Boolean Algebra Application in Creating a Dynamic Seven-Segment Wall in No Man's Sky

Hanif Muhamad Gana - 13518127
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518127@itb.ac.id

***Abstract*—Computers have gone through many advancements. Most computers nowadays can easily compute up to 2^{64} in numbers, but most programs do not make use of it. One program that does is a video game called No Man's Sky, with 2^{64} available to explore. However, players have no way to create dynamic messages to mark those planets. Fortunately, as the game supports electricity and electrical switches, it is should be possible to create a dynamic seven-segment display within it. Unfortunately, building such a thing would require quite a lot of resources, therefore, we need to find the most minimal and effective circuit by applying Boolean algebra and Karnaugh's Map method.**

***Keywords*—Boolean algebra, Seven-segment display, Logical circuits, Exploration game.**

I. Introduction

As time went on, humans create more and more advanced technology. One of the most important technology is the computer. The computer allows human to be more productive by doing several complex calculations on the background at the same time. What began as a programmable analytical engine in the 19th century has now become a multipurpose device capable of feats such as simulating physics, solving mathematical calculations, bookkeeping, office work, and being a source of entertainment easily found within most buildings in the world.

These days it is not hard to find a computer in most countries of the world. In addition to its cost becoming more consumer friendly, the computer now has many varieties for many different purposes, such as smartphones for handheld communication, laptops for portable computing, and game consoles for playing video games. That being said, one technological leap in one of those varieties could easily be found in the other varieties, and as such they tend to share several similarities with one another. One of those similarities is that most computers these days have a 64-bit CPU as its processor, easily able to compute up to 2^{64}.

However, most programs using those computers do not use the full potential of the computer they are running on. One program that does, however, is a video game called No Man's Sky by Hello Games. No Man's Sky is a game that allows the players to explore 2^{64} procedurally generated planets with procedurally generated plants, buildings, weather, terrain, creatures, among other things. In addition to exploring planets, players are able to create bases and buildings powered by electricity controllable by electric switches that can be turned into grand structures that amaze the mind.

As grand as that structure may be, however, players do not really have a way to write messages visible from a distance, especially one that is dynamic in accordance to the state of other structures near it. A dynamic seven-segment display structure could be used to help put dynamic messages visible from a distance. Seven-segment displays can be created using electricity, electric switches, and lights, all of which are available within the game, and can be programmed to be dynamic. However, it is important to note that resources in No Man's Sky are required to build and generate said electricity, electric switches, and lights, therefore it is important that the seven-segment display created use as few resources as possible.

II. Theoretical Basis

A. Boolean Algebra

Boolean Algebra is a branch of mathematics, found by an English Mathematician named George Boole in 1854. He was intrigued when he saw that the properties of set mathematics and of propositional logic had many similarities and created what is now known as Boolean Logic in his book *The Laws of Thought,* which become the basis of Boolean Algebra. Boole's work was continued by Claude Shanon in 1938 who used Boolean Algebra to create an electrical circuit what accepts 0 and 1 and outputs in 0 and 1. Claude Shanon's findings become the basis of the modern digital computer used today.

One of the most common way to abstractly define Boolean Algebra is to define the its building elements and operations. Let B be a set that is defined to two binary operators, + and ·, and a unary operator, '. Let 0 and 1 be two different elements within B such that the tuple

$$< B, +, \cdot, ', 0, 1 >$$

is a Boolean algebra if for every a, b, c ∈ B these axioms (also known as Huntington's Postulates) apply:

1. Identity

(i) a + 0 = a
(ii) a · 1 = a

2. Commutative
(i) a + b = b + a
(ii) a · b = b · a

3. Distributive
(i) a · (b + c) = (a · b) + (a · c)
(ii) a + (b · c) = (a + b) · (a + c)

4. Complement
For every a ∈ B there exists a unique element a’ ∈ B such that
(i) a + a’ = 1
(ii) a · a’ = 0

Note that 0 and 1 are two unique elements in B. 0 is referred to as the smallest element, while 1 is referred to as the largest element. The two unique elements within B can be many different things (e.g., *Ø* and *U* in set Boolean algebra, and **F** and **T** in propositional logic). Element 0 is also known as the *zero* element while the element 1 is also known as the *unit* element. The operator + or OR is known as the Boolean sum operator, the operator · or AND is known as the Boolean multiplication operator, and the ‘ or NOT is known as the Boolean complement operator.

