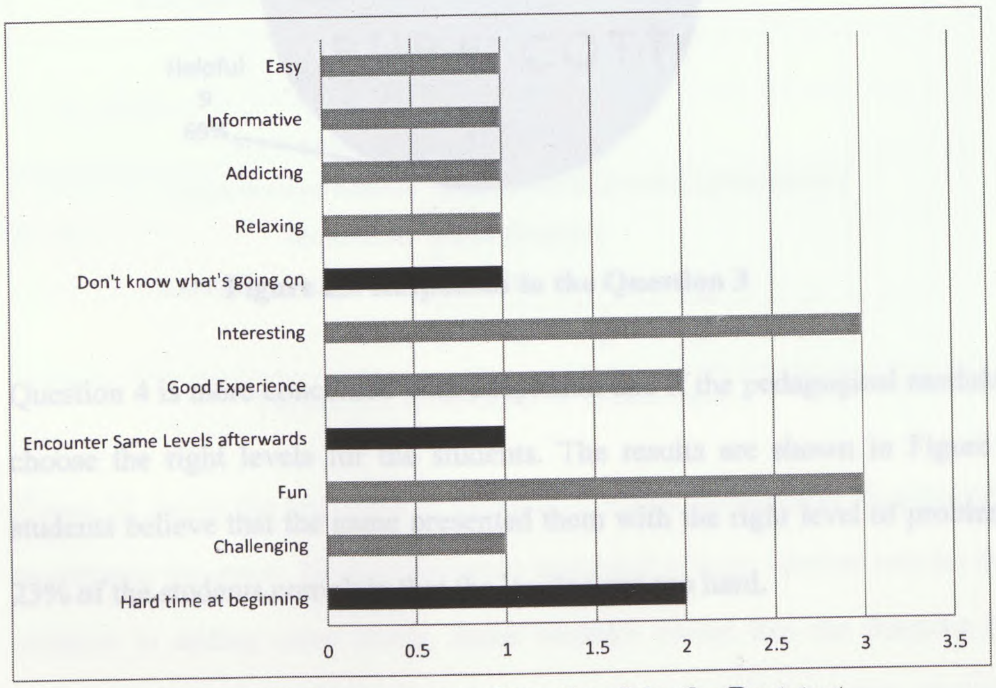


Figure 22. Responses to Question 2 in the Post-test

From the above figure, we can see that the students got mostly a good game experience. Most of the participants stated that the game was interesting and fun. One participant said "the game was easy, informative, addicting and relaxing". However, 2 participants mentioned that they experienced a hard time at the beginning and one participant did not get the idea of the game at all.

Question 3 is concerned with learning and knowledge transfer. The results are shown in the Figure 23. Most participants believe the game was helpful.

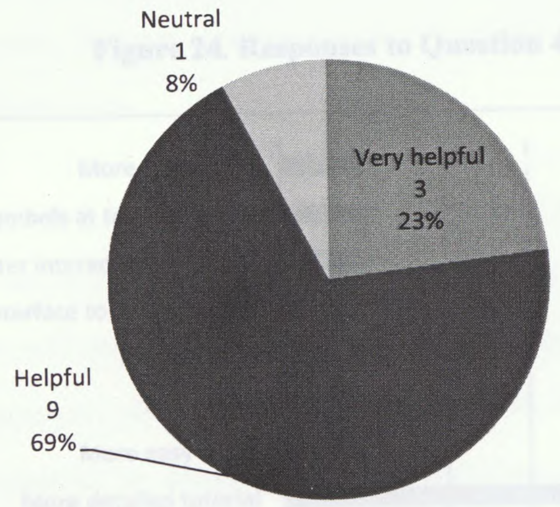


Figure 23. Responses to the Question 3

Question 4 is more concerned with adaptation and if the pedagogical module was able to choose the right levels for the students. The results are shown in Figure 24; 77% of students believe that the game presented them with the right level of problems. However 23% of the students complain that the levels were too hard.

