2A Delivery

By

Julian Garcia

An Honors Thesis Submitted in Partial Fulfillment
of the Requirements for Graduation from the
Western Oregon University Honors Program

Dr.Becka Morgan,
Thesis Advisor

Dr. Gavin Keulks,
Honors Program Director

Western Oregon University

June 2015

Acknowledgements

I would like to take the time to thank all those who helped me through this thesis project. First I would like to thank my thesis and major advisor, Dr. Becka Morgan, for her facilitation of my ideas, project, and final work. Visiting assistant professor Mitchel Fry was also instrumental in keeping me on task and consistently programming. Also, I would like to thank all my new found friends and family in Oregon without whom I would not have the local support structure I needed to complete this project. Lastly, I would like to thank my family for all their sacrifices that have allowed me to succeed thus far.

Thank you all.

Table of Contents

I. Abstract

For students who want or need to learn how to count in binary and hexadecimal, the 2A Delivery Android application is a teaching tool that will help students learn by playing a game. This game will simulate a career with a fictional delivery company and will encourage the dedication of the student with pay raises, bonuses for great work, upgrades to vehicles, and map expansions. Unlike the current educational games available on the market, our product will use a storyline that will boost learning and stimulate the mind by teaching how to convert between multiple number bases.

II. Introduction

The traditional path for an incoming computer science major at Western Oregon University starts with a broad introduction to the computer science realm. This introduction covers the history of computers, a small look at how computers work, and the potential that computers have. When students are taught about how computers work, two main ideas are covered: binary and logic. Without these two ideas, there is no foundation for computer science. At the end of the traditional path, computer science majors must complete a senior project on a topic of their choosing. I chose to create an educational game that could aid in learning the fundamental building blocks of computer science.

My application, 2A Delivery, is a teaching tool for students who want to learn and practice their binary, decimal, and hexadecimal conversions. I combined this with my love for outer space to create the storyline for 2A Delivery. Students start as a lowly package delivery driver for the 2A Delivery Company, located in the Crathea planetary system. This company must deliver packages from the Tenze planet to the proper address on the Banearth planet. The only problem is that the Tenzians do not understand the address system on Banearth, because it is based in binary rather than decimal, and it is up to the delivery driver to make sure packages go where they need to. The student, acting as the delivery driver, must convert the decimal address to its binary equivalent. As the story progresses, the student should develop the skills to quickly convert binary numbers.

In order to keep a student's interest, a task that is not simple, as they progress they will unlock different features of the game. They will have the opportunity to hire more

delivery drivers, which corresponds to how many lives a player has, upgrade delivery vehicles, which corresponds to how quickly a player can earn a life back, earn larger profits, and expand their business. After expansion, the player will be able to travel to Hexdron, which uses a hexadecimal address system.

III. Background

One of the first academic tasks children learn how to perform is how to count to ten. Most people, after they have grown to adulthood, never look back and ask, "Why only use the numbers zero to nine per digit?" Modern civilization is constantly surrounded by this decimal, also known as base ten, numbering system. Why? Was another base ever taught and utilized? One field of business today is based off of multiple numbering systems. Computer scientists utilize the binary system, base two, and the hexadecimal system, base sixteen.

The decimal system, as those who are taught today know, uses one number from zero to nine for each digit to count. Each digit place represents a power of ten. The first digit is a number between zero and nine times 10^0, which is equal to one, so the number 6 is really

$$6 * 10^0 = 6_d$$

where the subscript 'd' shows the reader that the number is a base ten number. The second digit is a number between zero and nine times 10^1, which is equal to ten, so the number 67 is really

$$6 * 10^1 + 7 * 10^0 = 60 + 7 = 67_d.$$

The third digit is a number between zero and nine times 10^2 and each digit after increases the power of ten by one, following the same pattern.

Unfortunately, the origin of the decimal numbering system is unknown. According to Ian MacFarlane's work, titled *The Enneagram - History of the Decimal*

System, the common theories of where this numbering system originated are China, a creation of Abū ʿAbdallāh Muḥammad ibn Mūsā al-Khwārizmī, who was a Persian mathematician, astronomer, and geographer, or it originated in the Middle East after the Indian numeral system traveled westward. The Indian numerals may have been inspired by the Buddhist pilgrims that brought the Hau Ma numerals to India between 400 and 700. There is evidence, however, that during the seventh century AD, Indian mathematicians were using a base ten numbering system, symbolized by nine numerals but had no zero digit. Using the zero digit for more than just representing nothing, like ten, for example, which is represented by a one followed by a zero, was developed around 874 by Muslim mathematicians. Other ancient cultures, like the Egyptians, Minoans, and Greeks used a base ten numbering system.

Today, the decimal system is widely used. It is used from teaching children their age to currency and scientific measurements. Having ten fingers and ten toes makes learning this numbering system so easy as a child. When a child is asked how old they are, they routinely hold up a certain number of fingers and say, "This many." By the time they are older than the number of fingers on each hand, they have learned how better to express the decimal system. American dollars, European euros, and Mexican pesos are a few currencies that used this system of counting. The International System of Units uses prefixes, like "deca-", meaning 10^1, "hecto-", meaning 10^2, and, more commonly, "kilo-", meaning 10^3, as units of measure. Base ten numbering is the most common system today.

The binary numbering system is based off of two numbers: zero and one. Each digit place represents a power of two. The first digit is a zero or one times 2^0, which is equal to one, so the number 1 in binary can be shown as

$1 * 2^0 = 0b1$

where the preceding "0b" shows the reader that the number is a base two number. The second digit is a zero or one times 2^1, which is equal to 2, so the number 3 in binary can be shown as

$3_d = 1 * 2^1 + 1 * 2^0 = 0b11.$

The third digit is a zero or one times 2^2 and each digit after increases the power of two by one following the same pattern.

To convert between decimal and binary, there is only one step but it is repeated until the last digit, 2^0, is found. First, find the largest power of two that can be subtracted out of the number in question without making it a negative number. Using the number 67_d,

Step 1: $67_d - 2^6 = 67_d - 64_d = 3_d.$

This means that 67_d has a 1 as its seventh digit in its binary representation. Now, repeat step one using the remainder found in the previous step.

Step 2: $3_d - 2^1 = 1_d.$

This means that 67_d has a 1 in its seventh digit and a 1 in its second digit. Repeat this step again using the remainder from the previous step.

Step 3: $1_d - 2^0 = 0_d.$

This means that 67_d has a 1 in its seventh digit, a 1 in its second digit, and a 1 in its first digit. 2^0 is the last digit to be found so this means that the full binary representation of 67_d becomes

$67_d = 0b1000011$.