Although some operators in Boolean algebra look like their counterparts in the algebra for real number arithmetic there are several differences. They are as follows:

1. The first distributive rule of the Huntington’s Postulates, a · (b + c) = (a · b) + (a · c), is known and applicable to real arithmetic algebra, however the second distributive rule, a + (b · c) = (a + b) · (a + c), while correct in Boolean algebra is incorrect and therefore inapplicable in the real arithmetic algebra.
2. Boolean algebra does not recognize any multiplicative inverse nor does it recognize any sum inverse; as such, there are no such things as division and subtraction operations within Boolean algebra.
3. The fourth postulate in Huntington’s Postulates describes something known as complements, also known as negations, which do not exist within real arithmetic algebra.
4. In the algebra for real number arithmetic, a set of real numbers is treated as if it has infinite elements. However, the elements of set B in Boolean algebra is undefined, but as for two-valued Boolean algebra the set B is defined as having exactly two elements, the smallest element 0 and the largest element 1.

As the set of B in Boolean algebra is undefined, to have a Boolean algebra one is needed to show:
1. elements in B,
2. the rules for the binary and the unary operators,
3. that the set B and the operators defined above, satisfy the four rules in the axiom above (Huntington’s Postulates).

If all three requirements above are fulfilled then the algebra can be considered a Boolean Algebra.

B. Two-Valued Boolean Algebra

As the elements of set B is undefined, then there exists an infinite number of Boolean algebra. However, the set B has at least two elements, as shown in Boolean algebra’s definition of the set B having two different elements.

The Boolean algebra where B has exactly two elements, 0 and 1 (usually called as bit – short of binary digit), where B = {0, 1}, two binary operators, + and ·, and a unary operator, ‘, is known as two-valued Boolean algebra. This two-valued Boolean algebra is one of the most popular and most applicable Boolean algebra and its operator rules are as shown in the tables below.

Table I

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

Table I

a	b	a · b
0	0	0
0	1	0
1	0	0
1	1	1

Table III

a	a’
0	1
1	0

With the operator rules defined above, the four axioms (Huntington’s Postulates) are fulfilled as follows:

1. Identity
From the tables above we can show that:
(i) 0 + 1 = 1 + 0 = 1
(ii) 0 · 1 = 0 · 1 = 0
which fulfills the identity property.
2. Commutative
This property is fulfilled as can be seen from the symmetry of the binary operators shown in TABLE I and TABLE II.
3. Distributive
In order to show that the rules satisfy the distributive property, a truth table with three operands can be used. The truth table for the first distributive rule, a · (b + c) = (a · b) + (a · c), is shown below.

Table IV

a	b	c	b + c	a · (b + c)	a · b	a · c	(a · b) + (a · c)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1

1	1	1	1	1	1	1	1

The truth table for the second distributive rule, a · (b + c) = (a · b) + (a · c), is shown below.

Table V

a	b	c	b · c	a + (b · c)	a + b	a + c	(a + b) · (a + c)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

4. Complement

From all the tables above, we can show that:

(i) a + a’ = 1, because for 0 + 0’ = 0 + 1 = 1 and for 1 + 1’ = 1 + 0 = 1, and

(ii) a · a’ = 0, because for 0 · 0’ = 0 · 1 = 0 and for 1 · 1’ = 1 · 0 = 0

As it has been shown that the four axioms are satisfied, it as been proven that the tuple < B, +, ·, ‘, 0, 1 >, with B = {0, 1}, the binary operators + and ·, and the complement operator ‘, is a Boolean algebra. Hereafter, all Boolean algebra in this paper will be considered a two-valued Boolean algebra without being explicitly stated as so.

In addition to the rules defined in Huntington’s Postulates, there are many other rules that the Boolean algebra exhibit, some of the most important ones are displayed within Table VI below.

