7-2017

Using Pedagogical Tools to Help Hispanics be Successful in Computer Science

Rodger Irish
The University of Texas Rio Grande Valley

USING PEDAGOGICAL TOOLS TO HELP HISPANICS BE SUCCESSFUL IN COMPUTER SCIENCE

A Thesis

by

RODGER IRISH

Submitted to the Graduate College of
The University of Texas Rio Grande Valley
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

July 2017

Major Subject: Computer Science

USING PEDAGOGICAL TOOLS TO HELP HISPANICS

BE SUCCESSFUL IN COMPUTER SCIENCE

A Thesis
by
RODGER IRISH

COMMITTEE MEMBERS

Dr. Andres Figueroa
Chair of Committee

Dr. Angela Chapman
Committee Member

Dr. John Abraham
Committee Member

Dr. Zhixiang Chen
Committee Member

July 2017

ABSTRACT

Irish, Rodger, Using Pedagogical Tools to Help Hispanics Be Successful in Computer Science. Master of Science (MS), July 2017, 48 pp., 4 tables, 2 figures, references 50 titles.

Computer science (CS) jobs are a growing field and pay a living wage, but the Hispanics are underrepresented in this field. This project seeks to give an overview of several contributing factors to this problem. It will then explore some possible solutions to this problem and how a combination of some tools (teaching methods) can create the best possible outcome. It is my belief that this approach can produce successful Hispanics to fill the needed jobs in the CS field.

Then the project will test its hypothesis. I will discuss the tools used to measure progress both in the affective and the cognitive domains. I will show how the decision to run a Computer Club was reached and the results of the research.

The conclusion will summarize the results and tell of future research that still needs to be done.

DEDICATION

The completion of my master's degree would not have been possible without the love and support of my family. My wife, Reina Irish, and my son, Tim, wholeheartedly inspired, motivated and supported me by all means to accomplish this degree. Thank you for your love and patience. I am also grateful for those who encouraged and prayed for me when I couldn't see the light at the end of the tunnel.

ACKNOWLEDGMENTS

I will always be grateful to Dr. Andres Figueroa, chair of my thesis committee, for all his mentoring and advice. As we both went through the process of proposal writing to IRB approval, and forming a plan for carrying out the research, he encouraged me to complete this process through his infinite patience and guidance. I also want to pay special tribute to Dr. Angela Chapman on my committee. Whenever I needed help on formulating the proposal, or designing the survey and interpreting its results, she was willing to help put thoughts into words. My thanks go to my thesis committee members: Dr. John Abraham, and Dr. Zhixiang Chen. Their advice, input, and comments on my thesis helped to ensure the quality of my intellectual work.

Also, I would like to acknowledge the many volunteers who participated in the focus group research and to Rio Grande City High School for allowing me to hold the Computer Club.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND BACKGROUND

There is a shortfall of graduating programmers in the United States. There is also a lack of diversity in those that are graduating. Indeed, by 2020 there is a projected shortfall of 98,000 jobs per year [7]. Part of the reason for these conditions is that women and minorities, especially Hispanics and African Americans are not seeking computer science (CS) as a career. This is seen in the following statistics. Women make up 57% of undergrads, but only 18%, of CS majors [13]. Hispanics are a fast growing group, but their portion of CS majors has actually fallen to 7.4% [34]. There have been many papers examining either one reason why there is this shortfall, or one theoretical solution. Some of these studies included: Preston talked about pair programming [4]; Becker advocated introducing Karel the robot [5]; and Paralic and Pietrikova suggested having the students create games as they learn programming [6]. Such is the method science has to define and test theories one variable at a time. I wanted to present a more comprehensive solution.

This chapter seeks to give an overview of the entire problem and how some of the already existing solutions may be combined to create the best possible outcome. I will examine some of the main reasons minorities are underrepresented and their possible solutions. Then, I will look at several of the computing languages that have been used throughout history to teach programming. Finally, I will share my vision in putting together several of these tools to make a

comprehensive solution. It is my belief that this solution can produce successful women and minorities to fill the needed jobs in the CS field.

Analysis of the Problem

There are two general reasons why there is lack of minorities and women in computer science. The first reason is a lack of interest. The second is a lack of preparation. I will explore these reasons as well as seek some suggestions to resolve them.

For women and minorities, a lack of interest in pursuing CS as a career comes from a variety of roots. One of these roots is that there is a lack of role models who are women or minorities for future programmers to follow [1]. Everyone knows about Steve Jobs, Bill Gates, and Mark Zuckerburg, but no one knows about Jose Perez or Patricia Moore (friends of mine in the field). (Grace Hopper and Francis Allen are exceptions as role models for women.) In Silicon Valley, less than 1% of the workers are from Hispanic origins [3]. This may seem like a paltry reason, but history tells us of the importance of role models in the recruiting of future programmers. When Neil Armstrong landed on the moon, almost all boys were dreaming about exploring space and becoming rocket scientists. This dream pushed some into that field and encouraged those who were pursuing that dream. When Michael Jordan was winning championships with the Bulls, boys, especially African-Americans, spent all their free time practicing so they could become the next Michael Jordan. Thirty years later, potential NBA stars are still being compared to him. When Bill Gates made it big with Windows, boys who were considered nerds all bought computer kits and tried to become programmers. These are examples from events around my lifetime where I or many of those around me were influenced by these people.

A second root for a lack of interest in entering the programming field is that it seems like it is too hard, or it is only for geniuses. Women might feel that they do not have the skill set to program [15]. Many minorities might feel they cannot commit to a four year college degree due to the perceived rigor or financial difficulties [29]. A large portion of minorities are brought up in poverty and see a four year degree as something for other people. In my high school, most of the students feel they can get a one year certificate for doing a skilled trade, but they do not consider getting a four year college degree as possible.

Another possible root for a lack of interest is a lack of compelling problems in learning programming. Research has shown that if the students were asked to construct a program to remedy a socially significant problem, then women and Hispanics would see the value of programming [29][15]. Young people across the board no longer want to just invent a "Hello World" program, but want something fun. Many see a group of disparate programs about manipulating a string or working a formula as boring and having no relevance to their lives.

There also is a lack of early opportunities for potential programmers. Even though Computer Science is a growing field, many high schools do not offer a class on programming. In fact, there has been a decline in the number of students taking these classes of over 20% [34]. Often, students who have to take a programming class as part of their curriculum in college have said that they might have considered programming, but they were already too far into their current choice of major [21].

A final root of lack of interest stems from the culture not seeing programming as relevant. Especially in minority groups, there is a view of computer programming that says it is okay for a hobby, but it is not a career that can offer stability and advancement [1]. Even though computer

science has helped to make some of the world's richest people and organizations, many do not seem interested in allowing others to pursue a career in this field.

The second main reason many students do attempt computer science classes or drop out is due to a lack of preparation. This is basically seen as a lack of math skills. In fact, Reilly and Tomai found that 45% of those who only had a college algebra background either dropped out or failed CS1 [22]. However, it can also be more generally applied to a lack of problem solving or logic skills.

Solutions to These 2 Problems

The fact that there are few famous role models for minorities or women who want to get into programming does make it a less visible option for young people [1]. Nevertheless, no one says that one of our students cannot become the first role model for their group. To help with this, students can have mentors from the local community who can tell them about what programming is like and help guide them [26]. This can build a pride in wanting to become a programmer. Another solution to this would be to give the students an authentic experience like Bridge to College. This is a camp or seminar where the students experience programming and meet with leaders in the field. In this way, they get a taste of programming, and lets the students hear how leaders got their start. This can help break the stigma of CS being too difficult or having a lack of relevance also [27]. Many classroom teachers might not have the resources to be able to accomplish the above solutions. On alternative that I used was to have a time to talk about some of the leaders in technology and how they got their starts.

As for the root of not being able to get a four year degree or that programming is only for geniuses or nerds there are several tools to help overcome this. Let us start with having good

guidance counselling in high school [2]. Students should be given career assessment tests early in their secondary education. This will help spot students with interests or aptitudes for programming. The counselor should make sure the student is well-prepare before entering a programming class (see paragraph on pre-requisites). The advisement should also include seeing the relevance of programming, career potential and the necessary education involved in obtaining those careers. Some of the facts that can be shared with students are that CS is part of the core curriculum for many degrees. Many might be able to imagine that engineers need at least some background in programing, but other careers like business also call for at least one programming class. Students should be shown that a four year degree is good, and does offer the most opportunities, but there are other paths into computer science as well. There are 1 year certificates and 2 year associate degrees that can open the doors for jobs where they can earn a living. For example, a 2 year web designer degree has an average starting salary of $60,000 and an equivalent computer support can start with a salary of $61,000. Of course programmers ($77,000) and software engineers (120,000) make more with their respective bachelor's and master's degrees [35]. Much of this also falls outside the scope of a classroom teacher, but posting this information and reinforcing it to the students they are able to reach may help.

This leads into the next solution. More high schools need to offer programming classes. One way to do this is to try to get the local mentors and businesses to express this need to high school administrators. If the superintendents and principals believe in its importance, they will find students to take these classes. Once a non-computer science major takes their required programming class in college, it is often too late for that person to switch careers if they feel successful [21]. Another reason for high schools to push programming classes is that there is only a limited time that Gates and Zuckerburg will inspire youths to seek to become self-taught

hobbyists. CS needs to be seen as a main-stream career. Again, this is outside of what many classroom teachers have resources.