The base two numbering system can also be used to perform regular math functions. The rules and techniques that are used for addition, subtraction, multiplication, and division are the same for binary as they are for decimal. For example, some simple addition and subtraction problems are as follows:

5	101	5	101
+ 3	+011	- 3	-011
8_d	1000_b	2_d	10_b

Multiplication becomes drastically more straightforward in binary. In elementary school, children are pushed to learn their multiplication tables. A child is expected to memorize one hundred and forty-four sets of multipliers and answers when they learn multiplication tables up to twelve multiplied by twelve. The binary system only requires the memorization of four different sets of multipliers and answers:

0*0 = 0

0*1 = 0

1*0 = 0

1*1 = 1.

For example, a simple multiplication problem is as follows:

5	101
x 3	x011
15_d	101
	+1010
	1111_b

Once this is understood and mastered, binary division can be understood with the proper effort just as decimal division can.

The modern binary number system is attributed to the work of Gottfried Leibniz, a German mathematician and philosopher, who lived from 1646 to 1716 (O'Connor, J. & Robertson, E., 1998). Through his development of binary, Leibniz was introduced to ancient Chinese figures in the *Figure of Eight Cova*, developed by Fu Xi who is believed to have lived more than four thousand years ago, by the French Jesuit Reverend Father Bouver. These figures were a series of vertical lines, as seen on the far left here:

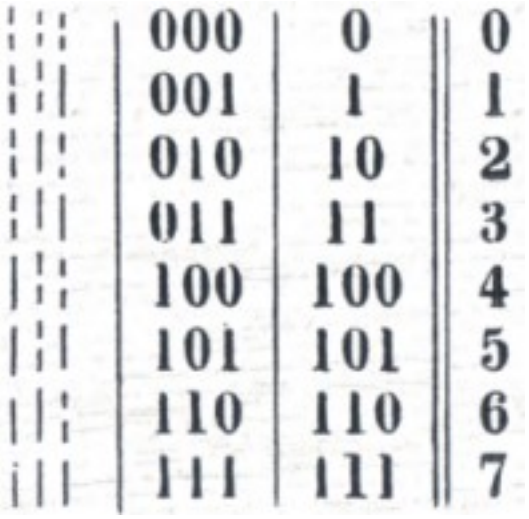


Figure 1 (*binary6.jpg (JPEG Image, 197 × 193 pixels)*, n.d.)

Unfortunately, the Chinese lost the meaning of the *Figure of Eight Cova* around a thousand years ago (Strickland, L., 2007). When Leibniz applied his idea of binary to the lines, he noticed that the unbroken lines correspond to zero and the broken lines to one. As a religious man, Leibniz described this revelation as

> "[A concept that] is not easy to impart to the pagans, is the creation ex nihilo through God's almighty power. Now one can say that nothing in the world can better present and demonstrate this power than the origin of numbers, as it is presented here through the simple and unadorned presentation of One and Zero or Nothing." (Dascal, pg. 416)

By 1679, Leibniz had fully developed his number system but did not publish his work, titled *Essay d'une nouvelle science des nombres*, until 1701 for his election into the Paris Academy of Sciences (O'Connor & Robertson, 1998).

Today, binary is used in a variety of areas, the main area being computer science. A computer's memory is stored using a binary system. There is read-only memory, or ROM, which is the static memory of a computer. This is where applications, operating systems, and saved files are stored on a computer. There is also random-access memory, or RAM, where current process data is stored. This means that the document a student has been writing for hours and has yet to save could be lost should the computer lose power for any reason. When that document is saved, the data is sent to ROM and the computer will be able to open and read the file even after it has lost and regained power. Thankfully, computers have advanced enough that they can recover data after losing

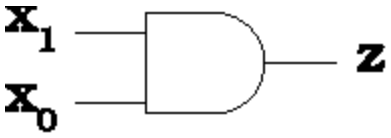
power or a computer crash, so that student can breathe a little easier, but they should still save their document.

Computer memory is organized into cells each containing one bit, meaning either a one or a zero, but how would text, that is not a set of numbers, be saved in a computer? Each word would still be stored as zeros and ones but the American Standard Code for Information Interchange, or ASCII, defines “a” through “z”, “A” through “Z”, and various punctuation marks as combinations of seven binary digits. This allows for 2^7, or 128, different characters. ASCII only uses seven bits because when it was developed in 1963 the creators acknowledged that the seven bit limitation of computers, at the time, could and would change in the future (Brandel, M. 1999). They designed what is known as an “escape sequence” which allows the computer to jump out of one language and into another. This has allowed hundreds of “extra-ASCII” alphabets to be defined and used on the computer.

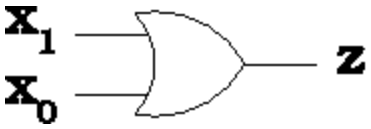
Computers also use binary for all logic based operations. The three basic logic gates computers use are the AND, OR, and NOT gates. The AND gate requires two data units of 1 to produce a 1. If it receives less than two, so either one or zero, data units of 1, it will produce a 0. The OR gate only needs one data unit of 1 to produce a 1. If one or the other data unit or both of the units are 1 it will produce a 1. The NOT gate receives a data unit and passes back the opposite. If a 1 is passed to a NOT gate, it will pass back a 0 and vise a versa. The more advanced logic gates are NAND, which means “not AND”, or NOR, which means “not OR”, or XOR, which means exclusive OR, and or XNOR, which means “exclusively not OR”. Each of these gates’ names describes their function,

so, for example, the NAND gate returns a 1 only if one of the data units is 0, which is the opposite of an AND gate. The NOR gate returns a 1 only if both data units are the same. The XOR gate returns a 1 only if both data units are different, which is the opposite of an NOR gate. All of these gates can be represented symbolically, as follows, where x_i is the input and z is the output.

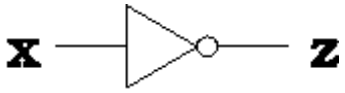
AND:



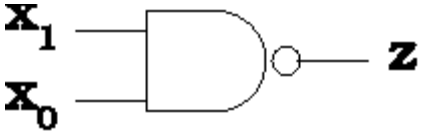
OR:



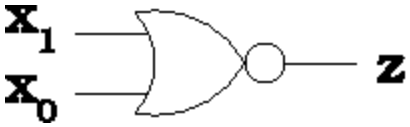
NOT:



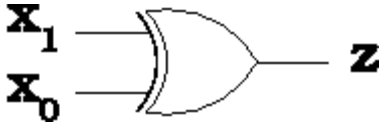
NAND:



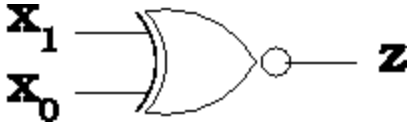
NOR:



XOR:



XNOR:



The base sixteen number system, hexadecimal, is widely used to shorten the character count of binary digits to make them more readable by humans. This numbering

system uses one of the characters “0” through “9” and “A” through “F”, where “A” corresponds with the decimal “10”, “B” with “11”, and so on, in each digit. The first digit is a number zero through nine or A through F times 16^0, which is equal to one, so the number 11 is represented in hexadecimal

$11 * 16^0 = 0xB,$

where the preceding “0x” shows the reader that the number is a base sixteen number. The second digit is a number zero through nine or A through F times 16^1, which is equal to sixteen. The third digit increments the power by one to become 16^2 and this pattern follows for all following digits.