Table VI

1. Identity rule: (i) a + 0 = a (ii) a · 1 = a	2. Idempotent rule: (i) a + a = a (ii) a · a = a
3. Complement rule: (i) a + a’ = 1 (ii) aa’ = 0	4. Domination rule: (i) a · 0 = 0 (ii) a + 1 = 1
5. Involution rule: (i) (a’)’ = a	6. Absorption rule: (i) a + ab = a (ii) a(a + b) = a
7. Commutative rule: (i) a + b = b + a (ii) ab = ba	8. Associative rule: (i) a + (b + c) = (a + b) + c (ii) a (b c) = (a b) c
9. Distributive rule: (i) a + (b c) = (a + b) (a + c) (ii) a (b + c) = a b + a c	10. De Morgan rule: (i) (a + b)’ = a’b’ (ii) (ab)’ = a’ + b’
11. 0/1 rule: (i) 0’ = 1 (ii) 1’ = 0	

C. *Boolean Functions*

Boolean functions, also called binary functions, is the mapping from B^n to B through Boolean expression. Boolean function can be denoted as follows :

$$f : B^n \rightarrow B$$

where B^n is a set containing ordered n-tuple in domain B.

Boolean expressions that specifies Boolean functions can be written in two forms, a sum-of-product form and a product-of-sum form.

f(x, y, z) = x’y’z + xy’z’ + xyz

and

g(x, y, z) = (x + y + z)(x + y’ + z)(x + y’ + z’)(x’ + y + z’)(x’ + y’ + z)

are equal. The difference being that f is a Boolean function written in sum-of-product form, while g is a Boolean function written in product-of-sum form. Each term in the expression contains the complete variable x, y, and z, be it with a complement or without. Generally speaking, there are two kinds of terms, minterm (the result of product) and maxterm (the result of sum).

A Boolean expression with n variables $x_1, x_2, \ldots, x_n$ is said as a minterm if it was written as

$$\tilde{x}_1 \cdot \tilde{x}_2 \cdot \tilde{x}_3 \cdot \ldots \cdot \tilde{x}_n$$

and as a maxterm if it was written as

$$\tilde{x}_1 + \tilde{x}_2 + \tilde{x}_3 + \ldots + \tilde{x}_n$$

with the $\tilde{}$ symbol being used to denote that x_i can be in its complement or not.

Seeing that there are multiple ways to write a Boolean expression specifying a Boolean function, there must be a way to obtain a simpler form of said Boolean expression with fewer literals or operations. For example, the function f(x, y) = x’y + xy’ + y’ can be simplified to f(x, y) = x’ + y’.

There are several methods that can be used to simplify a Boolean function, they are:

1. Algebraic, using Boolean algebra rules.
2. Karnaugh’s Map method.
3. Quine-McCluskey method.

D. *Karnaugh’s Map Method*

Karnaugh’s Map or K-Map is a graphical method to simplify a Boolean function. This method uses a diagram/map formed from adjancent squares. Each square in the diagram represents a minterm, with each neighbouring squares only differing in 1 literal in the minterm that they represent.

Here are several K-Maps:

		y	
		0	1
x	0	x’y’	x’y
	1	xy’	xy

A K-Map with two variables

		yz			
		00	01	11	10
x	0	x’y’z’	x’y’z	x’yz	x’yz’

1	xy’z’	xy’z	xyz	xyz’

A K-Map with three variables

		yz			
		00	01	11	10
	00	w’x’y’z’	w’x’y’z	w’x’yz	w’x’yz’
w	01	w’xy’z’	w’xy’z	w’xyz	w’xyz’
x	11	wxy’z’	wxy’z	wxyz	wxyz
	10	wx’y’z’	wx’y’z	wx’yz	wx’yz’

A K-Map with four variables

One can use K-Map to simplify a Boolean function’s expression by filling each square representing a minterm within the Boolean function with 1 and the remaining squares with 0.

For example, the function f(x, y, z) = x’yz’ + xyz’ + xyz would have a K-Map as follows:

		yz			
		00	01	11	10
x	0	0	0	0	1
	1	0	0	1	1

Which can be written in the simplified form f(x, y, z) = xz’ + y.

In order to simplify a Boolean function using a K-Map one needs to group adjacent squares with the value 1. These groups can be one of these kinds:

1. Dual, with two adjacent squares.
2. Quad, with four adjacent squares creating a line or a square.
3. Octet, with eight adjacent squares creating a rectangle with two rows and four columns.

Note, however, that the map wraps around itself. Therefore, the squares at the right-most edge are adjacent to the squares at the left-most edge.