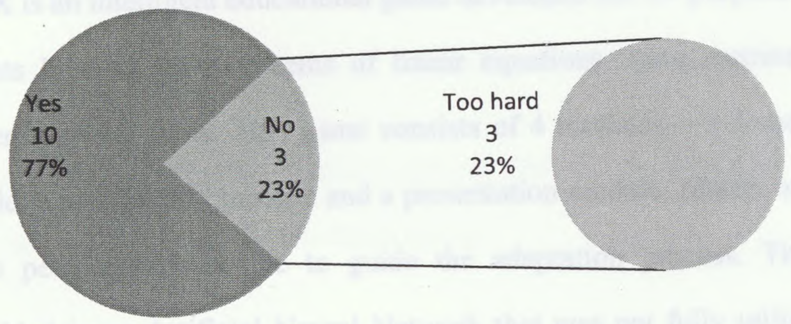


Figure 24. Responses to Question 4

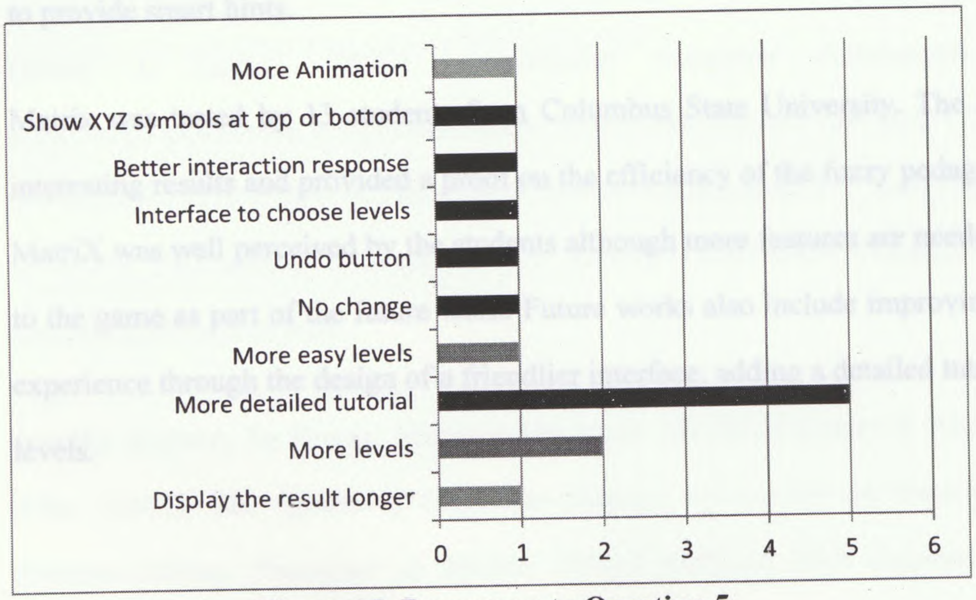


Figure 25. Responses to Question 5

Figure 25 shows the responses to question 5, which asked about the participant's opinion to enhance the game. Most participants requested a more detailed tutorial in the game in addition to adding more levels. Some students would like the freedom to choose the levels they would like to play. Some mentioned that the game should have some animation as the automatic row switching sometimes would confuse them.

Chapter 6 Conclusion

MatriX is an intelligent educational game developed for the purpose of this thesis to teach students how to solve systems of linear equations using matrices and improve their problem solving skills. This game consists of 4 modules – a domain module, a student module, a pedagogical module and a presentation module. Matrix utilized a fuzzy system in the pedagogical module to guide the adaptation process. The fuzzy system was embedded in an Artificial Neural Network that was not fully utilized in this work. The important achievement of this thesis is the implementation of an automatic solver that solves a system of linear equation problems instantly and helps the pedagogical module to provide smart hints.

Matrix was tested by 13 students from Columbus State University. The study showed interesting results and provided a proof on the efficiency of the fuzzy pedagogical module. MatriX was well perceived by the students although more features are needed to be added to the game as part of the future work. Future works also include improving the player's experience through the design of a friendlier interface, adding a detailed tutorial and more levels.

References

Steve K. Robbins, (2013), *Problem Solving: Techniques, Strategies & Skills for Solving Problems*, CreateSpace Independent Publishing Platform.

Bonnie Averbach, (2012), *Problem Solving Through Recreational Mathematics (Dover Books on Mathematics)*, Dover Publications.

Rania Hodhod, (2010), *Interactive Narrative for Adaptive Educational Games: Architecture and an Application to Character Education*, Ph.D. Thesis. The University of York

Robert J. Lopez, (2010), *Introduction Gaussian Elimination*, Maplesoft. https://mail-attachment.googleusercontent.com/attachment/u/0/?ui=2&ik=73efa70173&view=att&th=1477e2cdafd19026&attid=0.1&disp=safe&realattid=f_hy6455ly1&zw&saduie=AG9B_P9yCwszkoYN3zEpXS-bmUV_&sadet=1406576467552&sads=6C6uhAoaT-SlHY6TrqlWQQu16WM

Jennifer Krawec, Jia Huang, Marjorie Montague, Benikia Kressler & Amanda Melia de Alba. (2012). *The Effects of Cognitive Strategy Instruction on Knowledge of Math Problem-Solving Processes of Middle School Students With Learning Disabilities.* Hammill Institute on Disabilities 2012.

Mayer, R. E. (1985). Mathematical ability. In R. J. Sternberg (Ed.), *Human abilities: An information processing approach* (pp. 127–150). San Francisco, CA: Freeman.

Robert W. Maloy, Sharon A. Edwards & Gordon Anderson. (2010). *Teaching Math Problem Solving Using a Web-based Tutoring System, Learning Games, and Students'*

Writing. University of Massachusetts Amherst. Journal of STEM Education Volume 11 • Issue 1 & 2 January-June 2010

Polya, G. (1973). *How to solve it: A new aspect of mathematical method. Princeton*, NJ: Princeton University Press.

Damien Djaouti, Julian Alvarez, Jean-Pierre Jessel, Olivier Rampnoux. (2011). *Origins of Serious Games*, ludoscience, Springer.

Irene Polycarpoua, Julie Krausea, Cyndi Radera, Chad Kembela, Christopher Pouporea & Eric Chiu, (2010) *Math-City: an educational game for K-12 mathematics*. Procedia Social and Behavioral Sciences 9 (2010) 845–850

Regina Stathacopoulou, (2006). *Student modeling using fuzzy logic and neural networks*. Department of Informatics and Telecommunications, University of Athens, Panepistimiopo-lis, GR-15784 Athens, Greece

Aaron Reed, (2010), *Learning XNA 4.0: Game Development for the PC, Xbox 360, and Windows Phone* 7, O'Reilly Media.

Michael Negnevitsky (2011), *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson Education Limited. pp268-277