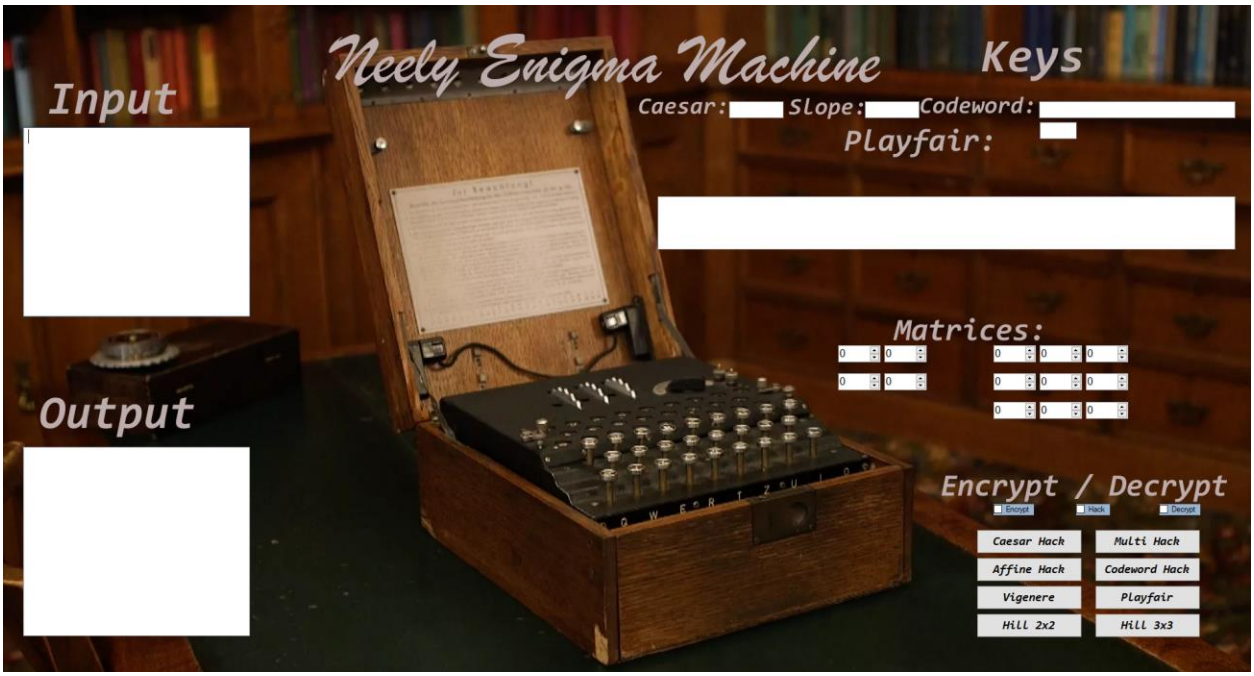
The Neely Enigma Machine

An instruction manual on the most secure and reliable cryptosystem application in human history*

*This statement may or may not be entirely fictitious, please do not actually encrypt your personal information using this device or any private key cryptosystems similar to this, unless your attackers are morons I guess…

1. Understanding the Menu Screen



When you first load the Neely Enigma Machine™, you may find yourself a bit overwhelmed by the sheer complexity of it, you may wonder “Woah! only a genius cold use this!”, but I assure you, even a monkey can use it. The Neely Enigma Machine™ is designed for simplicity.

The **Input** menu is where the string that you want to encrypt should be written down, the characters in your string can be any case but only include the characters, no spaces or separators should be used when writing down strings in the **Input** menu.

The **Output** menu is simply where the result of your encryptions and decryptions and hacks should go. Remember that if a particularly large result appears in the **Output** menu, you can navigate up and down

in it using the arrow keys or by highlighting downwards and upwards. (Note: It is possible to edit the contents of the **Output** menu, but you should refrain from doing so unless you want to ruin your brand-new string, also note that every encryption, decryption, and hack will erase the contents of the **Output** menu and replace it with a new string, so don't keep anything sacred in there, this isn't Notepad+++ for crying out loud).

The **Keys** menu contains the most important information in your usage of this cryptosystem, the private keys. Let us talk about them individually. (Note: any use of hacking will not involve this menu, as hacking is independent of private key information.)

The **Caesar** key slot is where the value for the intercept in the Affine cipher will be, as well as the value of the Caesar key. You can use any number between negative 2 billion-ish and positive 2 billion-ish for reasons involving the computer's memory, but when you do input any integer larger than 26, the modulo operation will be performed before encrypting and decrypting begins.