E. Logical Circuit

A Boolean expression can be represented as a logical circuit. A circuit can receive input and sends output in the form of electrical pulses that can be seen as either 0, when the pulse has low voltage, or 1, when the pulse has high voltage. Boolean algebra is used to model a logical circuit. One of the most elementary elements of a logical circuit are logical gates, each representing a Boolean operation, and created from electrical switches. There are three basic gates, the AND gate, representing a multiplication operation, the OR gate, representing a sum operation, and a NOT gate, representing a complement operation.

Each of those three gates have a symbol to represent them in a circuit diagram, as well as a corresponding truth table. They are as follows:

1. AND gate

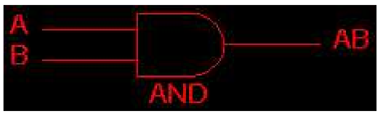


Image 1. The AND gate symbol
(Source: http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/index.html)

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

Table VI. AND gate truth table

2. OR gate

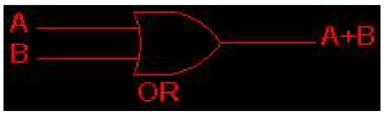


Image II. The OR gate symbol
(Source: http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/index.html)

A	A	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Table VII. OR gate truth table

3. NOT gate

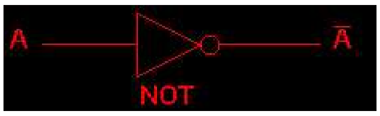


Image III. The NOT gate symbol
(Source: http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/index.html)

A	A’
0	1
1	0

Table IX. NOT gate truth table

F. Seven-Segment Display

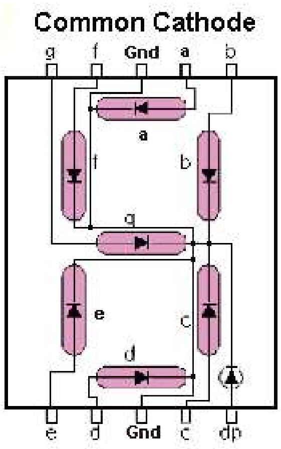


Image IV. Seven-Segment display
(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)

A Seven-Segment display is a display consisting of seven LEDs (or any other object to emit light), arranged in a rectangular fashion. Each of the LEDs creeating a seven-segment display forms a part of a numerical digit when lit up. Several seven-segment displays can be used to display a number, or a word, with a length equal to the amount of seven-segment display being used.

Each of the seven LEDs in a seven-segment display has a positional segment with a connection pin to light up the corresponding LED. These connection pins are usually labeled a through g, as seen in Image IV above. However, most Seven-Segment display circuits would usually have a circuit that can receive 4 inputs to light up the seven-segment display.

Let, A, B, C, and D, be the variable representing the 4 inputs the circuit can receive, and a, b, c, d, e, f, and g, represent the state of one of the seven segments as shown on Image IV, the truth table to make the seven-segment display each decimal number would be as follows.

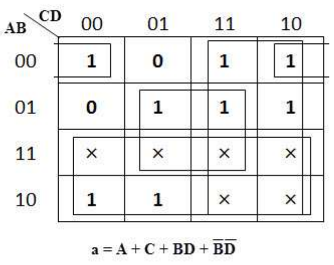
Digit	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

Table X. Seven-Segment truth table
(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)

From the truth table, we can write the Boolean expressions for each output functions as

a = F1 (A, B, C, D) = ∑m (0, 2, 3, 5, 7, 8, 9)
b = F2 (A, B, C, D) = ∑m (0, 1, 2, 3, 4, 7, 8, 9)
c = F3 (A, B, C, D) = ∑m (0, 1, 3, 4, 5, 6, 7, 8, 9)
d = F4 (A, B, C, D) = ∑m (0, 2, 3, 5, 6, 8)
e = F5 (A, B, C, D) = ∑m (0, 2, 6, 8)
f = F6 (A, B, C, D) = ∑m (0, 4, 5, 6, 8, 9)
g = F7 (A, B, C, D) = ∑m (2, 3, 4, 5, 6, 8, 9).

However, the circuit that would be created from those expressions would be very complex and require many logical gates. As such, it is important that we simplify them with Karnaugh's Map method before creating their logical circuit representation.