The perception that computer science as not a compelling career needs to be attacked on two fronts. First, the material needs to be more engaging. Making the lessons have their own inspiration helps keep students motivated. Writing a program to find the area of a room to be painted or how much tax will be charged is okay for practicing certain programming functions, but they lack any relevance to the student programmer. For women and Hispanic students setting the problems in a socially significant way can give the students a sense of satisfaction [3]. For example, in programming formulas to give the user results, a programmer could incorporate a formula that tells the owner of a pet rescue how much dog food to feed to different sized dogs.

Also, youth always enjoy games. An easy way to get students into the programs they are writing is to have them write programs they can play with when they are done [19]. Here are some examples one could use in a beginning CS class. Use a Mad Libs like game to teach input/output (I/O). Make a dice game to teach random numbers and control statements. Tic-tac-toe uses arrays. Tetris can introduce graphics. Any game that demonstrates the skills learned in the chapter but do not require going beyond that is fine. I have seen papers where teachers mention hangman, and Sokoban. The games I chose to use are Mad Libs, Number guess, Dice, and Tic-Tac-Toe for the first semester.

Secondly, the teaching methods need to be more engaging. Class should be more than a teacher just reading PowerPoint slides. Computing is hands on. Although I, as a math geek, don't mind working alone so that I can make all the decisions, it is also a group activity. There are many types of grouping. A traditional group of 4 or 5 students with a leader, a recorder, and various titles for the other members may work well in an English class. In programming though,

it has a lot of waste because really only 1 or 2 are doing the work. Even open grouping where everyone in the group is equal turns out to be the same thing; only 1 or 2 do the work.

A grouping technique that seems to work is called pair programming. In pair programming, two students work together (remember 1 or 2 do the work so little waste). One is the "driver" and uses the computer to type code. The second student is the "advisor" who is observing to give corrections and alternate solutions [30]. This technique necessitates that the instructor design projects that take two people. It also works best if there is at least some degree of friendship between the two before the project starts.

This technique helps solve many problems for our target groups. The increase in socialization is a boon for women and African Americans who otherwise think of programming as solitary and lonely. Having a pair has built in support. This is important for all minority groups because the affirmation of having two say the same thing gives confidence [20]. Results show several areas of improvement. There is a deeper understanding of the project and a higher enjoyment factor by the pair. The pair spends less time coding, but the code is more optimized than if the students were working individually. These factors contribute to higher course completion rates [20]. I will be using pair programming most of the time in my class.

Looking at the second problem – lack of preparation, there are a number of factors that can contribute to eliminating it. First of all, I already mentioned guidance counselors. These counselors should make sure that students take the necessary pre-requisites. I mentioned math as a significant pre-requisite. Math is integral to programming especially having a strong algebra skill set. On the one hand, higher mathematics like trigonometry and calculus can enhance one's ability to program, on the other hand, logic and problem solving could also meet much of that need [7]. I would propose that successfully navigating a proof based geometry, a logic, or a

rhetoric class would also help the students overcome most problem solving situations. Secondly, offering only advanced AP or dual enrollment classes will not work, but offering the students the opportunity take an introductory class will prepare them for the rigor of AP [31]. This class might incorporate a "robotic" language (this will be discussed below) or some other learning language, but it would give students more time to understand the basics. Finally, I would pair the woman or minority students with a mentor. This mentor could be there when the student gets frustrated and be able to help him/ her refocus and not give up. Once the program is running for a couple of years, the mentors will be students who are in high level classes. Until then, the teacher has to be everyone's mentor to the best of his/ her ability.

Choosing the Programming Languages

Ever since the beginning of computers, people have been trying to create easier ways to communicate with them and to teach others how to use them. At first, one had to use punch cards, then machine language, then assembly language, and finally high level languages. Some of these languages have been created for beginners or for use in educational settings. There are also a category of languages I call "robotic" languages [24]. These are languages that use immediate feedback so students can see what their code is doing. They employ either an avatar of a robot, or an actual robot.

Basic was developed in the 1960's and is an acronym for Beginner's All-purpose Symbolic Instruction Code [36]. This was a procedural language with English like commands and was meant to be an educational language. Basic has developed into .NET VisualBasic. It can now be used as either a procedural or object oriented language and with Visual Studio, it allows for easy GUI creation for professional programs. Its integration into MS Office also

makes it essential for business to have someone who can program in it. I chose not to use Basic because of two reasons. First, it can only be used with Windows operating systems. Secondly, it uses indentations instead of braces for grouping and Dim to introduce a variable. These differences cause a learning curve when transitioning to other languages.

A competitor to Basic was Pascal. It was designed in 1970 strictly as an educational language. However, Pascal has mostly disappeared and it was not designed for industry use.

Another language called Python was designed in the 1990's to fill an on the job need [37]. Yet, its simple syntax and high level made it great for beginners. While it is popular in education, it also has a growing importance in job markets. Still, it also has several problems with using it as an introductory language. Like Basic it uses indentations instead of braces for groupings. It also only has vectors and not arrays. Finally, it is an interpreted language which means it is not compiled once and is set. It gets compiled every time you want to run it.

These languages are great for introducing students to programming because they are not quite as strict or and syntax heavy as more traditional languages like C, C++, and Java. Java is the language I will be teaching my students. It also has faults. It is difficult to learn the basics; even to run simple programs requires skills like classes and instances. It also is an interpreted language. In spite of this, Java is very widely used around the world. In fact, it is the language the state of Texas recognized for high school programming. It is also used in Texas UIL competitions and on the national AP exam.

Alongside these languages, "robotic" languages also appeared. The first of these to enter the field was LOGO in 1967 [8], which used an on-screen turtle to obey commands typed from the programmer. The turtle would leave a trail and the programmer would make designs from

that trail. However, LOGO is its own language, not connected to others. It works great for teaching commands and loops, but there is little else to do with it.

Then in 1981, Karel the robot was introduced. It was originally based on Pascal, but Java, C++, and Python versions also exist [5]. Karel was essentially a red, triangular cursor on a grid. Now it looks like a robot with a red arrow on the stomach. The programmer could send the following commands: move forward, turn left, pick up/ put down beacon, and detect beacon. Any other command would be a function made from these [12]. With Karel, the programmer must type in the commands using the appropriate syntax. It comes with a maze that the robot can navigate. It also allows for instances of classes; therefore, many "robots" could be on the screen acting independently of one another. This allows for extreme scalability. Unlike other "robotic" languages like Scratch, one must actually type in the commands and follow a syntax. One of the best things about Karel is since it is adapted to several languages, switching from it to a traditional language requires no additional relearning [28]. One college of note that uses Karel is Stanford. This is what I will be using.

Lego Mindstorms is a true robotic language because you build a robot using Lego parts. Just like one snaps pieces of Lego together to build the robot, one snaps together programming blocks to make the robot do what is planned. It uses the following categorization of code blocks. Green is for actions like move, turn, etc. Orange is flow control like loops and conditionals. Yellow is for input from the sensors. Red is data operation, which means variables. Allowing variables makes Mindstorms a full functioning language [18]. Programming in Mindstorms is done by connecting (snapping) the appropriate blocks on the screen in the order desired to try to get the robot to do what was intended. This teaches algorithm design, programming constructs,

and I/O [17]. A top university that uses Mindstorms is MIT. Unfortunately, each robot costs over $200; this puts it out of my reach.

The final language in this group is Scratch. Scratch was based on a language called Squeak [24]. It is multiplatform in that it runs in a browser with Flash. It has come a long way since its inception. It is no longer only controlling a cat on the screen. Now there are untold numbers of sprites. Users can use Scratch to make games, animations, and stories. While teaching programming constructs and logic, it avoids syntax. It does this by using block shapes based on their functions. For example, it uses a C shape to denote loops [11]. MIT and Harvard have been involved with this project. Kids like this, and it does teach programming concepts. But, it is not practical, and it does not translate to other languages.

My Vision to Fix the Problem

My vision is to develop 3 classes. These include an Introduction to Programming, Programming 1, and Advanced Programming. I want to develop projects for each of these classes that meet mass appeal. This means have projects that include some games and some where the programmer solves a social problem. I plan on using Pair Programming to help bolster the students' confidence and where possible match them with mentorships to current programmers. Throughout the courses, I will also tell them of leaders in the field and new accomplishments in order to keep them encouraged and motivated.

The Introduction to Programming class would use JKarel but maybe start off with Scratch if time permits. Some of the topics covered would be conditionals, loops, objects, and classes. Flowcharting to help develop logic and program planning would be taught to help

develop logic. Other tools to help teach logic would include truth tables, deduction, rhetoric, logic games, and designing algorithms.

The Programming 1 class would use Java. I would teach 4 chapters and at the end of each chapter the students would design a game instead of taking a test. Chapter one would teach basics like remarks, Input, Output, variables, arithmetic and string manipulation. The game would be Mad Libs. Chapter two would cover relational, Boolean, logical and conditional operators, plus random numbers. The game would be a number guess game. Chapter 3 teaches loops and its game would be a dice game. Chapter four would cover arrays, sorting, and file manipulation. Its game is Tic-tac-toe.

The advanced Programming class would use Java and prepare the students for the AP exam. The primary difference is that it would cover objects, classes, and structures as well as other topics, which are tested on the AP test. The tests here would be based on building a real world application that could help out a community program. It would be a capstone project.