To convert between decimal and hexadecimal, the process is similar to converting between decimal and binary. There is only one step but it is repeated until the last digit, 16^0, is found. First, find the largest power of sixteen that can be subtracted out of the number in question without making it a negative number. Using the number 300_d,

Step 1: $300_d - 16^2 = 300_d - 256_d = 44_d$.

This means that 300_d has a non-zero number as its third digit in its hexadecimal representation. How many times 256_d can only be subtracted out of 300_d will show which non-zero number should be placed in the third digit spot of the hexadecimal representation. This subtraction can only be performed once so a 1 is 300_d’s third digit. Now, repeat step one using the remainder found in the previous step.

Step 2: $44_d - 16^1 = 28_d - 16^1 = 12_d$.

This means that 300_d has a non-zero number as its second digit in its hexadecimal representation and because 16^1 can be subtracted out of 44_d twice that non-zero number is a 2. Now, repeat step one using the remainder found in the previous step.

Step 3: $12_d = C_h$.

Now all three digits have been determined, so

300_d = 0x12C.

Every four binary digits, which can represent up to 2^4, or sixteen, different numbers, can be represented by a single hexadecimal digit. The number 300 in binary is 100101100. If the front of the number is padded with three more zeros it becomes 000100101100, which allows this to be converted hexadecimal much easier. This is easier because of the pattern that emerges due to hexadecimal using base sixteen and that $16 = 2^4$, meaning it is a multiple of two and that each four binary digits make one hexadecimal digit. The first step is to separate every four binary digits:

Step One: 0b 0001 0010 1100.

Now each grouping will be represented by one hexadecimal digit. Step two is converting each of these groups.

Step Two: 0b0001 = 0x1

0b0010 = 0x2

0b1100 = 0xC.

The last step is combining these into one whole number.

Step Three: 0x12C

Converting hexadecimal back to binary is just as easy as converting binary to hexadecimal. First, separate each hexadecimal digit, then convert each digit to its binary representation, and last combine the results into one whole number. For example, the number 0xA5D1:

Step One: 0x A 5 D 1

Step Two: 0xA = 0b1010

0x5 = 0b0101

0xD = 0b1101

0x1 = 0b0001

Step Three: 0xA5D1 = 0b1010 0101 1101 0001.

This is much more straight forward than converting between hexadecimal and decimal.

The origins of the hexadecimal numbering system are convoluted. Modern hexadecimal is an invention of John W. Nystrom, a Swedish-American civil engineer. Nystrom lived from 1825 until 1885. Nystrom held many patents that ranged from a refrigerator to calculating machines. In his book *Project of a New System of Arithmetic, Weight, Measure, and Coins: Proposed to be Called the Tonal System with Sixteen to the Base*, Nystrom discussed why using a base twelve or a base sixteen numbering system would be better for transactions in the market. Base twelve offered one more binary division than base ten but "the number 16 admits binary division to an infinite extent, and would, therefore, be the most suitable number as a base for arithmetic, weight, measure, and coins," according to Nystrom. He claimed that the uneducated class of people naturally wanted to divide quantities into even sections, such as halves, quarters, eighths,

and sixteenths. Base ten only allows for even whole number division into halves. Base twelve would allow whole number division into quarters but base sixteen allows whole number division from halves to sixteenths. Nystrom also discussed the ease at which the average semi educated person could learn to use this base sixteen system. He believed that after the introduction of the sixteen base, it would only take a few years for everybody to grasp the knowledge. Once it is was understood, Nystrom thought it would become the most widely implemented base. Unfortunately, this numbering system did not take hold until the invention of computers, almost a century later, but it still is not as common than the standard decimal system.

One of the first products that utilized hexadecimal was the Bendix G-15 computer which was developed in 1956(Members, 2011). The Bendix G-15 was not a small computer by any means. It was about five feet by three feet by three feet and weighed almost 950 pounds, about the size of a refrigerator.

Figure 2 ("Bendix G-15," n.d.)

This computer did not use "A" through "F" as its hexadecimal representation of the decimal eleven through fifteen. It used the lower case "u" through "z" instead. The base system itself cost $49,500 at the time. The whole working unit cost about $60,000, which translates to over $500,000 today.

Hexadecimal is primarily used within the computer science world. Hexadecimal can be used to make binary more readable to humans because it uses a fourth of the digits that binary does. The standard webpage language, HyperText Markup Language, or HTML, uses hexadecimal and the RGB color model to represent colors. This model uses an additive primary color model which is shown in the following picture.

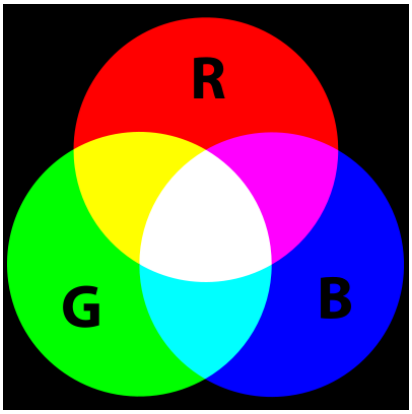


Figure 3 (*440px-AdditiveColor.svg.png (PNG Image, 440 × 440 pixels) - Scaled (0%)*, n.d.)

Two digits of hexadecimal can be used to represent the values zero through two hundred and fifty-five, which is the standard range for each color in the RGB color model. This means that six hexadecimal digits represent any color from the RGB color model. The first two digits represent the red color value. The second two digits represent the green color value. The final two digits represent the blue color value. For example, to obtain the color white, the hexadecimal representation of the RGB color model would be #FFFFFF, where the # symbolizes that it is a RGB color.

Hexadecimal numbers are also used as “magic numbers” in computer processors, operating systems, and debuggers. A “magic number” is used in special situations during a process. These numbers are usually created using words that can be generated using the hexadecimal numbers. The characters “A” through “F” are most common, substituting zero for “O”, substituting one for “L” or “I”, substituting five for “S”, and substituting seven for “T” are also used to create a wider range of words. For example, the exception code 0x8BADF00D, pronounced "ate bad food," is used in the iOS provided by Apple Inc. The Apple Inc. developer website states this exception code will be thrown if “the

application took too long to launch, terminate, or respond to system events." This informal language is known in the computer science community is known as Hexspeak.