The **Slope** key slot is where the value for the slope in the Affine cipher and key of the Multiplicative cipher will be. Modulo operations will be performed and static memory is still between negative 2

billion-ish and 2 billion-ish. Remember to make the keys coprime with 26, if you don't, the application will not crash but I have made it so that it will yell at you for being so stupid, no offence…

The **Codeword** key slot is where the codeword for the Vigenere cipher and Codeword cipher will be. Lowercase Latin letters should be used to make the codeword. Underneath it is where the code-letter for the Codeword cipher will be. (Note: because explicit conversion in C# works the way it does, attempting to encrypt or decrypt the Codeword cipher without a lowercase code-letter will cause the application to crash, I am still debugging this problem but it has taken me days to even scratch at the surface of modifying explicit Convert.ToChar() calls.)

The **Playfair** key slot is where the key for the Playfair Cipher will be. For simplicity's sake, all you need to do is type in your Playfair square as you would read it left-to-right in lowercase letters. For example:

Q W E R T
Y U I O P
A S D F G
H K L Z X
C V B N M

Would appear as this on the screen:

The **Matrices** key slots are used in the Hill cipher, it should be quite intuitive to understand which one is the 2x2 and which one is the 3x3, as well as understanding how they should the input should be made. Remember to make sure your determinant is nonzero and coprime to 26, or else the program will yell at you for being stupid just like with the slope key.

The **Encrypt/Decrypt** menu gives you three options in the form of checkboxes, and there are 8 buttons, each with the name of a particular cipher on them. The buttons that end with the word “Hack” can be used to brute force hack the input string when the **Hack** checkmark is checked. Ones that do not end with “Hack simply perform decryption when the **Hack** checkmark is checked. Since the first 4 ciphers have 3 operations they can perform (encrypt, decrypt, hack) and the last 4 have 2 operations they can perform (encrypt, decrypt), this machine can perform a grand total of ***20*** unique operations with a given input string. There is so much to do with the Neely Enigma Machine™!!!

2. Understanding the Encryption

The encryption and decryption process for the Caesar, Multiplicative, Affine, Codeword, and Vigenere are fairly straightforward and have no special rules that apply to them in the Neely Enigma Machine™. However, the Playfair, Hill 2x2, and Hill 3x3 are all special in some ways, let's talk about that.

The Playfair cipher is strange because while the different conditions for pairs are always the same, the action you can take may differ. In this specific cipher

- Same Row: encrypts to the letters shifted to the right
- Same Column: encrypts to the letters shifted downwards
- Different Row and Col: Completes the rectangle as usual
- Same character: This cipher does not encrypt doubles at all, adding special characters ruins multi-layered ciphering and isn't generic from a coding perspective
- Note: To get around the problem with having to tear off one specific character from the Playfair block, I've made it so that "j" is always treated as an "i", this means that encrypting "Jaguar" and decrypting will result in the string "iaguar". However, it is easy to see what the message is meant to be if you are in the know, call it another form of "concealment" perhaps…

The Matrix ciphers and Playfair ciphers also have something unique to them that no other ciphers on this machine have, they encrypt in what are called “poly-graphs” (no, not the lie detector kind). These polygraphs are the pairs and trios we make from the plaintext and encrypt in groups so as to deter any kind of monoalphabetic frequency analysis. However, there are a few small problems that must be addressed concerning this type of encryption.

Because strings need to be divisible by the number of characters in each polygraph (digraph cipher string being even in length, trigraph cipher string being divisible by 3 in length), a few “x”s will be added to the input string near the end in order to make it divisible if that is required. This means that the string “helloworld”, when encrypted and decrypted by a Hill 3x3 cipher, will result in the string “helloworldxx” and “helloworldx” by a digraph cipher, but the message is still retained.

Another thing to note about multilayer encrypting is that the order of your encryptions must be ascending in terms of the length of the polygraphs in order to retain the message. This means that:

$$2x2\ Hill \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow Playfair \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow 3x3\ Hill \begin{Bmatrix} i \\ j \\ k \end{Bmatrix}$$

Is valid but:

$$2x2\ Hill \begin{pmatrix} i \\ j \end{pmatrix} \rightarrow 3x3\ Hill \begin{Bmatrix} i \\ j \\ k \end{Bmatrix} \rightarrow Playfair \begin{pmatrix} i \\ j \end{pmatrix}$$

Is not valid.