My vision is that by putting together several tools and methods, schools can help women and minorities successfully enter the CS field. As a nation, we need to act or risk computer innovations moving out of the country. Women, Hispanics, African-Americans, etc. need to be in the nation's pool of programmers to afford the broadest possible set of solutions. By adding a beginner's class and using projects that interest everyone, our country can have a diverse, successful group of programmers.

CHAPTER II

TESTING TOOLS

To test my hypothesis that not only will students learn the material quicker and in a deeper way, they will also be influenced to pursue computer science (CS) as a career. This means I have to have a tool to test their cognitive ability with regard to programming, but I also need a tool to test their attitude in various attributes in regards to pursuing CS as a career.

Once I have the surveys and the tests from the students, I will redact their names and assign a Pedagogical Tools in Computer Science (PTCS) number. For the academic test, I will grade them and see if their score improves or not. For the attitudinal test, I will input the results into an Excel spreadsheet, and use the capabilities of the program to collate their scores for each of the attributes. In this way, I should be able to see if there were any changes of significant difference.

Finding a tool to test knowledge of programming is easy. Any comprehensive test that covers the first year of programming would be fine. Since many classes are based around the AP exam (and the fact that our school has a book with released AP tests), I used the first part of a released AP test for both my pre-test and my post-test. To see how much each student

To test the affective significance of my plan, I needed a survey that used the following attributes: Interest in pursuing computer science (CS) as a career, Self-efficacy, Relevance of CS in the student's future, Determination, Intrinsic motivation, and Extrinsic motivation. In order to make sure the survey did correctly quantify these attributes, I decided to look for already tested

surveys. I found two that I put together and adapted for computer science. The first one come from a program at the University Transportation Center for Railway Safety (UTCRS) at the University of Texas RGV [38]. The second survey I used was from Eric Wiebe called "Computer science attitude survey."[39] With some help from Dr. Chapman, I combined the 2 surveys and narrowed the survey into a few background questions and 40 questions that would quantify the above attributes. I would use this tool as both a pre-survey and a post-survey.

This survey gives numbers for each answer. 4 means Agree Strongly; 3 means Agree Somewhat; 2 means Disagree Somewhat, and 1 means Disagree Strongly.

To find how much influence this program made, I will find the differences in their answers and compile them according to the different attributes. This will give us a total change for each category.

CHAPTER III

IMPLEMENTATION

Plans A through D

The implementation of my plan was not without its problems. I got the idea of doing a thesis on the pedagogy of computer science after an assistant principal at my high school told me I would probably be teaching programming the following year. I know that for a truly scientific test of a hypothesis one needs a control group as well as an experimental group. My high school had a shakeup of administrators; we ended up with a new principal the following year, and the assistant principal who had given me the hope of teaching programming was moved into a principalship in another school. The new principal explained to me that our school did not have enough math teachers as it was and that he definitely couldn't give me two classes of programming.

Plan A was gone. Therefore I came up with Plan B. I would find programming teachers in another school district and try to convince them to follow my model in one of their computer classes. As a teacher can understand wanting to teach students my way and not be dictated to by someone who wasn't even there. I wrote to several teachers in the Rio Grande Valley, but needless to say I did not hear back from them.

Plan C was to hold a "Computer Camp." I proposed holding and intensive camp during either Thanksgiving or Christmas break where in three days, I would try to cover the first 6

weeks of lessons. However, I was informed that the school would not be open and that transportation could not be provided. As our school is a rural school and many students do not have their licenses, this option would not work.

On to Plan D, this involved running a computer club after school. I was previously avoiding this situation for several reasons. First, I already had to sign up for two classes for both semesters and knew just those would push my limits. Secondly, most students do not like to stay after school unless they have to. For example, band students stayed after because it was part of their grade for their class. Thirdly, the transportation issue for our students is real, and even the tutorial program was not started yet due to a lack of funding for busing.

In December, the funding was finally approved for bussing after school. I decided to go ahead with the plan. I got permission from my principal. I found out clubs have to be organized so I wrote a charter for the computer club (referred to as the Club from here on out). To prepare the computer lab, I got the administrator password for a lab and installed publically available programs including JKarel, Java jdk, NetBeans, and Eclipse. I also uploaded a book that I convinced the previous administration to buy called *Blue Pelican Java*[40]. I made and distributed a poster and started recruiting students. I was encouraged that before Christmas break, I had 15 students show an interest in the club.

The Club was to meet two 1.5 hour periods after school each week for about 12 weeks. Unfortunately this is only possible in ideal conditions. It turns out to start that I had about 8 different students attending during the week. Some would first go get their free "supper" from the school; this took about ½ an hour. Others could only come for the first ½ hour. Some came on Tuesday, others came on Wednesday. Then EOC/ STAAR testing came, and the IT department re-imaged the computers without the tools to use Java. I moved the Club to my

classroom and had them take turns using my laptop to code altogether. Eventually I scrounged up two more working computers and put them in my room, but by this time I only had 3 regular students coming. Even though I was still going forward with the lessons, I lost the rest of the students when the school district stopped the bussing when I was only halfway through the lessons. I was fortunate to get all three of the students who attended most to fill out the post-surveys, but only two of them returned the post-test.

The Lessons

I started the lessons by showing the students how to access Scratch (41). I showed the students how to move the cat around, use the When clicked event, create a loop, and change what the Sprite (the cat) looked like. Then I let them have independent time where they were supposed to create something interesting. They had a great time exploring, but it served it dual purpose after just the first week. Those purposes were to create excitement and interest in programming and to allow the students to take the pre-test and pre-survey.

The next week I started with JKarel. I showed them how to load up a pre-configured maze and showed them how to type in the commands to move JKarel in the maze. There are only 5 commands to learn, but in typing them, the students learn the syntax of having the program run a method on an object including the ubiquitous semicolon. They had to first define the class and its main method. They had to initialize an instance of the "robot," and they had to put braces to start and end each section. I also taught them about adding comments to the code.

In the next class I taught them how to make their own methods to give JKarel more abilities, like the ability to turn right. They liked this much better instead of typing in jkarel.turnleft(); three times (copy and paste were disabled). The assignment was to navigate the

sample maze and pick up the three "beepers." The last lesson in JKarel dealt with using a new robot with more abilities that included the abilities of loops and if else conditionals. The assignment was a more complicated maze with more beepers.

Although the students were enjoying manipulating the robot and getting used to the syntax of object-oriented programming, I had to keep the program moving. As I say lesson, may I insert a reminder that students were either coming Tuesdays or Wednesdays; therefore each lesson basically took a week.

The rest of the lessons deal strictly with traditional Java. I had to show them how to start a class and then all how what to type for Java's Main method. They also were introduced to importing classes at the beginning of each class and how to use comments. The first lesson dealt with input/ output (I/O) and was simply a hello world style program using *println*. They had to practice adding their own statements to get printed.

The second lesson taught *print* and *printf* statements. I discussed the three main types of variables and how to declare and store values in them. The practice was to redo their statements from the last lesson using *printf*, *\n*, and *%s*. I also demonstrated how you can have a number before the % sign to help format how much room it will take.

The third lesson dealt with input. Thanks to already knowing how to initiate a robot in JKarel, they were able to quickly comprehend how to initiate a Scanner to read input into a variable. Now the class could start getting ready for its first game. Unfortunately, this is when the lab was closed.

The fourth lesson was to create a Mad Libs style game. I gave each student a story with a lot of blanks on it. I talked them through the different parts of the program they would need. They would need to import the classes for I/O, have a main method, and come up with variables

for each of the blank lines. They would also need interactions asking the user to input certain types of words or numbers, and then later displaying them inserted as part of the whole story. Since I was down to just three students at this time, the strongest programmer was actually the one who showed up the most. I had him be the navigator and let the other two be the drivers. This program, while simple, did take a lot of typing. It probably took the students two weeks of Club meetings to get everything typed. One of the requirements was to use the *printf* command and make sure the story lined up nicely on the display.

The next subject the class tackled was control statements. I talked about the symbols <=, <, >, and >=. Then I discussed about how = and == are different in programming. Then, they learned about the *if* and *if else* statements, and how to use braces to show which lines go with which statements. The students practiced these concepts by printing different statements that depended on which grade that was input into the program. They also learned how to make a random number generator and assign its value to a variable.

The game they did for this section is called guess the number. The only thing is that the students still have not been introduced to loops. Therefore, they had to come up with a way for the user to guess at least three times with different statements occurring after each time the user guessed wrong.

Once they got that game finished, I asked the students if they wished there were not a way to make it less typing. I showed them how they could have made a new method, and call that method whenever the user was going to guess a number. This reminded them of how they used to make new methods for JKarel and so it made sense to them. They re-wrote the program to take advantage of methods. I was going to show them for loops in the next lesson, but during

this lesson came a call over the intercom that this would be the last day of busing. The Computer Club was over.

Just to wrap everything up and get the post- survey and post –test taken care of, I had to hunt them down. Even then, I had to keep reminding them to finish and turn in the test and survey. It was not how I wanted to finish because the three that were coming had a lot of potential and I was afraid that they would not be able to continue receiving lessons in computer programming since I'm hoping to have a job in a community college soon.

At the end of the year many students came and told me they heard about the Club. They were not able to take part of it this year, but they were hoping it would be offered next year.

CHAPTER IV

INTERPRETING THE RESULTS

Although there were not enough students to achieve a large significance and the program did not get to run until completeness, I will attempt to interpret the results according to the data I compiled.