IV. Design and Implementation

There are many methodologies that programmers use to aid them in design and implementation. The traditional methodology is known as the Waterfall model, which focuses on completing one step at a time before moving on to the next. ("Waterfall model," 2015) During the senior project sequence, my fellow classmates and I learned about and worked with the Scrum process. ("Scrum Reference Card," 2015)

The Waterfall model focuses on completing one phase of implementation before moving on to another. There are six different phases in the Waterfall model: requirements analysis, design, code, integration, test, and deploy. (Figure 4) This model needs the programmers to completely understand product requirements before they start coding. In theory, this model should word quite well, but in practice it is difficult to anticipate all dependencies and keep errors to a minimum.

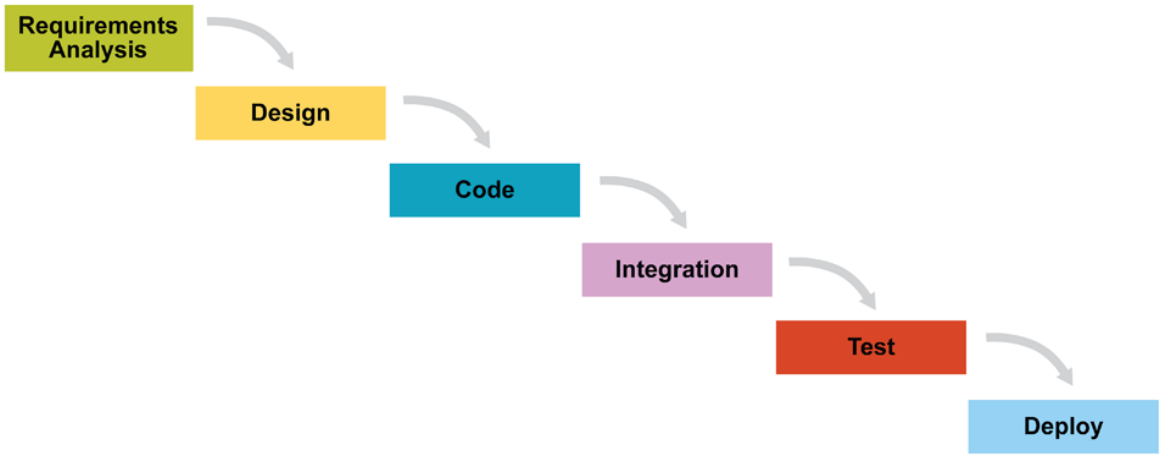


Figure 4 (*traditional-waterfall-development.png (PNG Image, 1024 × 404 pixels)*, 2015)

Unlike the Waterfall model, the Scrum process mixes every developmental phase into programming cycles. (Figure 5) Essentially, the Waterfall model is used over a short

timeframe, usually one to two weeks, to create potentially shippable product every iteration. This allows product owners and financial stakeholders to see a small scale working product and if they are not satisfied with a feature, it saves money because it can be addressed and implemented in the next iteration. This is the main benefit for using the Scrum process over the Waterfall model. If a feature was not implemented in the way the product owner and stakeholders envisioned, the team would have wasted months on its implementation in the Waterfall model but only a couple weeks using the Scrum process.

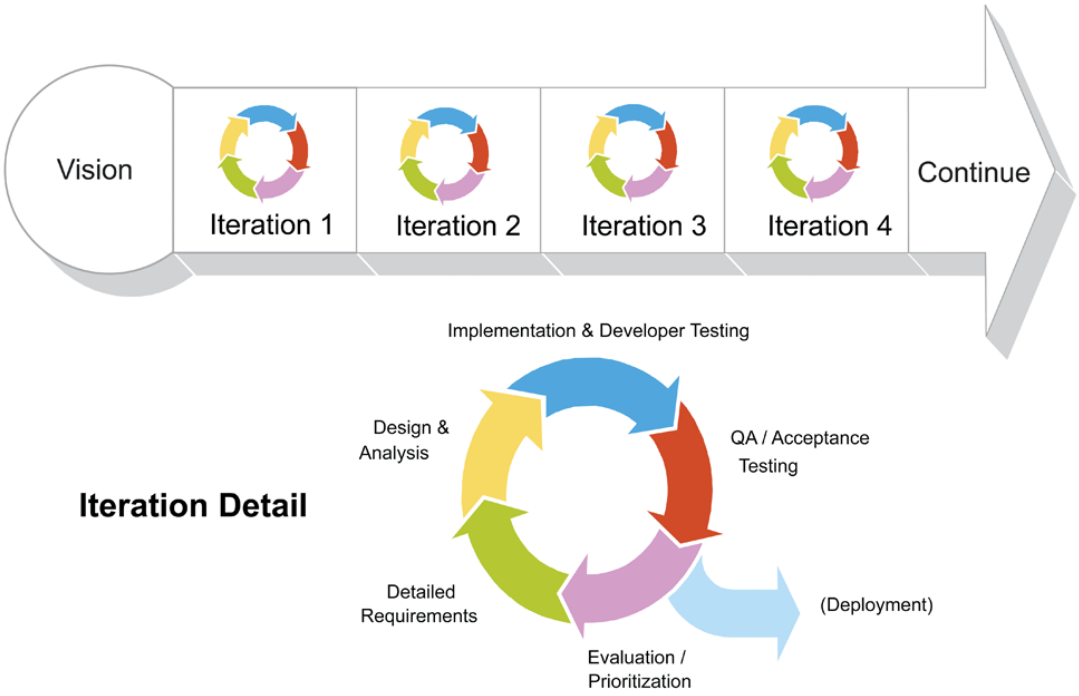


Figure 5 (*scrum-iteration-detail.png (PNG Image, 1024 × 666 pixels) - Scaled (0%)*, n.d.)

Another benefit of the Scrum process is that it puts the responsibility of creating efficient and easily maintainable source code equally on all members of the development team. The team has no traditional manager role. Instead, the Scrum Master assists the team in resolving progress impediments, creating a self-organizing environment, and

shields the team from external interferences and distractions. (“Scrum Reference Card,” 2015) The Scrum Master facilitates all Scrum meetings. In each iteration, the team has five types of meetings that help maintain the flow of progress:

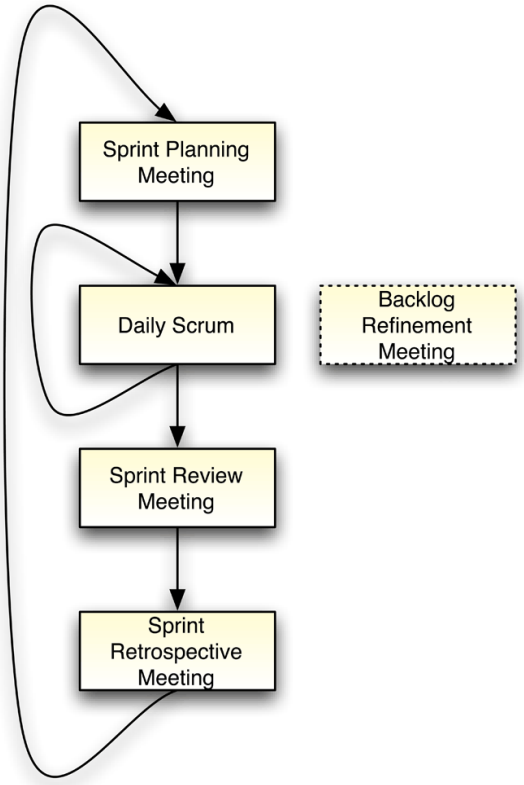


Figure 6 (*scrum-flow.png (PNG Image, 1024 × 1019 pixels) - Scaled (0%)*, n.d.)