3. An Example of Encryption/Decryption

In order to prove the validity and to give an example of how to use the Neely Enigma Machine™, I have decided to encrypt and decrypt the famous line from Schierke in my favorite Manga series, *Berserk*.

"No matter how strong, for a human to fight a monster means he has submerged his humanity and transformed himself into a greater monster."

We will encrypt this with a Playfair, 2x2, and then finish it off with a 3x3! The keys for these ciphers are going to be:

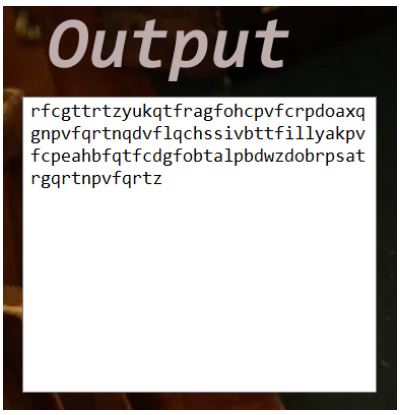
Q	W	E	R	T
Y	U	I	O	P
A	S	D	F	G
H	K	L	Z	X
C	V	B	N	M

7	8
11	11

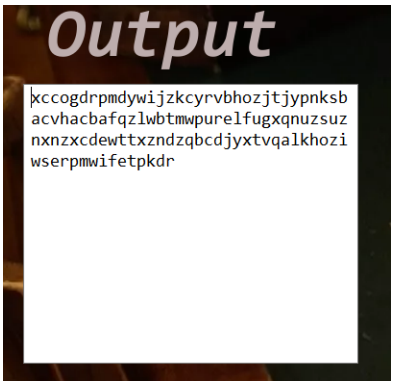
1	0	2
10	20	15
0	1	2

Inserting these keys and the quote into the Neely Enigma Machine™ should look a little like this:

Make sure to check the **Encrypt** checkmark and begin encrypting! Press the **Playfair** button first!

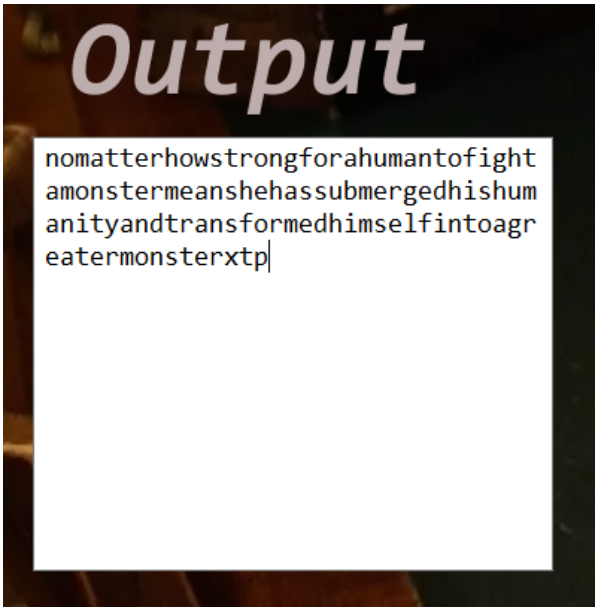


The output should look like this, we can now copy and paste this text into the **Input** menu and repeat this process with the **Hill 2x2** and **Hill 3x3** buttons…



The output now looks like this! This is our ciphertext for this message. We can now reverse the process by checking the **Decrypt** checkmark, and repeating our process of copy, paste, and

press. This time, in reverse order, **Hill 3x3**, **Hill 2x2**, and **Playfair**.



Now we arrive back to our original quote! This time with a little added “xtp” at the end. This is because of those little “x”s that I talked about earlier. Clearly, they do not stain the meaning of the quote at all.

4. Open Source Use and Conclusion

The Neely Enigma Machine is around 2,236 lines of code long, that is a fairly large project in terms of what I have done before, but nothing has engaged me more than this project, and nothing has engaged me more in my learning than this class.

Since copying my code into this Word document would probably kill MS Word quicker than Mr. Klingler can kill my Advanced Physics grade, I will be posting a fully open-sourced copy of the Neely Enigma Machine™ to GitHub, specifically this address:

https://github.com/FredRex/Cryptography-Class

Thank you for teaching me this course, this really was the best class I've ever taken. I can't believe that we can't have any more classes together, but I hope this project will at least help leave a lasting impression.

Farewell,

David Lynn