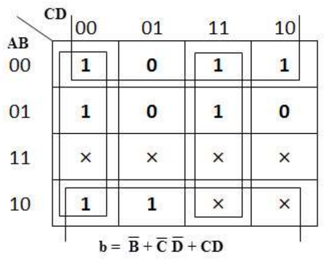
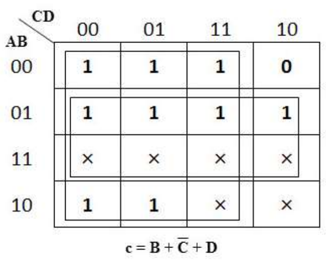


Image V and VI, Karnaugh's map for segment a and b.
(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)



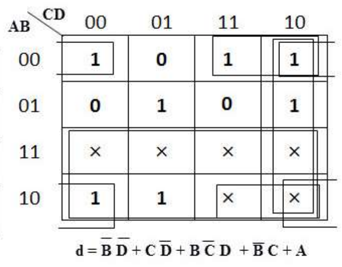
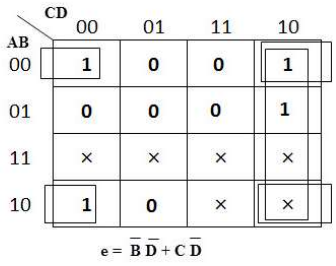


Image VII and VIII, Karnaugh's Map for segment c and d
(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)



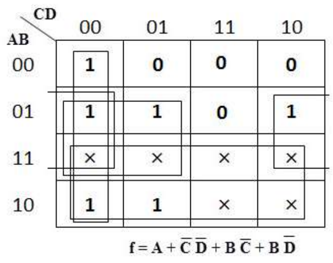


Image IX and X, Karnaugh's Map for segmend e and f
(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)

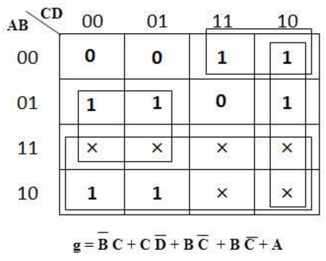


Image XI, Karnaugh's Map for segment g

(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)

Every Karnaugh's Map represented by Image V through Image XI, represents each segment in the seven-segment display. The seven output functions from before can now be written in a much more simple and compact form, requiring significantly fewer logic gates. The simplified Boolean expressions are as follows:

$$a = A + C + BD + \bar{B}\bar{D}$$
$$b = \bar{B} + \bar{C}\bar{D} + CD$$
$$c = B + \bar{C} + D$$
$$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$$
$$e = \bar{B}\bar{D} + C\bar{D}$$
$$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$$
$$g = \bar{B}C + C\bar{D} + B\bar{C} + B\bar{C} + A$$

This simplified expressions can be turned into an effective logical circuit. This circuit would have four inputs, namely A, B, C, and D, and seven outputs, namely a, b, c, d, e, f, and g. The logical circuit for the seven Boolean expressions would be as follows.

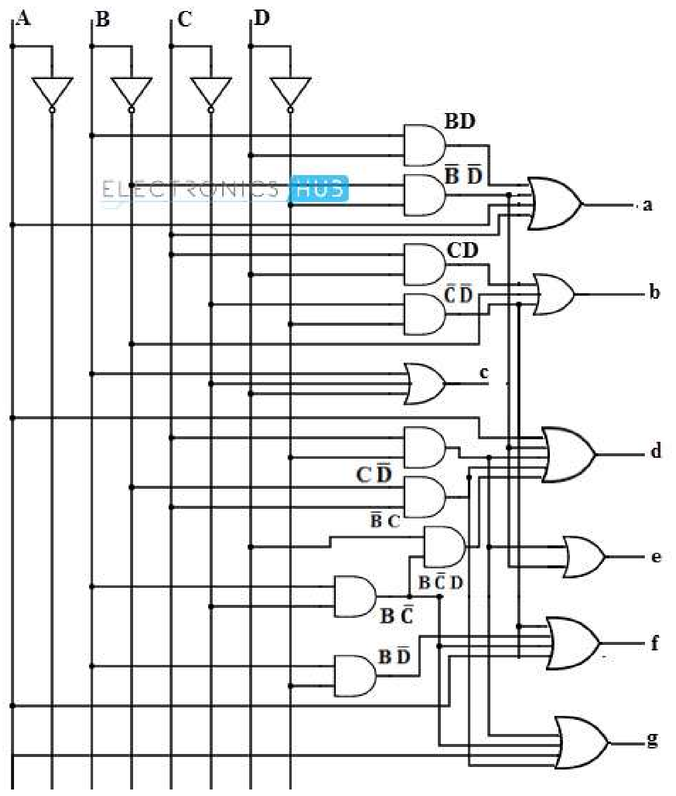


Image XII. The logical circuit of a seven-segment display

(Source: https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/)

