Enigma

Introducere

Proiectul a fost realizat folosind Unity Engine si un 3d model realizat de către **Science Museum Group.** Scopul acestui proiect este simularea unei maşini de criptare Enigma cu patru rotoare, dintre care un reflector, si zece mufe. Un utilizator poate sa modifice setarile de criptare (rotoare si mufe) si sa introduca text folosind tastatura fizica ca mai apoi sa vizualizeze cum textul s-a modificat prin procesul de criptare.
Acest program a fost realizat de Pirtac Andrei-Marius.

Interfata

General

La deschiderea programului apare meniul de tutorial in care sunt prezentate butoanele din program si care poate sa fie închis din stanga sus folosind butonul “X”.
In acest program sunt folosite următoarele butoane:

- X : pentru a închide un meniu deschis
- O : pentru a deschide un meniu
- ? : tutorial
- Exit : dreapta sus, inchide aplicatia
- X RESET ALL : şterge toate setarile introduse la categoria mufe

Tastatura

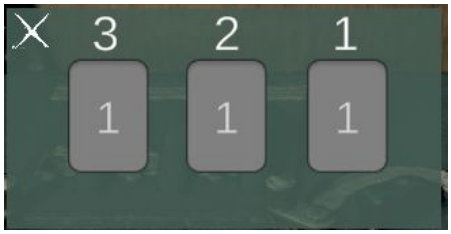
Pe modelul 3d se pot observa cerculete cu litere in ele deasupra tastaturii. Aceste cerculete sunt reprezentative pentru becurile care se aprind pentru a arata litera criptată.
La apasarea unei litere de pe tastatura fizica un bec se va “aprinde”, ca mai apoi sa se stinga cand tasta este in pozitia normala (ridicata).

Foaie pentru input si output

In dreapta se poate observa o foaie goala cu un buton “O” deasupra ei. Acest buton deschide meniul de vizualizare pentru mesajul introdus de la tastatura fizica împreuna cu varianta lui criptată.

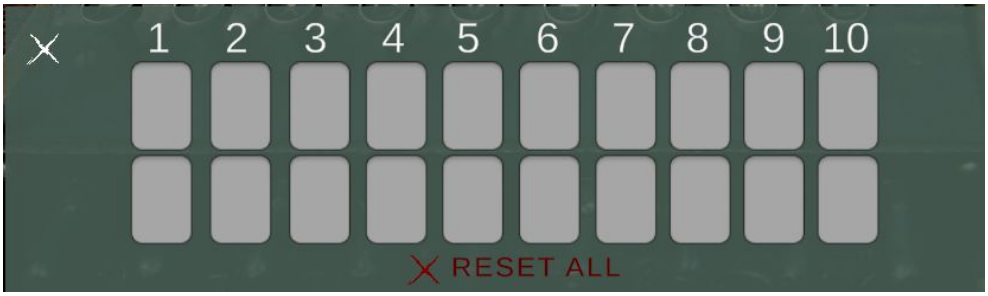
Rotoare

Deasupra tastaturii şi beculetelor se afla un buton “O” care deschide meniul pentru rotoare. În acest meniu utilizatorul poate sa vizualizeze schimbarea rotoarelor si sa le modifice pozitia. Programul o sa initializeze rotoarele cu valoarea 1.
Cum se poate observa în figura de mai jos, se pot schimba valorile pentru trei rotoare din patru pentru ca unul dintre ele este un reflector (vezi reflector); valoarea din dreapta este pentru primul rotor, cea din mijloc pentru al doilea şi cea din stanga pentru al treilea rotor.



Mufe

Sub tastatura se afla un buton “O” care deschide meniul pentru mufe unde un utilizator poate sa vadă şi sa modifice mufele. În acest meniu sunt prezente zece perechi de cate doua casete de input. În fiecare casuta se poate adaugă o litera care mai apoi sa fie legate la alta litera (perechea ei). O astfel de pereche este o coloana, doua casute una peste alta. Acest meniu contine şi un buton pentru resetarea valorilor. Cat timp acest meniu este deschis inputul de la tastatura fizica nu este citit pentru criptare.
Mufele sunt opţionale si sunt iniţializate cu “ “, astfel nu exista perechi la deschiderea programului.



Tutorial

In interfata de tutorial utilizatorul poate sa citeasca ce reprezinta fiecare buton de tipul “O”, cum funcţionează masina si cum se decripteaza mesajele. De asemenea este prezent şi un exemplu de decriptare in care utilizatorul primeşte mesajul “HFAWCJ” cu setarile 20|10|5 si AB DF OY JQ MN. Decriptarea ar trebui sa rezulte in mesajul “POTATO”.