I did not compare the differences in the students' background information because that information does not change, even though there were some answers that were slightly different. I interpreted this as their feelings of recent events changed their perspectives. For example, being strongly encouraged or somewhat encouraged can change if a teacher said that their input was wrong that day.

100% of my students were Hispanic; there were two boys and one girl. Only one had a parent go to college, and only one is very sure of going to college even though the school encourages them to go constantly. Only of the students had a computer in the home and a USB drive. I provided USB drives to those who didn't have one in order to save individual work.

Although just a small population, the students still represented a wide range of abilities. One was in the AP program, one was mostly in regular classes, and one had some special education classes (though this student had a lot of artistic talent). Before the closing of the computer lab, I had 2 special education students 3 AP students and 3 regulars.

Question one through 3 dealt with their perceptions of grades in general and then for math and science. Most expected to get A' and B's.

From the question #4, most of them answered that they were at least fairly diligent with their class work. To find this, I assigned a number from 1 to 5 based on their answers. I also assigned numbers to their answers on questions #5 on how many people did they turn to for help with math classes. Most of them looked to only 1 source for help. For more information, please see the appendices for the survey questions and the table showing the compiled background information.

The Survey

One of the problems in compiling the following results is that some of the questions were written in the negative, and sometimes the student may not have noticed. For example, question 21 states "Programming is boring." PTCS01 said Strongly Disagree on the pre-survey, but on the post-survey, the student put Strongly Agree. Unfortunately, I do not want to read into what this student put because maybe that was his/ her opinion on that day. Another problem is that PTCS02 and PTCS03 each forgot to answer a question, which mean there was no difference to measure on those questions. This at least does not skew the results.

The survey consists of 40 questions that mixed up the six attributes we are trying to influence. The first category is Career Interest. In other words, are the students interested in pursuing a career in CS? The second is Self-Efficacy, or how sure are they of their ability to program? The third is Relevance. Do the students view the information about computer science relevant to their future lives? Next is Determination. This attribute measures how far the student is willing to push him/ herself to be successful. The last two categories deal with internal and external motivation. These measure what makes the students want to pursue programming.

PTCS01	
Career	3
Self-efficacy	0
Relevance	0
Determination	3
Motivation	0
External	0

Table 1 – Compiled Differences PTCS01

The first thing I did was put the students' answers into an Excel spreadsheet, and found the change from the pre-survey to the post-survey. Then, I had the program sum the differences in each category. As one can see in Table 1, PTCS01 had no change except in Career and Determination. Again, maybe Career would have been 0 if it were not for question 21. The rise in determination shows that the student probably learned that is work involved in programming, but this student was willing to do it. The equal rise in career interest shows that the student might be a little more inclined to pursue CS as a career. But these numbers though positive still only show a slight change.

PTCS02	
Career	-2
Self-efficacy	0
Relevance	-5
Determination	-1
Motivation	-3
External	-3

Table 2 – Compiled Differences PTCS02

Looking at Table 2, one will see that several categories went down for PTCS02. Most of these changes were small. For example, Question 5 was a Relevance question. The answer went from Strongly Agree to Somewhat Agree. This might be interpreted as the student feels computer science will not be as important as once thought. Or it might be that during the pre-

survey, the student didn't feel like taking much time to think about the difference between strongly and somewhat. At any rate, this is exactly the opposite of what this study was hoping to see. Relevance was the only category to make a significant change since the average drop was by about .68. Both internal and extrinsic motivation seemed to fall a little but in reality, this student was the most consistent attendee of all the students and brought the most visitors to see if others wanted to join the Club.

PTCS03	
Career	-1
Self-efficacy	-7
Relevance	-2
Determination	1
Motivation	-1
External	-2

Table 3 – Compiled Differences PTCS03

Table three shows that several categories also went down, but most by only the smallest of amounts. Determination actually increased a little. The big drop, however, came in Self-efficacy. This -7 represents an average drop of 1.4 per question. This was because this student thought that programming was going to be easy. This student did struggle with learning the concepts, but still learned them before the class moved on as a group. Considering that this student had a harder time than the others, I think that the slight rise in determination, and consistent desire to go into computer science speaks volumes for this student's future in computer science.

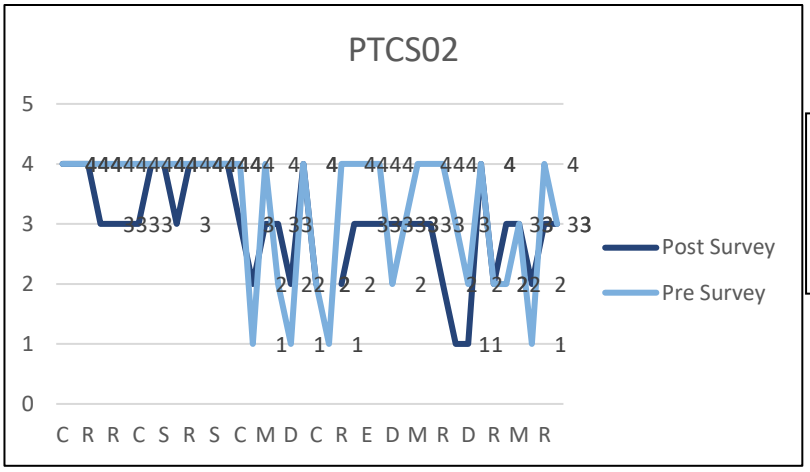


Figure 1 –This graph shows all of the answer choices of PTCS02

Let's look at how PTCS02 answered questions in figure 1 (PTCS01 and PTCS03 also shared this trend). One can see that the pre-survey almost only had 1's and 4's as answers except for 6 questions, but the post-survey includes more answers in the middle than the extremes. This is due to the ceiling effect. The students answered so high on the first survey, in part due to exuberance, in part due to ignorance, that there was no way to go higher. I should have used a second survey that asks how much did the student's attitude change since beginning the program. In this way, no matter what the student put at the beginning, I would not have doubts about the data.

	Mean Pre	Mean Post	Difference of Means
Career	3.19	3.19	0.00
Self efficacy	4.00	3.53	-0.47
Relevance	3.29	2.83	-0.46
Determination	2.63	2.74	0.11
Motivation	3.27	2.93	-0.33
External	3.89	3.67	-0.22

Table 4 – Differences of Means per Category

Table 4 is another look at this data. Since we already know that the data was limited due to the ceiling effect, seeing just how little the average change was will help us to see that fundamental hypotheses are not necessarily proven wrong with this data. The interest in

pursuing computer science as a career stayed exactly the same even if CS as relevant to their future careers dropped by about ½ of a point. If the students thought that CS was not going to be relevant to their careers, they would not have still had the same score for pursuing it as a career. Determination rose slightly even as self-efficacy also dropped by ½ a point and both internal and external motivation also dropped by smaller amounts. If people are not motivated either by motivations within themselves or by some outside influence, they usually would not be very determined, yet determination actually rose. Logically, if one is not very good at doing something, many times that person avoids that task, not becomes more determined to do it. This leads me to believe that either because of only having such a small pool of data or because the instrument needs to be adjusted, but it in no way disproves my original hypothesis that these changes to the pedagogy of teaching programming will help the students become successful in the CS field.

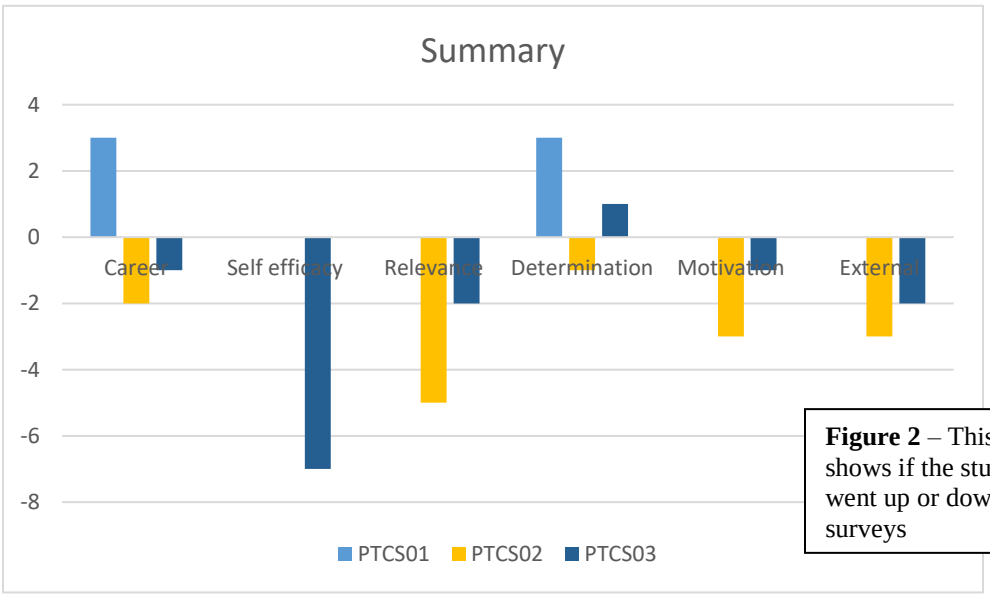


Figure 2 – This graph shows if the students went up or down on the surveys

In summary on the pre-and post- surveys, I believe the students learned a lot of what programming is like, and they all thought it was worth it to pursue programming more. One wanted to be a graphic designer, another wanted to be a game designer, and the last student was

wavering between CS and engineering. Finally, in looking at Figure 2, we do see the students going down in some categories, but for the most part they are minimal. The only significant drop was by as student in self-efficacy, which is probably a more realistic picture of how hard that student will have to work in the future to become successful.