During the Sprint Planning Meeting the team commits to what tasks or features each individual will complete. Each task should be small enough for one or two people to complete in a single work day. The team then holds themselves responsible and accountable by explaining what they have done, what they will do, and any obstacles they have had since the last Daily Scrum meeting. As its name implies, the Daily Scrum should happen everyday at the same time. Once the one or two week timebox for the iteration, or sprint, has expired, the team, scrum master, product owners, and stakeholders

have a Sprint Review Meeting to present what the team has completed thus far. The goal of each sprint is to add to the function of a product while still creating a potentially shippable product every time and this meeting is where they will receive feedback. After this meeting, the team and scrum master meet during the Sprint Retrospective Meeting to discuss how the team functioned as a unit through the sprint. This allows any issues to be resolved or worked out before they become a large problem for the team. Finally, periodically the team will meet to have a Backlog Refinement Meeting to reorganize and discuss the tasks in the backlog of what needs to be done for the program.

The Scrum process is designed for teams of four to seven so I had to modify it to be used by only one person. I maintained the sprint structure of committing to work that should be completed in the sprint timebox, reviewing the work completed during every sprint, and refining the backlog every sprint.

After learning the basics of the Scrum process, the next step was to find the proper integrated development environment, or IDE, to program in. Fortunately, the senior project sequence introduced me to Android Studio, an IDE specifically used for designing Android applications. Developers have the opportunity to use many different programming languages with Android Studio. ("Introduction to Android | Android Developers," n.d.) I chose to develop the application, 2A Delivery, in Java while using XML, or EXtensible Markup Language.

The two languages are utilized in different ways. Java is used for all the processing and logic of the application. The XML is used to design the user interface, UI, meaning where text, buttons, and animations are shown and what they look like. Android

Studio allowed me to test my application directly on my mobile device, a Samsung Galaxy 5S. The simplicity of that functionality amazed me. To be able to take my application anywhere and any time, as long as I had my cell phone with me, later proved to be vital in testing my application.

For the duration of the senior project sequence, I continued to implement further functionality and maintained my coding style through refactoring. Through the programming life cycle, there are many points when a programmer, like myself, looks at the source code they are working on and says “Wow. I have abused the copy and paste functions within this program.” A good programmer is quick to notice and often can predict when source code needs to be refactored, or pulled apart and rearranged into a more manageable code base. I like to think of the source code as the misshapen pieces at the end of a game of Tetris when the board is covered. When I am refactoring, my job is to take apart all the pieces to rearrange them in blocks that fit well and match in functionality.

V. Results

As far away as the end of my college career seemed on the first day of fall term in 2011, the years quickly passed and soon it was the end of May, 2015. The modified implementation of the Scrum process helped me organize and break down large tasks of this project into more manageable pieces. Exploring what the Android platform and Android Studio had to offer was slow going at first but has kept my interest.

The most time consuming part of learning a new process is the research and preparation that one must fight through to become proficient. Through the research I did for 2A Delivery, I came upon many interesting features of the Android platform. My two favorite were the custom dialog boxes and the directions overlay. As you can see in Figure 7, a dialog forces the user to look at a specific portion of the screen and note acknowledgement by pressing a button, usually an affirmation button or a cancellation button.(“Dialog | Android Developers,” n.d.) The directions overlay, shown in Figure 8, was created by placing one layout on top of another using a special layout, called a Frame Layout. (“FrameLayout | Android Developers,” n.d.)