Exit

In colţul din dreapta sus se afla butonul care închide aplicaţia.

Mod de utilizare

Deschidere

Este recomandat ca la prima deschidere a programului utilizatorul sa citeasca meniul de tutorial. Pentru a incepe criptarea unui mesaj utilizatorul trebuie sa inchida meniul de tutorial folosing butonul “X” din coltul din stanga sus.

Dupa inchiderea acestui meniu se poate incepe introducerea de la tastatura fizica a unui mesaj care va fi criptat cu setarile initiale si care poate sa fie vizualizat folosind meniul pentru input si output (foaia).

Rotoare

Pentru a schimba valoarea unui rotor se va selecta textul din casuta de input pentru acel rotor si se va modifica cu un numar cuprins intre 1 si 26 (inclusiv).

Pentru a modifica un rotor utilizatorul trebuie sa modifice si rotoarele care urmeaza dupa cel modificat.

Exemple pentru modificarea rotoarelor:

1. Iniţial rotoarele au valorile 1|3|7. Utilizatorul doreşte sa modifice valoarea primului rotor din 7 in 1. După aceasta modificare rotoarele o sa arate asa 1|3|1. Deşi utilizatorul a schimbat valoarea rotorului cum a planuit, acesta trebuie sa modifice si valoarea rotorului doi din 3 in 3 si apoi rotorului trei din 1 in 1, la final avand tot 1|3|1.
2. Iniţial rotoarele au valorile 2|24|1. Utilizatorul doreşte sa modifice valoarea rotorului doi din 24 în 3. După aceasta modificare rotoarele o sa arate asa 2|3|1. Deşi utilizatorul a schimbat valoarea rotorului cum a plănuit, acesta trebuie sa modifice şi valoarea rotorului trei din 2 in 2, dar nu trebuie sa modifice şi pentru rotorul unul, pentru ca acesta se afla în spatele rotorului modificat.

Dacă utilizatorul doreşte sa modifice toate rotoarele acestea trebuie modificate in ordine crescătoare, primul rotor, al doilea si apoi al treilea.

Cum se poate observa in exemple, deşi utilizatorul nu doreste sa modifice valoarea rotorului trei, aceasta trebuie modificata pentru ca după modificarea rotorului doi se executa un cerc complet de rotiri care o sa schimbe implicit si valoarea rotorului trei (vezi constructia rotoarelor).

Meniul cu valorile rotoarelor poate sa ramana deschis in timpul introducerii literelor pentru criptare, astfel se observa cum la fiecare input nou valorile se schimba cu +1.

Mufe

Dupa deschiderea meniului de mufe inputul de la tastatura fizica nu o sa mai fie citit pentru criptare, acest lucru fiind datorat faptului ca programul se asteapta ca utilizatorul sa modifice mufele folosind un input de tipul litera, care este de asemenea folosit in mod normal pentru criptare.

In acest meniu se pot modifica casutele goale, in fiecare fiind acceptata o litera care nu este deja intr-o pereche. Daca inputul introdus intr-o casuta este un text care cuprinde mai multe litere sau o litera deja intr-o pereche sau aceeasi litera ca perechea ei, casuta pentru aceasta litera o sa se goleasca impreuna cu casuta pentru pereche.

Dupa modificarea acestor casute, pentru schimbarea lor in alte valori trebuie folosit butonul “RESET ALL” pentru ca valorile de conexiuni sa se reseteze.

Nu este permisa introducerea unui numar sau simbol special.

Dupa ce utilizatorul termina de facut modificarile, pentru a reincepe criptarea unui text, trebuie sa inchida acest meniu.

Mecanismul

Introducere

Masina fizica enigma folosea cabluri care se conectau intr-un mod complex de la o litera de pe tastatura la un bec cu simbolul unei litere. Cand tasta era apasata circuitul era pornit si curentul aprindea un bec, astfel apasand o tasta era “returnata” o alta litera care era valoarea criptata pentru tasta apasata.