F. No Man's Sky

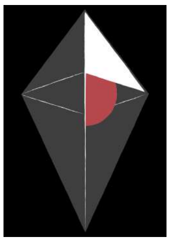


Image XIII. No Man's Sky Logo

(Source: https://www.nomanssky.com/press/)

No Man's Sky is a game about exploration and survival in a near infinite procedurally generated galaxy. No Man's Sky is a PS4, XBOX ONE, and Windows PC video game created by Hello Games released on August 2016. The main focus of the game is exploring 2^{64} procedurally generated planets with procedurally generated plants, buildings, weather, terrain, creatures, among other things.

In the game, the player begins as a stranded explorer in an alien planet with minimal resources and broken equipments. The main goal within the game is to explore the universe, its planets, its lore, and its reaches to gather enough resources, such as sodium, carbon, etc., to reach dangerous space anomalies containing more lore and hopefully reach the galactic center.

In the game, players only have a limited amount of inventory with which to hold resources. These resources are often required to craft objects that may be benificial in one way or another. Although it is impossible for players to ever run out of planets to harvest resources, harvesting them often requires a lot of space travel and time. As such, it is important for players to learn how to manage and minimize their resource consumption.

The game has received eight major updates since its release up to December 2019. Those major updates are known as:

1. The Foundation update, which added base building, new game modes, new materials, purchasable freighters and the first galaxy regeneration.
2. The Pathfinder update, which added land vehicles, online base sharing, and ship/weapon classes.
3. The Atlas Rises update, which added a new story, better graphics, new planets, a new alien race, the first form of multiplayer, and the second galaxy regeneration.
4. The NEXT update, which added new multiplayer interactions, including PvP and co-op base building and exploration, the option to play in third person, an expanded storyline, and the third galaxy regeneration.
5. The Abyss update, which added many expanded underwater features, including an exocraft submarine, and increasing the likelihood of an oceanic planet.
6. The Visions update, which added new biome types and many different land features, as well as expanding the game's color generation.
7. The Beyond update, which added VR, Expanded Multiplayer and an increase in features in most game aspects, including electrical power grid and switches.
8. The Synthesis update, which added ship salvaging and upgrading, new weapons upgrades, as well as several quality-of-life fixes.

Through all those updates, however, players still do not have a way to easily display a message dynamically. Though building structures that look like messages can be an option, it is often a lot of work to rebuild the structure should the player wish the change the message.

Since the Beyond update, No Man's Sky supports the creation of electrical grids which can be controlled using several different switches. The switches can be used to control how power flows in a player's base and acts a lot like real life electrical switches. These switches can be formed together to create logic gates and therefore dynamically programmable seven-segment displays. However, it is important to note that building the aforementioned switches cost resources and since resources have to be gathered, it is important to create the a seven-segment display with the least amount of switches and gates.

There are several electrical switches in No Man's Sky, however in this paper we will only see at three. The Wall Swtich is an electrical switch in No Man's Sky that the player can trigger to choose whether to let electricity flow through or not. The Auto Switch is an electrical switch in No Man's Sky that the player will let electricity flow through if a controlling signal flows through it. The Power Inverter will let electricity flow through if a controlling signal does not flow through it.

III. Dynamic Seven-segment Display in No Man's Sky

No Man's Sky allows its players to create and build bases within which they may have electricity generated by one of the power generators available and which flow is controllable using the available electric switches. However, players do not really have a way to build programmable displays which can be dynamically changed at will. Keeping in mind that the No Man's Sky game supports electricity in its game mechanics, and that the players can build electrical switches to control its flow, which is the basis of logic gates, it is then not too far fetched to say that creating a seven-segment display in the game might be a good option to tackle this problem.