The Test

I will now move on to discuss the academic tool, the pre- and post-test. Remember, this test is usually given at the end of a 36 week course, but the Club did not quite complete week 6. Therefore most of the test contained elements which were never covered. Also, it is a multiple choice test, but rigorous.

Unfortunately, I only have 2 students who completed both parts and turned them in. PTCS01 missed 12 on the pre-test and 14 on the post test. PTCS02 missed 15 and 17 respectively. Again, although these numbers seem to be going the wrong way, I know these students learned the lessons that were covered well. It was probably a mistake to use such a tool once I knew I had only a short-termed Computer Club instead of full-length classes to compare. Most of the questions were beyond the knowledge that was taught. One bad outcome of using multiple choice tests is that it is possible to guess the right answer at times. I have to attribute the better pre-scores to chance. Maybe I should have given them the uncorrected copies of their pre-tests and see which answers they would change.

CHAPTER V

SUMMARY AND FUTURE INVESTIGATIONS

Doing research in the real world with real world people is much different than doing research in a lab. In a lab, one can be change of everything. In the real world, there are many things that might be out of the control of the researcher. Researching people contains even more variables. For a scientific paper to be accurate, bumps in the road would mean scrapping the plan, re-evaluating, and starting again. Unfortunately, my need to graduate by this time dictates that either I find a way to still do the research, or I quit. I am not a quitter.

I realize that I was not able to come to final conclusions on my hypothesis, but I do have some suggestions in how future research con be done, and I can say that those who participated in the Computer Club remain highly motivated to learn programming and to seek a career in, or near that field. This means three Hispanics down, and only 89,997 to go.

As for further investigations, I would like to one day accomplish this study. I truly believe that if teachers can reach these students while they are in high school, then they can be successful as they study for a computer science degree and make an impact in their jobs. However, some changes should be made. First of all, being done as a Club made it hard to have a cohesive cohort. Some would come on time, some late. Some came on day 1, others on day 2. It needs to be tested in a classroom atmosphere.

Secondly, because “real” life came up and interfered with the amount of lessons I was able to teach, I should have adapted the tools. The post-survey should have had questions that

asked how much does one's feelings or opinions change since the beginning of the program. This would eliminate ceilings. The post-test was way above the level of the questions that were taught. They were meant for the end of the year. I should have used a test that reflected more accurately the information that was covered in the time the Club met.

A final change I would make has to do with relevance and motivation. I would start off one class a week with a talking point designed to increase the relevance of CS. Sometimes it was why I was motivated by programming, or that story about the high school that won the underwater robotics competition, but some of the students would not have arrived yet. I think it would have had more impact if all the students were there and we had a time to discuss it or maybe write a little about it in a journal.

Meeting more regularly in itself would great influence the students because then we would make more steady progress through the lessons and the students would have seen their progress especially as the games got more complicated and more fun.

REFERENCES

1. Ashcraft, Catherine; Eger, Elizabeth; Friend, Michelle. “Girls in IT: The Facts.” *National Center for Women & Information Technology*. 2012; http://www.ncwit.org/thefactsgirls.

2. Andujar, Marvin; Aguilera, Lauren; Jimenez, Yerika; Zabe, Farah; and Morreale, Patricia. 2013. “Improving Hispanic High School Student Perceptions of Computing.” In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13).* ACM, New York, NY, USA, 741-741. DOI=http://dx.doi.org/10.1145/2445196.2445448.

3. Backer, P.; Wei, B. “Work in progress-recruiting Hispanic students into computing through community service learning.” *Frontiers in Education Conference (FIE), 2010* IEEE Year: 2010 Pages: F4D-1 - F4D-2, DOI: 10.1109/FIE.2010.5673302.

4. Beck, L. and Chizhik, A. 2013. “Cooperative learning instructional methods for CS1: Design, implementation, and evaluation.” *ACM Trans. Comput. Educ.* 13, 3, Article 10 (August 2013), 21 pages. DOI: http://dx.doi.org/10.1145/2492686.

5. Becker, Byron Weber. "Teaching CS1 with karel the robot in Java." *ACM SIGCSE Bulletin*. Vol. 33. No. 1. ACM, 2001.

6. Brusilovsky, P. et al. “Teaching Programming to Novices: A Review of Approaches and Tools.” *World Conference on Educational Mulitmedia and Hypermedia*. Vancouver, Canada. 1994 Pages 103-110.

7. Christin Landivar, Liana. “Disparities in STEM Employment by Sex, Race, and Hispanic Origin.” *American Community Survey Reports*. United States Census Bureau. Sept 2013.

8. Crum, W.N.; Capobianco, B. “Work in Progress - Collaboration Pedagogy in the Introductory Computer Science Programming Course for Engineers.” *Frontiers in Education, 2005. FIE '05*. Proceedings 35th Annual Conference Year: 2005 Pages: S1E - S1E, DOI: 10.1109/FIE.2005.1612181.

9. DiSalvo, Betsy & Bruckman, Amy. “Education: From Interests to Values.” *Comunication of the ACM*. Vol 54 No 8 August 2011 pgs 27-30. Doi:10.1145/1978542.1978552.

10. Esquinca, A.; Villa, E.Q.; Hampton, E.; Ceberio, M.; Wandermurem, L. “Latinas' resilience and persistence in computer science and engineering: Preliminary findings of a qualitative study examining identity and agency.” *Frontiers in Education Conference*

(FIE), 2015. 32614 2015. IEEE Year: 2015 Pages: 1 - 4, DOI: 10.1109/FIE.2015.7344172.

11. Franklin, Diana; Conrad, Phillip, et al. “Assessment of Computer Science Learning in a Scratch-Based Outreach Program.” 2013. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 371-376. DOI=http://dx.doi.org/10.1145/2445196.2445304.

12. Guoyu Sun; Wenjuan Chen; QingJie Sun; Haiyan Li “Teaching innovation based on robot Karel auxiliary program design.” *Computer Science & Education (ICCSE), 2013* 8th International Conference on Year: 2013 Pages: 1346 - 1349, DOI: 10.1109/ICCSE.2013.6554131.

13. Holland, Nancy. http://www.directemployers.org/2012/08/16/the-college-class-of-2013-current-demographics Aug 16, 2012.

14. Jacobson, D.; Davis, J. See one, do one, teach one . . . two faculty member's path through student-centered learning.” *Frontiers in Education Conference, 1998*. FIE '98. 28th Annual Year: 1998, Volume: 2 Pages: 795 - 799 vol.2, DOI: 10.1109/FIE.1998.738802.

15. Margolis, Jane, Allan Fisher, and Faye Miller. "The anatomy of interest: Women in undergraduate computer science." *Women's Studies Quarterly* 28.1/2 (2000): 104-127.

16. Minjie Li. “Research on the pedagogies of computer science.” *Computer Science and Education (ICCSE)*, 2010 5th International Conference on Year: 2010 Pages: 266 - 270, DOI: 10.1109/ICCSE.2010.5593637.

17. Mota, M.I.G. “Work in progress - Using Lego Mindstorms and Robolab as a Means to Lowering Dropout and Failure Rate in Programming Course.” *Frontiers in Education Conference* - Global Engineering: Knowledge without Borders, Opportunities without Passports, 2007. FIE '07. 37th Annual Year: 2007 Pages: F4A-1 - F4A-2, DOI: 10.1109/FIE.2007.4418124.

18. Nagchaudhuri, A.; Singh, G.; Kaur, M.; George, S. “LEGO robotics products boost student creativity in precollege programs at UMES.” *Frontiers in Education*, 2002. FIE 2002. 32nd Annual Year: 2002, Volume: 3 Pages: S4D-1 - S4D-6 vol.3, DOI: 10.1109/FIE.2002.1158729.

19. Paralic, M.; Pietrikova, E. “Learning by game creation in introductory programming course: 5-Year-long study.” *Emerging eLearning Technologies and Applications (ICETA), 2014* IEEE 12th International Conference on Year: 2014 Pages: 391 - 396, DOI: 10.1109/ICETA.2014.7107617.

20. Preston, David. 2006. “Adapting pair programming pedagogy for use in computer literacy courses”. *J. Comput. Sci. Coll*. 21, 5 (May 2006), 84-93.

21. Redmond, Katie; Evans, Sarah and Sahami, Mehran. 2013. “A large-scale quantitative study of women in computer science at Stanford University.” In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 439-444. DOI=http://dx.doi.org/10.1145/2445196.2445326.

22. Reilly, C. F. and Tomai, E., "An examination of mathematics preparation for and progress through three introductory computer science courses," *Frontiers in Education Conference (FIE)*, 2014 IEEE, Madrid, 2014, pp. 1-9. doi: 10.1109/FIE.2014.7044349.

23. Reilly, C. F.; D La Mora, N./ “The Impact of Real-World Topic Labs on Student Performance in CS1.” *Frontiers in Education Conference (FIE)*, 2012 pgs 1,6,3-6. Oct 2012. Doi: 10.1109/fie.2012.6462329.

24. Resnick, Mitchel; Maloney, John et al. “Scratch: Programming for All.” *Communication of the ACM* Vol 52, No. 11, November 2009 Pgs 60-67. doi:10.1145/1592761.1592779.