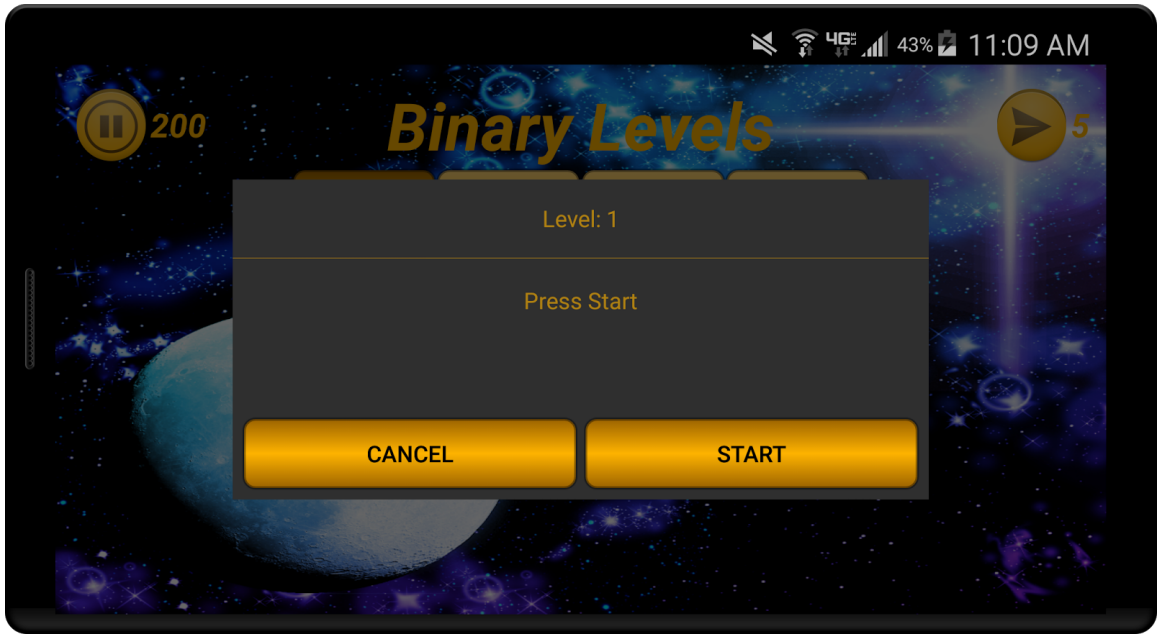


Figure 7: 2A Delivery Start Level Dialog

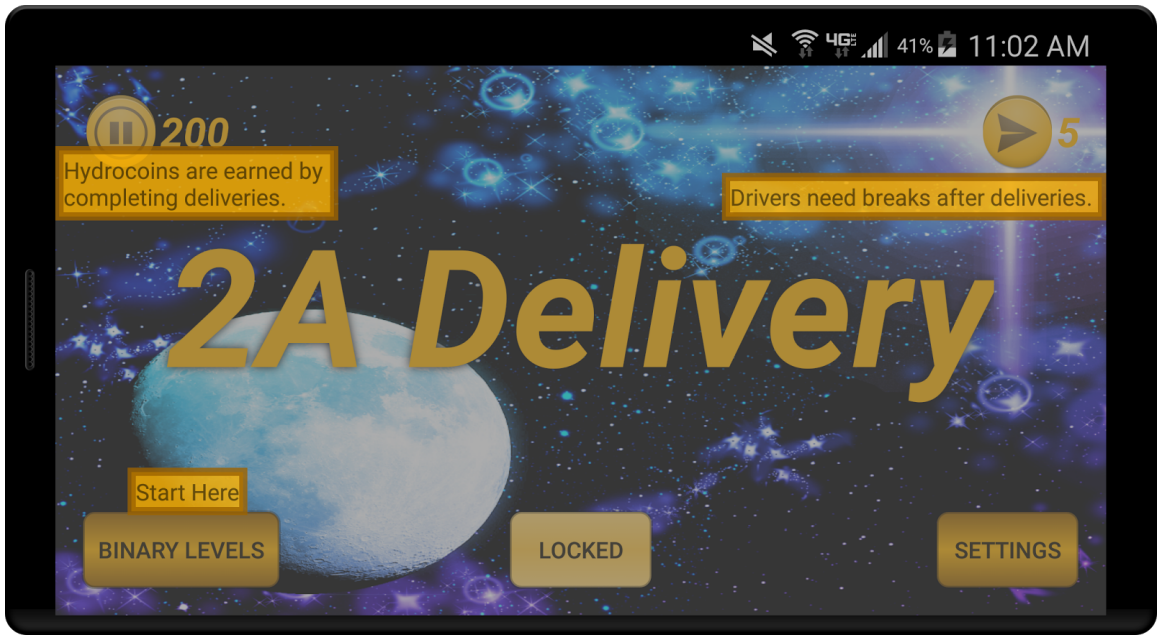


Figure 8: 2A Delivery Main Menu Overlay

In the future, I would like to expand on this project in two ways: by expanding to hexadecimal levels and implementing social interaction and competition. As I mentioned before, much of my time creating this project involved refactoring the source code to

make it more efficient. This has two main benefits: the program will run more efficiently and the program will be easier to update and add too in the future.

The first release has fully functioning binary levels, which means that the major implementation and framework is complete. Copying and then modifying the binary levels' functionality to produce hexadecimal levels will take considerably less time. The layout of the user interface is designed using multiple components for the title, timer, text fields, and fragment. A fragment is a group of components usually determined after some program computation. In 2A Delivery's case, the buttons that the user inputs the binary address with can be replaced with a set of buttons that correspond to the hexadecimal numbering system. This replacement would be determined from which location the user wished to deliver packages to, or whether the user wanted to convert from decimal to binary or decimal to hexadecimal. After that, the next release could include levels where the users convert from binary to decimal or hexadecimal to decimal.

Over a longer period of time, small releases would be perfect to push new social features to 2A Delivery. This will be an extended task due to the competition aspect of this feature. Users would connect via social media accounts. Then they would be able to compete on the tough battleground of the workplace. The more deliveries the user can run with their drivers, the higher up in the company they will move. The higher up in the company they move, the more perks they will receive for their status.

The exploration required during this project reinforced my belief that software engineering is the right path for me. I discovered something new every day of the project. The ability to write in a computer's language to create whatever I wanted to see on my

computer has always amazed me. From the animation of the sunrise and sunset over the childish house to the complex word dictionary sorting required to perform the spell check function, my problem solving was forced to adapt and discover creative solutions. The accomplishment felt after completing such a task makes the critical thinking enjoyable and rewarding.

VI. Appendix A: Bibliography

Apple Inc. (2012). Understanding and Analyzing iOS Application Crash Reports. Retrieved from

Introduction to Android | Android Developers. (n.d.). Retrieved May 25, 2015, from http://developer.android.com/guide/index.html

Lin, C. (2003). Logic Gates [Webpage Document]. Retrieved from http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Comb/gates.html

MacFarlane, I. C. (2005). The Enneagram - History of the Decimal System [Webpage Document]. Retrieved from http://www.endlesssearch.co.uk/philo_enneagram_dec.htm

Nystrom, J. W. (1862). Project of a New System of Arithmetic, Weight, Measure, and Coins: Proposed to be Called the Tonal System with Sixteen to the Base. Retrieved from http://books.google.com/books?id=aNYGAAAAYAAJ&source=gbs_navlinks_s

O'Connor, J., & Robertson, E. (October 1998). Gottfried Wilhelm von Leibniz. Retrieved from http://www-history.mcs.st-andrews.ac.uk/Biographies/Leibniz.html

Scrum Reference Card | The Scrum Reference Card, and Other Articles by Michael James (MJ), Software Process Mentor. (n.d.). Retrieved May 11, 2015, from http://scrumreferencecard.com/

Stack Overflow. (n.d.). Retrieved May 25, 2015, from http://stackoverflow.com/

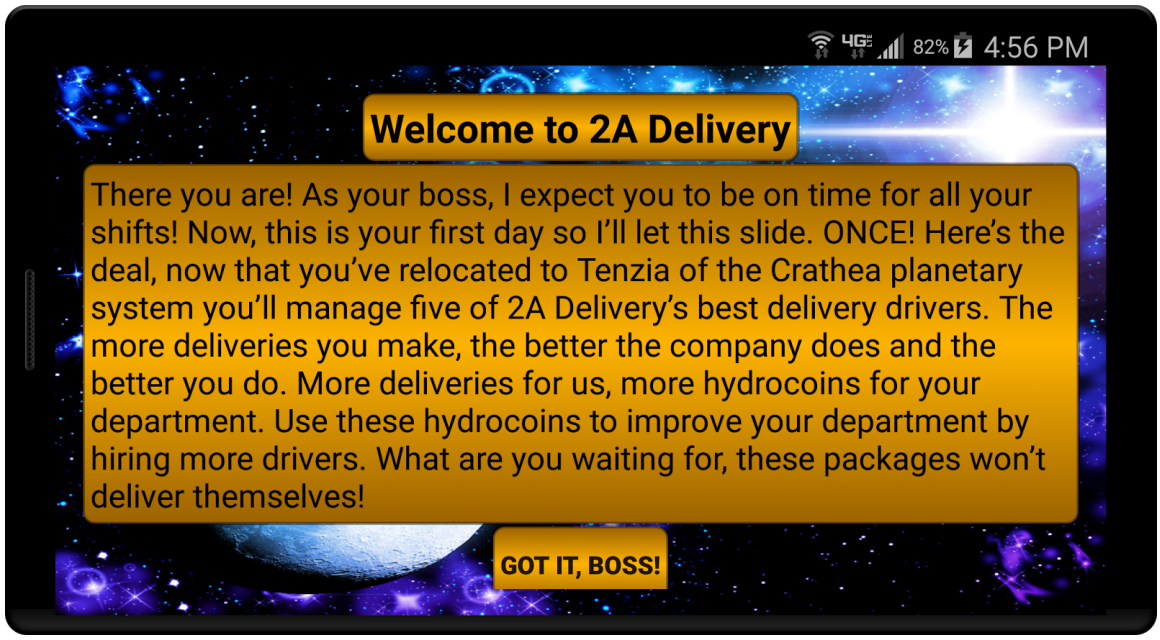
Strickland, L. (2007). Explanation of Binary Arithmetic. Retrieved from http://www.leibniz-translations.com/binary.htm