A. Logic Gates in No Man's Sky

No Man's Sky does not have logical gates in its building catalog the can be instantly used to build the seven-segment display circuit as shown on Image XII. However, it does support electrical switches, which are the basis of real life logical gates. As such, it is then possible to build, or at least simulate, logical gates within No Man's Sky.

The AND gate can be built by using a series of Auto Switches available within the game with the controlling flow connected to different electrical flows that can be controlled. The input of the series of Auto Switches would be a constant electrical flow, such that when all of the control signal flows are flowing the last of the Auto Switches would have power that will then be sent to the object needing it.

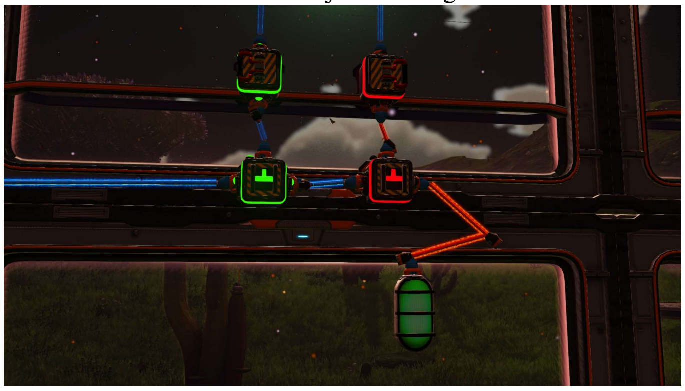


Image XIV. An AND gate in No Man's Sky
(The blue color means that the cable has electricity, while the red-orange color means that it does not have electricity. Likewise, the green color on the switches means it is letting electricity flow through it)

The OR gate can be built using one Auto Switch available within the game with the controlling flow connected to different electrical flows, and the input of the Auto Switch being a constant electricity flow. This setup allows the Auto Switch to let electricity flow through when at least one of the control signals flow through it, as is how an OR gate woul work.

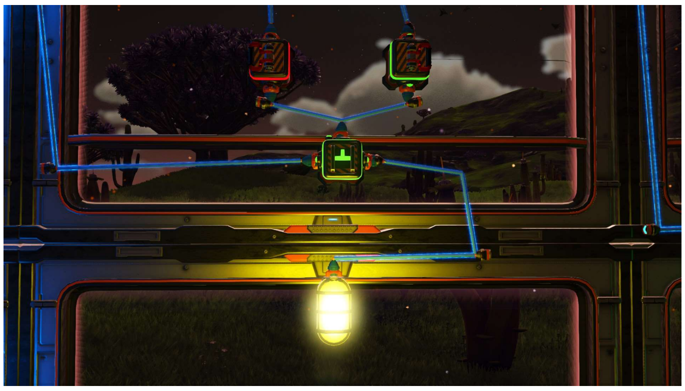


Image XIV. An OR gate in No Man's Sky

The NOT gate can be built using a Power Inverter available within the game, with the controlling flow connected to an electrical flow which is to be complemented, and the input flow connected to a constant electrical flow. This setup would make the Power Inverter let electricity flow through when the controlling signal does not have eletricity, just like a NOT gate.

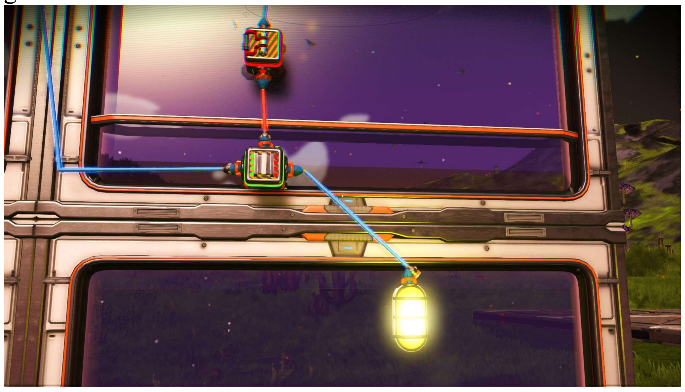


Image XV. A NOT gate in No Man's Sky

B. Seven-Segment display in No Man's Sky

A seven-segment display is made up of two things, the display part, created from LED lights that create segments of a decimal digit, and the circuity part that controls which segments would light up with a given input.