25. Rios, David; Chebotko, Artem; Reilly, Christine; Tomai, Emmet, et al. “Improving STEM Education in Research: Preliminary Report on the Development of a Computer-Assisted Student-Mentor Research Community.” *Creative Education.* Vol 3, No. 5, pgs 612-618. 2012. DOI 10.1236/ce,2012.35090.

26. Strayhorn, Terrell L. “Work in Progress – Social Barriers and Supports to Underrepresented Minorities’ *Success in STEM Fields*. 4oth ASEE/IEEE Frontiers in Education Conference. 2010 Pages S1H-1 – S1H5.

27. Tangney, B.; Oldham, E.; Conneely, C.; Barrett, S.; Lawlor, J. “Pedagogy and Processes for a Computer Programming Outreach Workshop—The Bridge to College Model.” *Education, IEEE Transactions on Year:* 2010, Volume: 53, Issue: 1 Pages: 53 - 60, DOI: 10.1109/TE.2009.2023210.

28. Untch, Roland H. “Teaching Programming Using the Karel the Robot Paradigm Realized with a Conventional Language.” On-line at: http://www.mtsu.edu/˜untch/karel/karel90.pdf. Feb 28, 2016.

29. Varma, Roli. 2006. “Making computer science minority-friendly.” *Commun. ACM* 49, 2 (February 2006), 129-134. DOI=http://dx.doi.org/10.1145/1113034.1113041.

30. Werner, Linda; Denner, Jill; Campe, Shannon; Ortiz, Eloy; DeLay, Dawn; Hartl, Amy C. ; and Laursen, Brett. 2013. “Pair programming for middle school students: does friendship influence academic outcomes?” In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. ACM, New York, NY, USA, 421-426. DOI=http://dx.doi.org/10.1145/2445196.2445322.

31. White, Kent; Giguette, Ray. “A three Semester Introductory Computer Science Sequence.” *33rd ASEE/IEEE Frontiers in Education Conference*. 2007 Pages T4C-6 – T4C-9.

32. Williams, L.; Layman, L.; Slaten, K.M.; Berenson, S.B.; Seaman, C. "On the Impact of a Collaborative Pedagogy on African American Millennial Students in Software Engineering." *Software Engineering*, 2007. ICSE 2007. 29th International Conference on Year: 2007 Pages: 677 - 687, DOI: 10.1109/ICSE.2007.58.

33. Woodman, M.; Griffiths, R.; Holland, S.; Robinson, H.; Macgregor, M. "Employing object technology to expose fundamental object concepts." *Technology of Object-Oriented Languages and Systems, 1999*. Proceedings of Year: 1999 Pages: 371 - 383, DOI: 10.1109/TOOLS.1999.779081.

34. www.exploringcs.org/resources/cs-statistics Exploring Computer Science UCLA Graduate School of Education and Information Studies. Feb 28, 2016.

35. Money.usnews.com/careers/ best-jobs/rankings/best-technology-jobs. Feb 22, 2016.

36. http://www.computerhope.com/jargon/b/basic.htm. April 14, 2016.

37. http://python-history.blogspot.com/2009/01/personal-history-part-1-cwi.html. April 14, 2016.

38. UTCRS School STEM Survey, UTRGV, 2016.

39. Wiebe, Eric, et al. "Computer science attitude survey." *computer science* 14.25 (2003): 0-86.

40. Cook, Charles E., *Blue Pelican Java*. Virtualbookworm.com. Refugio, TX 2005.

41. https://scratch.mit.edu/ 6/22/2017

42. Haden, Patricia. 2006. The incredible rainbow spitting chicken: teaching traditional programming skills through games programming. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52* (ACE '06), Denise Tolhurst and Samuel Mann (Eds.), Vol. 52. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 81-89.

43. Felden, Maria, Clua Osvaldo. "Work in Progress: Cultural Borders in CS1." 36th ASEE/IEEE Frontiers in Education Conference M3E-21. San Diego, CA. October 28, 2006. DOI:1-4244-0257-3/06.

44. D. C. Cliburn, "Experiences with the LEGO Mindstorms throughout the Undergraduate Computer Science Curriculum," *Proceedings. Frontiers in Education. 36th Annual Conference*, San Diego, CA, 2006, pp. 1-6.
doi: 10.1109/FIE.2006.322315

45. Scott Leutenegger and Jeffrey Edgington. 2007. A games first approach to teaching introductory programming. *SIGCSE Bull.* 39, 1 (March 2007), 115-118. DOI: https://doi.org/10.1145/1227504.1227352

46. Backer, Patricia R. Wei, B. and Sundrud, J. "Increasing Hispanic Engagement in Computing Through Service Learning" (2011) San Jose State University. Available at: http://works.bepress.com/patricia_backer/27/

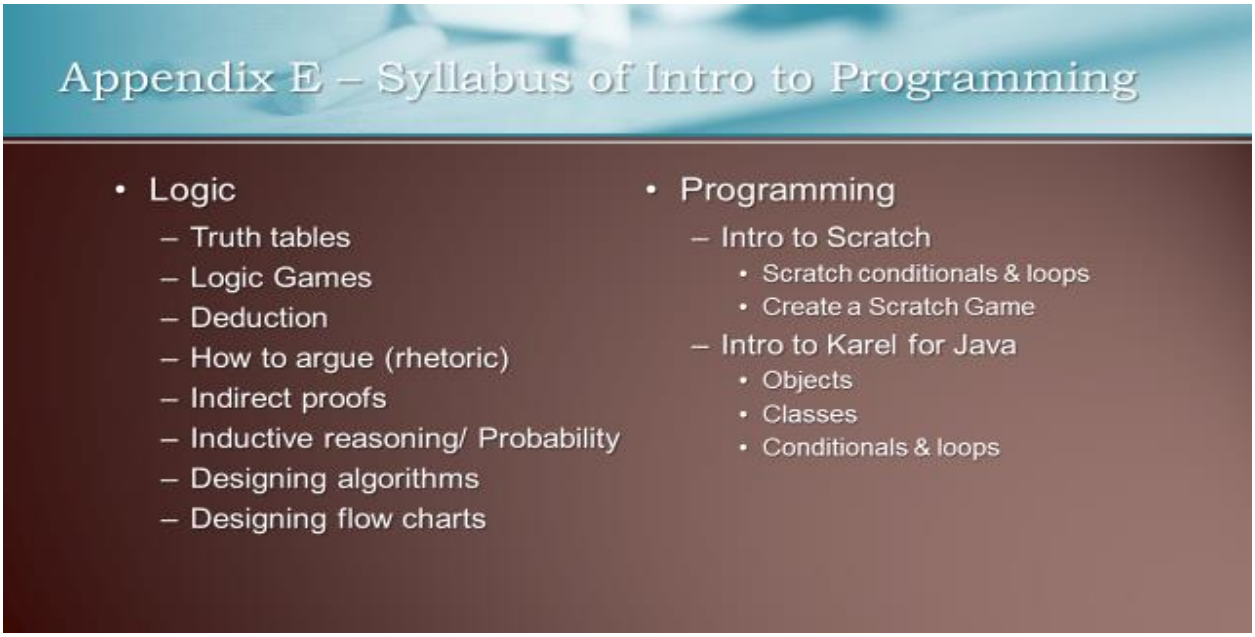
47. Harris, James, Vladan Jovanovic. 2010. "Introductory Programming Course Using Games." *Issues in Information Systems*. source: http://iacis.org/iis/2010/104-113_LV2010_1444.pdf

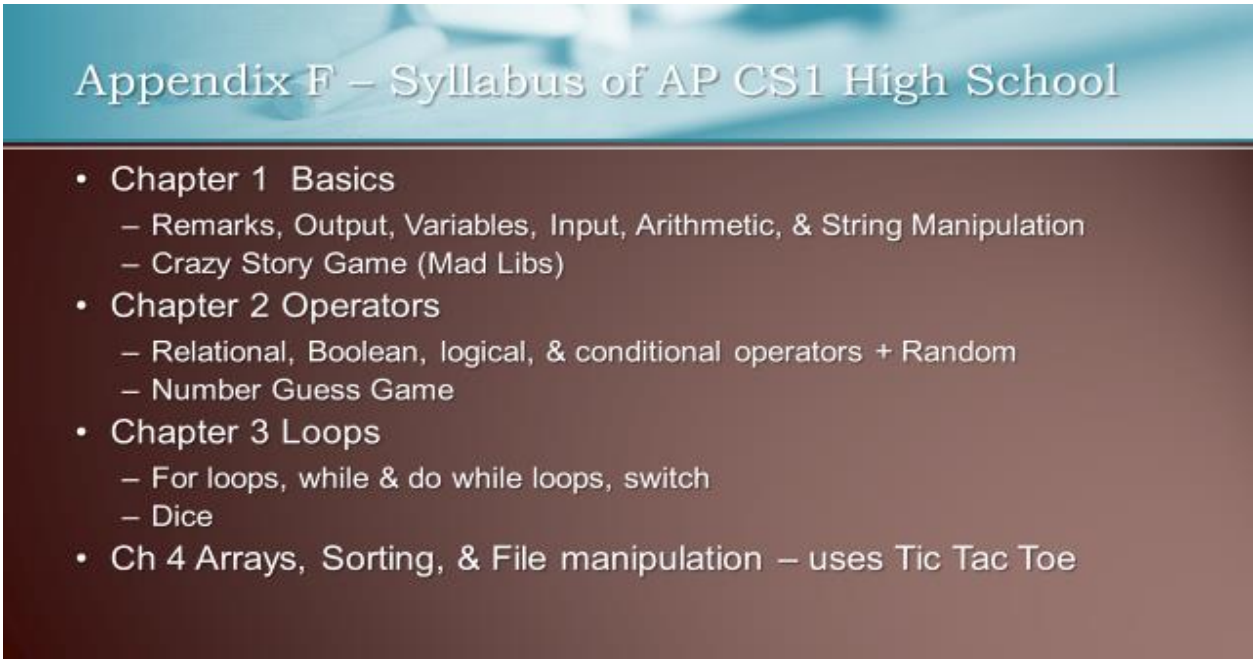
48. Law, Kris MY, Victor CS Lee, and Yuen-Tak Yu. "Learning motivation in e-learning facilitated computer programming courses." Computers & Education 55.1 (2010): 218-228.