Unknown (2011, September 29). The First Generation Computers. Retrieved from http://members.iinet.net.au/~dgreen/index.html

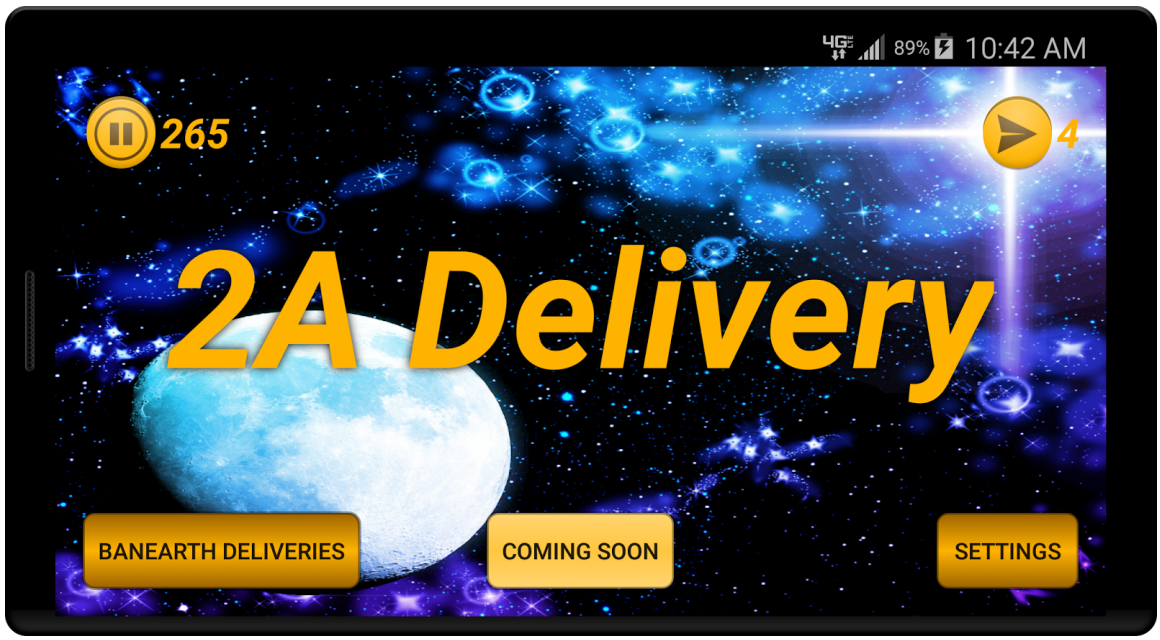
Waterfall model. (2015, May 5). In *Wikipedia, the free encyclopedia*. Retrieved from http://en.wikipedia.org/w/index.php?title=Waterfall_model&oldid=660906933

440px-AdditiveColor.svg.png (PNG Image, 440 × 440 pixels). (n.d.). Retrieved from http://upload.wikimedia.org/wikipedia/commons/thumb/c/c2/AdditiveColor.svg/440px-AdditiveColor.svg.png

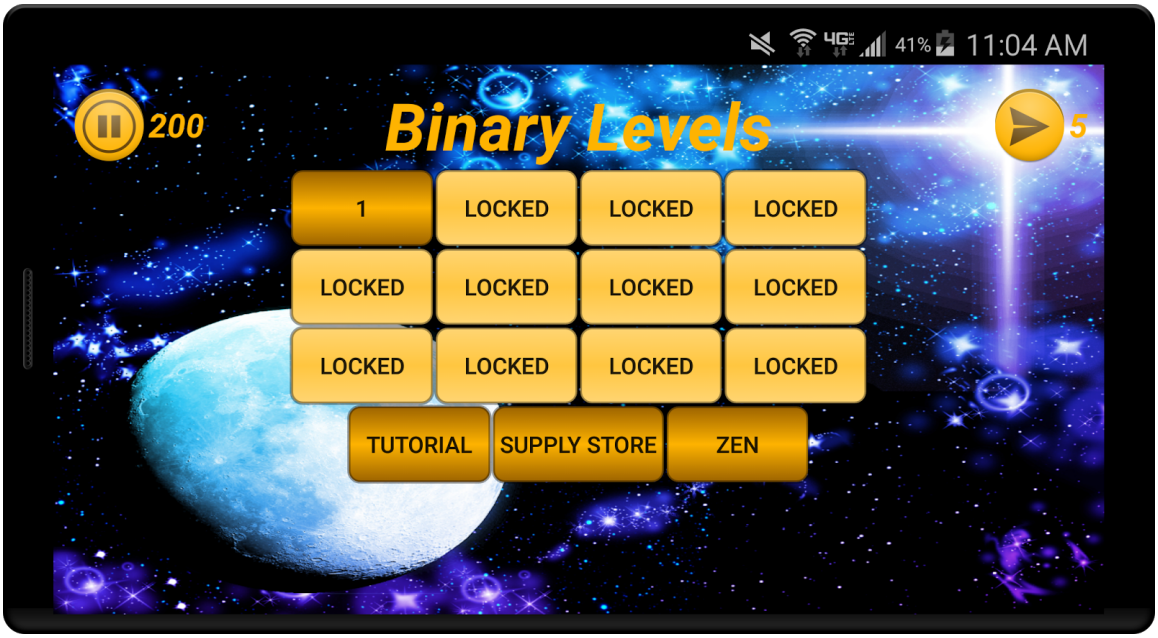
VII. Appendix B: Application Screenshots