The display part of a seven-Segment display in No Man's Sky can be built using one of the several lights available within the game. For the purposes of this paper, the large Light Box will be used. There are many ways to build the display structure of the seven-segment display, for this paper a structure made entirely of the game's Light Boxes, following the way a seven-segment display is built as shown on Image IV will be used.

Now that it has been shown that it is possible to build logical gates within No Man's Sky and that the display part of the seven-segment display has been built, the only thing left is to use those logical gates to build the logical circuit for the seven-segment. Fortunately, we have simplified the seven-segment display circuitry into an effective and minimal form as shown on Image XII.

By simulating logic gates in No Man's Sky as shown in the previous section, and following the logical circuit shown on Image XII, the seven-segment circuit in No Man's Sky would appear like below.

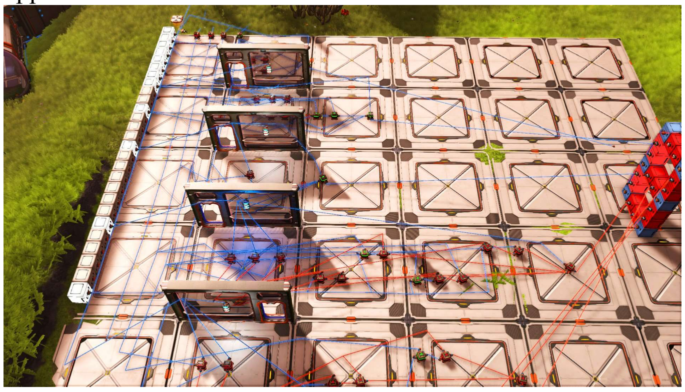


Image XVI. A Seven-segment circuitry in No Man's Sky

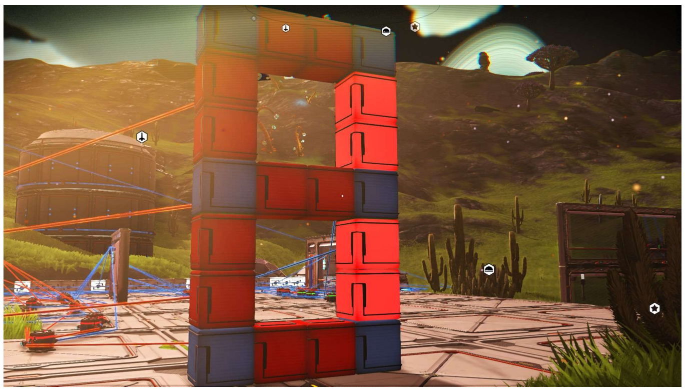


Image XVII. A Seven-segment display showing the number "1" in No Man's Sky

IV. Conclusion

Boolean algebra has many uses, one of them is creating minimal and effective logical circuits. This paper shows that one such use of Boolean algebra is creating a system in a game to create dynamic displays. The work in this paper can later be improved and expanded upon to create larger displays, and perhaps even to simulate more complex devices within the game of No Man's Sky.

VII. Acknowledgment

In this section I would like to raise my praises to Allah SWT, without whose mercy and grace I would not have the opportunity, strength, not determination to finish this paper. I would also like to thank Mr. Rinaldi Munir, class 01 IF 2018 Discrete Mathematics lecturer. Without his teachings, I would not have any clue as to the possibilities of discrete mathematics and would not have any knowledge capable of creating this paper.

References

[1] R. Munir, Matematika Diskrit, Bandung: Penerbit Informatika, ed. 4, 2010, ch. 7
[2] "Aljabar Boolean" by Rinaldi Munir. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2019-2020/Aljabar-Boolean-2019.pdf [accessed 5/12]
[3] http://www.ee.surrey.ac.uk/Projects/CAL/digital-logic/gatesfunc/index.html [accessed 5/12]
[4] https://www.electronicshub.org/bcd-7-segment-led-display-decoder-circuit/ [accessed 5/12]
[5] https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html [accessed 5/12]
[6] https://nomanssky.gamepedia.com/No_Man's_Sky [accessed 5/12]
[7] https://www.nomanssky.com/press/ [accessed 5/12]
[8] https://www.nomanssky.com/news/ [accessed 5/12]

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 6 Desember 2019

[illegible]

Hanif Muhamad Gana 13518127