49. Whitton, Nicola. "Motivation and computer game based learning." *Proceedings of the Australian Society for Computers in Learning in Tertiary Education, Singapore* (2007).

50. Katrin Becker. 2001. Teaching with games: the Minesweeper and Asteroids experience. *J. Comput. Sci. Coll.* 17, 2 (December 2001), 23-33.

APPENDIX


Appendix E – Syllabus of Intro to Programming
• Logic
– Truth tables
– Logic Games
– Deduction
– How to argue (rhetoric)
– Indirect proofs
– Inductive reasoning/ Probability
– Designing algorithms
– Designing flow charts
• Programming
– Intro to Scratch
• Scratch conditionals & loops
• Create a Scratch Game
– Intro to Karel for Java
• Objects
• Classes
• Conditionals & loops


Appendix F – Syllabus of AP CS1 High School
• Chapter 1 Basics
– Remarks, Output, Variables, Input, Arithmetic, & String Manipulation
– Crazy Story Game (Mad Libs)
• Chapter 2 Operators
– Relational, Boolean, logical, & conditional operators + Random
– Number Guess Game
• Chapter 3 Loops
– For loops, while & do while loops, switch
– Dice
• Ch 4 Arrays, Sorting, & File manipulation – uses Tic Tac Toe

Letter to Principal for Permission and Plan

Mr. Saenz

As you know, I want to teach computer programming. As I'm working very hard to complete my master's this year (I will have 6 credits each semester), I don't have a lot of time. However, after thinking through several ideas to complete my thesis on the pedagogy of computer science, my advisor and I are interested in trying to form an after school computer club that meets a couple of times a week.

I do want to get my feet wet in teaching these skills; however, it won't be quite as structured as a class. I know any, who join the class will have to sign some papers giving me permission to use surveys for my thesis. I also know the club would be competing with many other after school activities. Because there would be no required attendance, the lessons I would teach would have to be self-contained projects.

I would hope that I could recruit between 12 and 20 students. We would need a computer lab. As tutorials are supposed to start soon, transportation would be taken care of. I would need a lab and I could work with Ernie to install the free programs needed to program in Java (the Texas standard for high school.) My time would be free.

I would follow the plan for the 1st semester of the programming class. 1st I would introduce them to jKarel so they can see immediate feedback of sending commands, and at the same time learning algorithms, conditional and loop commands. Then we would build from that to writing simple code using text to discover remarks, variables, input/ output, String manipulation, and errors. We would end each unit by making a game from what we've learned. Other topics the students will cover is debugging, arrays, and using classes.

At the beginning of the club, and then at its end, I would give them a pre and post test and a survey. I will complete an IRB through the University to make sure the students' involvement will follow established guidelines.

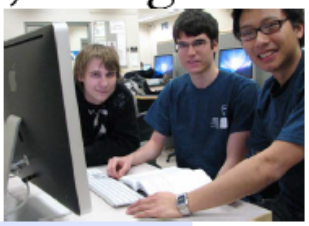
Sincerely,

Rodger Irish

Wanted Programmers

The world needs more programmers!

Join a Club, have fun, and get skills that can pay.



You can do it!! Sign up today

This Club will be used as part of a study to test pedagogical methods by Mr. Irish.

*Prepare for college comp classes

*Projects are to write games

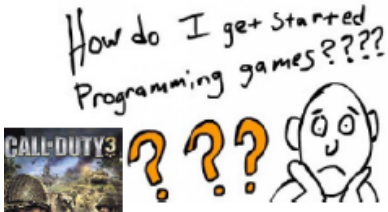
*No previous knowledge needed

*Helps prepare for AP exam

*Prepares for Dual-enrollment

*Optional -- UIL team.

Software Developer....	**$80,500**
Software Manager.....	**$115,00**
Web Developer.....	**$58,000**
Systems Developer.....	**$93,000**
Teacher.....	$40,000
Pipeline worker.....	$43,000
High School Grad.....	$24,000
Drop out.....	$14,000



F118

Charter and Bylaws of the Computer Club

The Computer Club of Rio Grande City

Bylaws

Article 1. Club Name

The name of the computer club is Rio HS Computer Club or Rio Programmer's Club, hereafter referred to as the club.

Article 2. Purpose of the Club

a. Teach programming to any who come to the meetings and promote creative use of those programming skills.
b. Increase awareness in the latest developments in the area of technology including career options in a diverse economy.
c. Provide a positive forum in which students can be creative, innovative, and marketable in their prospective area of interests.
d. Provide a "user friendly" place for students, faculty, staff, and visitors to learn and use the latest computer technology.
e. Promote Rio Grande City CISD through the effective use of a website and community projects.
f. Form a competitive team that can go to various programming events such as UIL competitions.
g. Enhance student knowledge by providing a forum of Questions and Answers and Internet resources through the effective use of a Website and newsletter (optional).

Article 3. Membership

a. Membership is open to any currently enrolled Rio Grande City High School student who has completed an entry exam and survey.
b. Membership is open to any current faculty and staff at Rio Grande City High School.
c. Only currently enrolled students are allowed to run for club offices.
d. Current members may bring up new business during the open floor session of regularly scheduled meetings.

Article 4. Officer Positions

a. The officers of this club shall be President, Vice President, Secretary, Treasurer, Webmaster, and Editor. Officer positions may be deleted or established by a majority vote of the club. All officer positions except President and Secretary are optional.
b. Officers will be elected for one year by a vote of the club at the beginning of each term year.
c. Officers must be currently enrolled Rio Grande City High School sophomores, juniors, and/or seniors and are current on annual dues (except for the first year).
d. Officers cannot miss more than two meetings per month.
e. An officer may be removed from office by a 3/4 vote at a club meeting.

f. If an officer position is becomes vacant for any reason, the club shall hold elections at the next club meeting.

Article 5. Officer Duties

a. President - principal officer and is responsible for leading the club in meetings and activities in accordance with guidance established by the Rio Grande City Consolidated School District and these bylaws.
b. Vice President - shall assist the president in club management, shall preside over club meetings in the absence of the president, and shall perform other duties assigned by the president.
c. Secretary - shall keep minutes of club meetings, maintain club membership records, and shall perform other duties assigned by the president. The secretary shall maintain an attendance roster for the club records.
d. Treasurer - shall maintain all of the financial holdings of the club including maintaining a current balance sheet. The treasurer shall make a financial report to the club on a semi-annual basis or whenever the faculty sponsor or president deems necessary. The treasurer shall make a financial report to the club before the election of a new treasurer or in the event the treasurer leaves office before regular elections.
e. Webmaster - shall create and maintain a dynamic website for the club.
f. Editor - shall be responsible for the editing of the monthly newsletter before the final draft is printed and distributed.

Article 6. Faculty Sponsors

a. There may be a group of faculty sponsors.
b. At least one sponsor shall be a full-time instructor at Rio Grande City High School. Part-time instructors are permitted to be club sponsors.
c. Faculty sponsors are not required to pay annual membership dues.
d. Sponsors may vote in meetings and participate in all club activities.
e. A sponsor must be present at all club meetings.
f. A sponsor will be responsible for supervising elections and maintaining order within the club.
g. A sponsor will work closely with the club officers in business matters of the club, including maintenance of the club website.
h. A sponsor will teach programming lessons after the regular business.

Article 7. Executive Committee

a. The executive committee will be composed of all club officers and faculty sponsors.
b. The executive committee will meet as required to make club decisions that do not need a majority vote of all club members.
c. Meetings will be called as necessary.
d. This committee may recommend the creation of other committees. The additional committees will be created by vote of the club.

Article 8. Voting

a. Each member in good standing may vote.

b. All proposed changes (amendments) to these bylaws must be approved by a majority of the club.

Article 9. Dues

a. The Club Treasurer will collect annual membership dues. The dues are $5.00 for annual membership and are to be paid in advance except for the first year where there will be no dues.
b. Dues may be changed by a majority of the club.
c. Dues paid are non-refundable.
d. The dues will go toward expenses not covered by the funds raised by the club (e.g., food and beverages at meetings).
e. Dues will be deposited into the club's school account.

Article 10. General Fund

a. The club will provide an annual budget designated for startup costs, equipment, activities, events, student travel, guest speakers, and certifications. This fund may cover other costs incurred by the club but cannot be used for the purchase of food and beverages.
b. This fund will be managed by the club and placed into the club's school account.

Article 11. Meetings

a. General meetings will be held biweekly. Meeting locations and times are subject to change by a decision of the sponsor or principal.
b. At least one sponsor is required to attend each meeting.