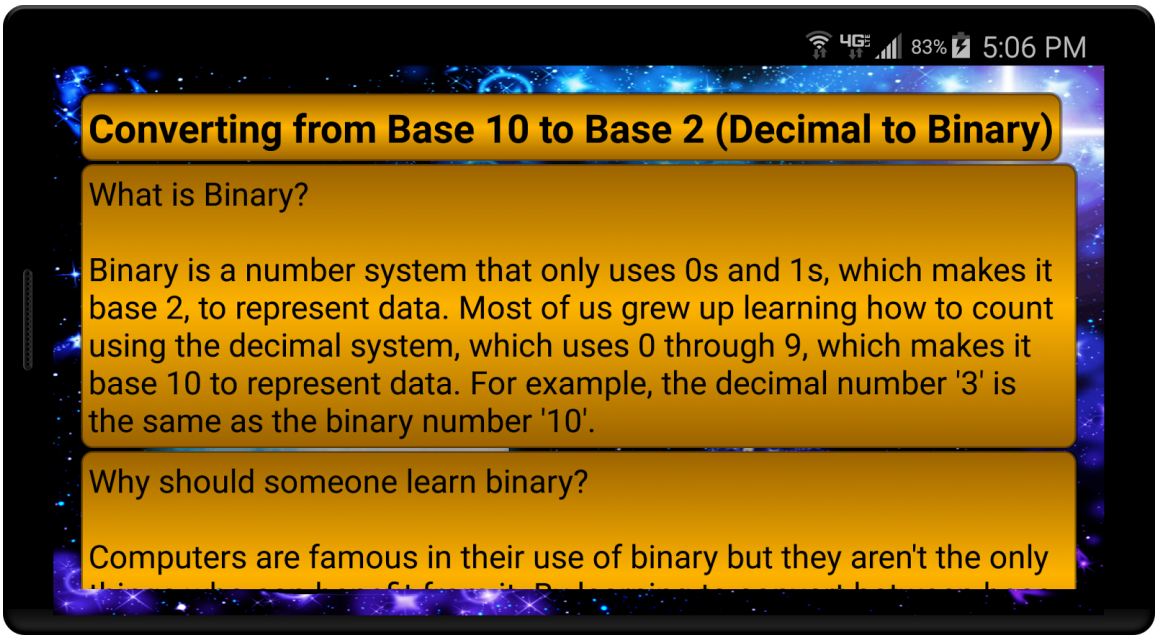
"Welcome"



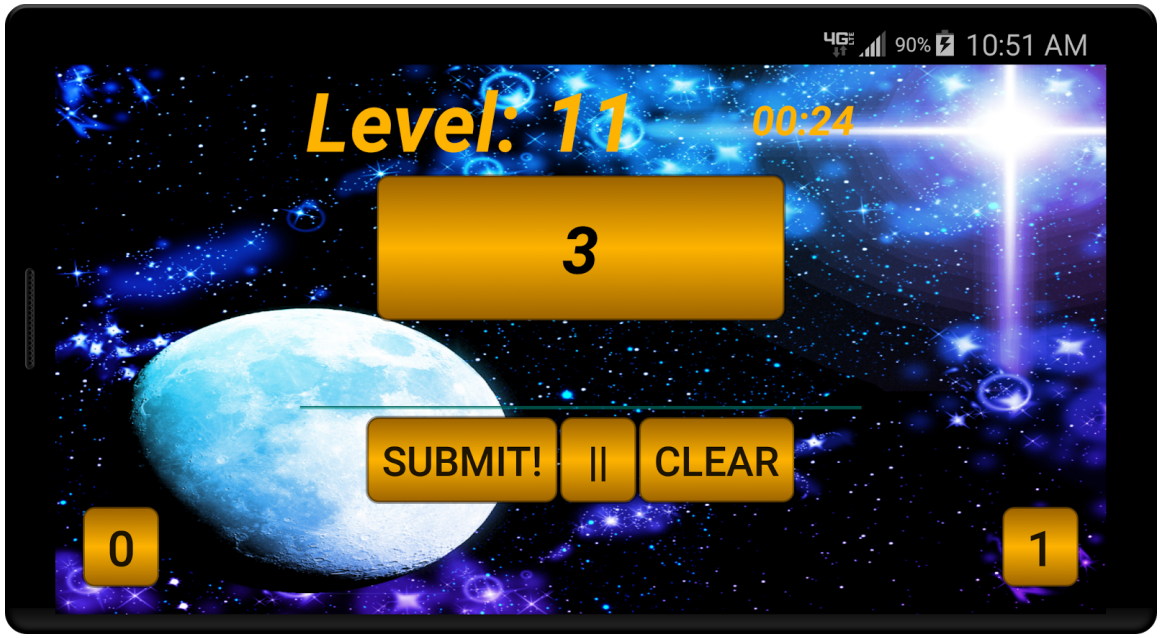
"Main Menu"



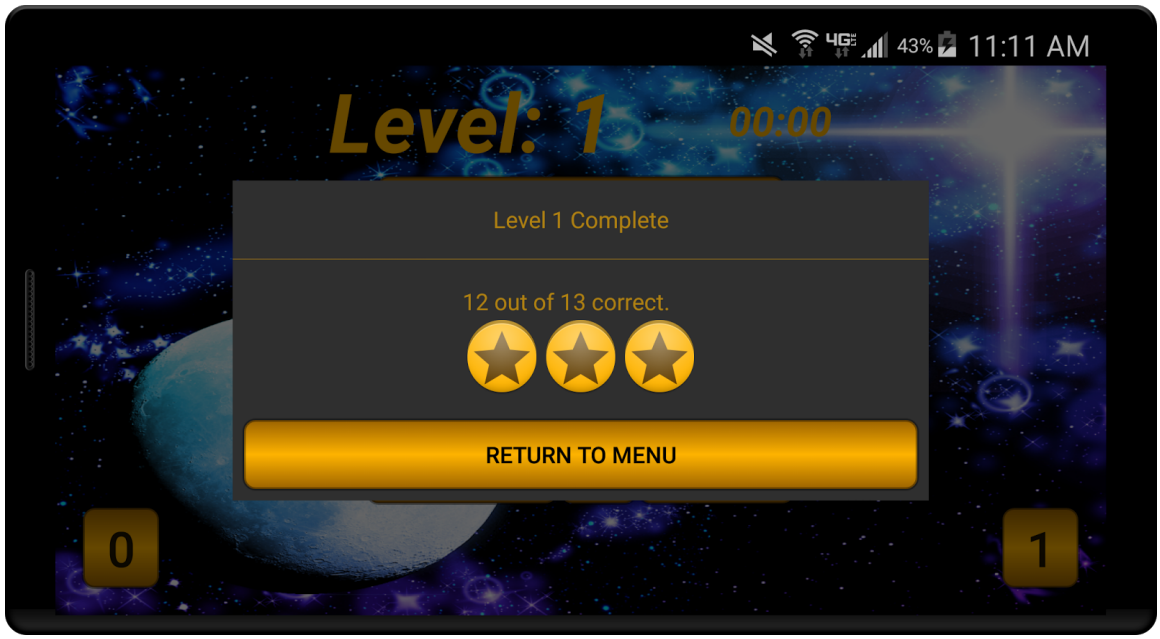
“Banearth Deliveries (Binary Levels)”



“Binary Conversion Tutorial”



“Banearth Address Reassignment”



“Level Complete Dialog”