Article 12. Activities

a. Fundraisers - proceeds from fundraisers will be deposited into the club's school accountant will go toward expenses not covered by the funds provided to the club by club member dues (e.g., food and beverages at meetings).
b. Competitions and Fairs
c. Site Visits
d. Certifications
e. Equipment
f. Software

Article 13. Newsletter (optional)

a. A monthly newsletter will be generated and distributed to all students.
b. The content of the newsletter may include, but is not limited to; club activities, meeting schedule, technology articles, technology help, school activities, etc.
c. The newsletter will also be published on the club website.
d. The newsletter will contain factual information and will not contain personal opinions (e.g., political).

Article 14. Website (optional)

a. The content of the website may include, but is not limited to:
    - Discussion
    - Q&A
    - Links
    - Tutorials
    - Downloads
    - Articles
    - Schedule
    - RGC HS program information
    - Newsletter
b. Items and links on the website will adhere to all Rio Grande City CISD policies.

Article 15. Code of Ethics

a. All members will be responsible for their actions and respect the genius of others' work and property.
b. Software piracy is not allowed.
c. Members shall abide by the student code of conduct published by Rio Grande City CISD.
d. The club shall operate under current school policy.

Article 16. Changes to Bylaws

a. Articles in this set of bylaws may be deleted or modified as deemed necessary by a majority of the club.
b. Changes to the bylaws will be done as amendments.
c. A majority vote is required to make any changes to the club's bylaws.

PRE AND POST SURVEY

1. What is your best guess as to how you did in all of your classes this past year? (Mark one)

☐ Mostly A's ☐ Mostly A's and B's ☐ Mostly B's
☐ Mostly B's and C's ☐ Mostly C's ☐ Mostly below C's
☐ A mix of A's, B's and C's

2. What is your best guess as to your grade in your math this past year? (Mark one)

☐ A ☐ B ☐ C ☐ Below C ☐ I did not take math last year

3. What is your best guess as to your grade in your science class this past year? (Mark one)

☐ A ☐ B ☐ C ☐ Below C ☐ I did not take a science class last year

4. For your math and science classes, how often did you (this past year): (Mark one)

	Always	Most of the time	Sometimes	Never	No opportunity to do this
Do the homework for class	☐	☐	☐	☐	☐
Do work for extra credit	☐	☐	☐	☐	☐
Participate in class discussions	☐	☐	☐	☐	☐
Ask questions in class	☐	☐	☐	☐	☐
Feel bored in class	☐	☐	☐	☐	

5. Have you asked any of the following people for help in your science and math classes this past year? (Mark one in each row)

Students who are doing well in the class	☐Yes	☐No
My close friends	☐Yes	☐No
Parent(s)/Other adults at home	☐Yes	☐No
Brother(s)/Sister(s)	☐Yes	☐No
My science and/or math teacher	☐Yes	☐No
Other teacher(s) or adult(s) at school	☐Yes	☐No

6. How much have your teachers, guidance counselors, and/or other adults at school encourage or discourage you to:

	Strongly encourage	Somewhat encourage	Somewhat discourage	Strongly discourage	We never talk about this.
Go to college	☐	☐	☐	☐	☐
Think about majoring in computer science in college	☐	☐	☐	☐	☐

7. How much do you want to go to college?

☐Very much ☐Somewhat ☐ Only a little
☐Not at all ☐I don't know

8. Have the following people in your family attended college? (Mark one in each row)

Mother, stepmother or both	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
Father, stepfather or both	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more older brother(s)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more older sister(s)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more other adults in my family (legal guardian, grandparent, aunt, uncle, etc.)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either

9. Do the following people in your family have a science, math, technology, or engineering related job?

Mother, stepmother or both	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
Father, stepfather or both	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more older brother(s)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more older sister(s)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either
One or more other adults in my family (legal guardian, grandparent, aunt, uncle, etc.)	☐Yes	☐No	☐ I'm not sure	☐ Don't have either

*taken from UTCRSS School Survey, UTRGV, 2016.

Statement	Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree
1. I plan to major in computer science.	**4**	**3**	**2**	**1**
2. I am sure that I can learn programming.	**4**	**3**	**2**	**1**
3. I study programming because I know how useful it is.	**4**	**3**	**2**	**1**
4. When a programming problem arises that I can't immediately solve, I stick with it until I have the solution.	**4**	**3**	**2**	**1**
5. Understanding computer science will benefit me in my career.	**4**	**3**	**2**	**1**
6. It would make me happy to be recognized as an excellent student in computer science.	**4**	**3**	**2**	**1**
7. Knowing programming will help me earn a living.	**4**	**3**	**2**	**1**
8. Once I start trying to work on a program, I find it hard to stop.	**4**	**3**	**2**	**1**
9. I am confident I will do well on programming labs and projects.	**4**	**3**	**2**	**1**

Statement	Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree
10. Being regarded as smart in computer science would be a great thing.	**4**	**3**	**2**	**1**
11. Computer science is a worthwhile and necessary subject.	**4**	**3**	**2**	**1**
12. I believe I can master computer science knowledge and skills.	**4**	**3**	**2**	**1**
13. I can get good grades in computer science.	**4**	**3**	**2**	**1**
14. I'd be proud to be the outstanding student in computer science.	**4**	**3**	**2**	**1**
15. I'll need a firm mastery of programming for my future work.	**4**	**3**	**2**	**1**
16. I am challenged by programming problems I can't understand immediately.	**4**	**3**	**2**	**1**
17. I am curious about discoveries in computer science.	**4**	**3**	**2**	**1**
18. Programming will not be important to me in my life's work.	**4**	**3**	**2**	**1**
19. The challenge of programming problems does not appeal to me.	**4**	**3**	**2**	**1**
20. I believe I can earn a grade of "A" in computer science.	**4**	**3**	**2**	**1**
21. In terms of my adult life it is not important for me to do well in computer science in college.	**4**	**3**	**2**	**1**
22. Programming is boring.	**4**	**3**	**2**	**1**
23. I enjoy learning programming.	**4**	**3**	**2**	**1**
24. Programming is enjoyable and stimulating to me.	**4**	**3**	**2**	**1**
25. I'd be happy to get top grades in computer science.	**4**	**3**	**2**	**1**
26. I will use programming in many ways throughout my life.	**4**	**3**	**2**	**1**
27. I don't understand how some people can spend so such time on writing programs and seem to enjoy it.	**4**	**3**	**2**	**1**
28. I like to do better than other students on a programming project.	**4**	**3**	**2**	**1**
29. Learning to program is interesting.	**4**	**3**	**2**	**1**
30. I put enough effort into learning programming.	**4**	**3**	**2**	**1**
31. Learning computer science makes my life more meaningful.	**4**	**3**	**2**	**1**
32. I spend a lot of time learning programming.	**4**	**3**	**2**	**1**

Statement	Strongly Agree	Somewhat Agree	Somewhat Disagree	Strongly Disagree
33. For some reason even though I work hard at it, programming seems unusually hard for me.	**4**	**3**	**2**	**1**
34. It would be really great to win a prize in computer science.	**4**	**3**	**2**	**1**
35. Programming is of no relevance to my life.	**4**	**3**	**2**	**1**
36. Taking computer science courses is a waste of time.	**4**	**3**	**2**	**1**
37. I like writing computer programs.	**4**	**3**	**2**	**1**
38. I would rather have someone give me the solution to a difficult programming problem than to have to work it out for myself.	**4**	**3**	**2**	**1**
39. Programming I learn is relevant to my life.	**4**	**3**	**2**	**1**
40. My career will involve programming.	**4**	**3**	**2**	**1**

**Taken from Wiebe, Eric, et al. "Computer science attitude survey." *computer science* 14.25 (2003): 0-86.

COMPILED BACKGROUND DATA

		background info								
	Question	1	2	3	4	5	6	7	8	9
PTCS01	Post	3	5	5	3.5	1	5	5	2	1
PTCS01	Pre	4	4	5	4	1	5	5	2	0
	difference	-1	1	0	-0.5	0	0	0	0	1
PTCS02	Post	6	5	5	4.5	1.5	3	3	3.5	1.5
PTCS02	Pre	7	5	5	4.5	1.5	4.5	4	3.5	1.5
	difference	-1	0	0	0	0	-1.5	-1	0	0
PTCS03	Post	5	4	4	4	1	4.5	1	1	1
PTCS03	Pre	6	4	4	3	1	4.5	1	1	1
	difference	-1	0	0	1	0	0	0	0	0

BIOGRAPHICAL SKETCH

Rodger Irish grew up in Flint, Michigan. Upon graduation, he attended Pensacola Christian College, Cornerstone University, and a couple of others before earning his Bachelors of Arts in 1999. Part of his undergraduate work was done in Ecuador where he learned Spanish. His BA is from the University of Texas-Pan American in history with a minor in mathematics. He earned his MS in Computer Science in July 2017 from University of Texas-Rio Grande Valley.

With his degree and passing the state exams, he became a math teacher and taught in Texas for four years before moving back to Michigan with his bride. In Michigan he became certified in both history and math and taught in an adult program funded by General Motors until the Great Recession.

It was at this time, through a Michigan initiative to retrain workers, that he rediscovered his love of programming. He took classes at a community college before taking a teaching job in Puerto Rico. After a couple of years there, he and his family relocated back to the Rio Grande Valley of Texas and took a job teaching math. This was so that it would be possible to study for his Masters in Computer Science. With this thesis, he will complete my degree and enter my second career.

My hope is to use my degree to be able to teach computer science in community colleges. Otherwise, I will apply to jobs within the industry. My current email is rodgerirish@gmail.com.