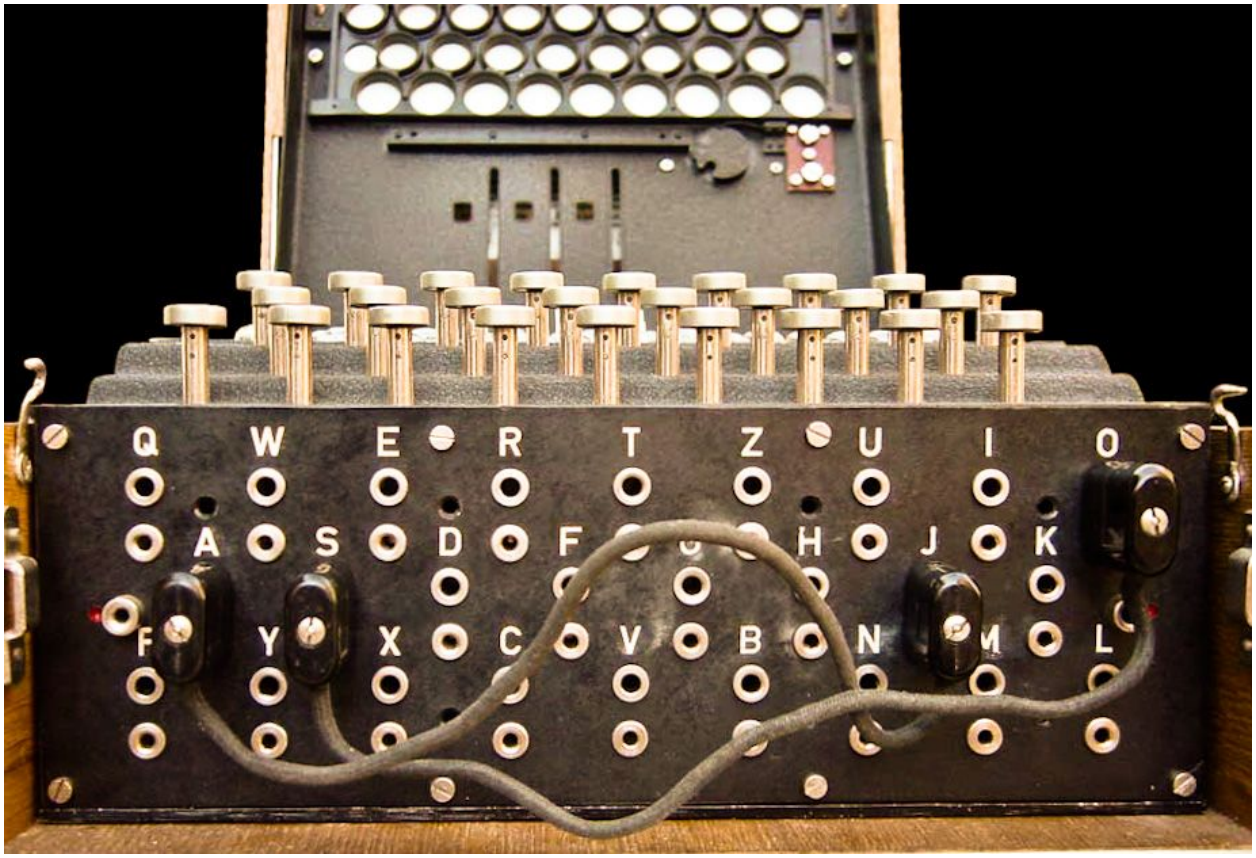
Procesul de criptare urmeaza acesti pasi:

1. Se apsa o tasta
2. Curentul ajunge la mufe unde daca este conectata o mufa litera introdusa se transforma in litera cu care este conectata
3. Curentul trece mai departe la rotoare unde intra cu valoarea precedenta. Aici fiecare litera este conectata la alta litera printr-un cablu, si astfel valoarea precedenta trece mai departe in rotorul urmator folosind noua valoarea la care este legata.
4. In al doilea rotor se repeta procus, litera precedenta trece print-o conexiune in alta litera si este transmisa mai departe
5. In al treilea rotor se repeta identic cu al doilea rotor.
6. In ultimul rotor numit si reflector valorile sunt conectate una la alta, circuitul este astfel intors invers.
7. Se intra in al treilea rotor cu valoarea obtinuta din reflector si se trece prin proces dar in mod invers.
8. Se repeta pasul 7 prin rotoarele doi si unu
9. Dupa ce valoarea iese din rotoare aceasta se intoarce la mufe unde, daca este conectata la ceva, se transforma in valoarea la care este conectata
10. Dupa tot circuitul se ajunge la bec care se aprinde in functie de ce valoarea este la final.

In continuare vor fi prezentate componentele in mod individual.

Mufe

Aceasta este prima componenta prin care trece litera introdusa.

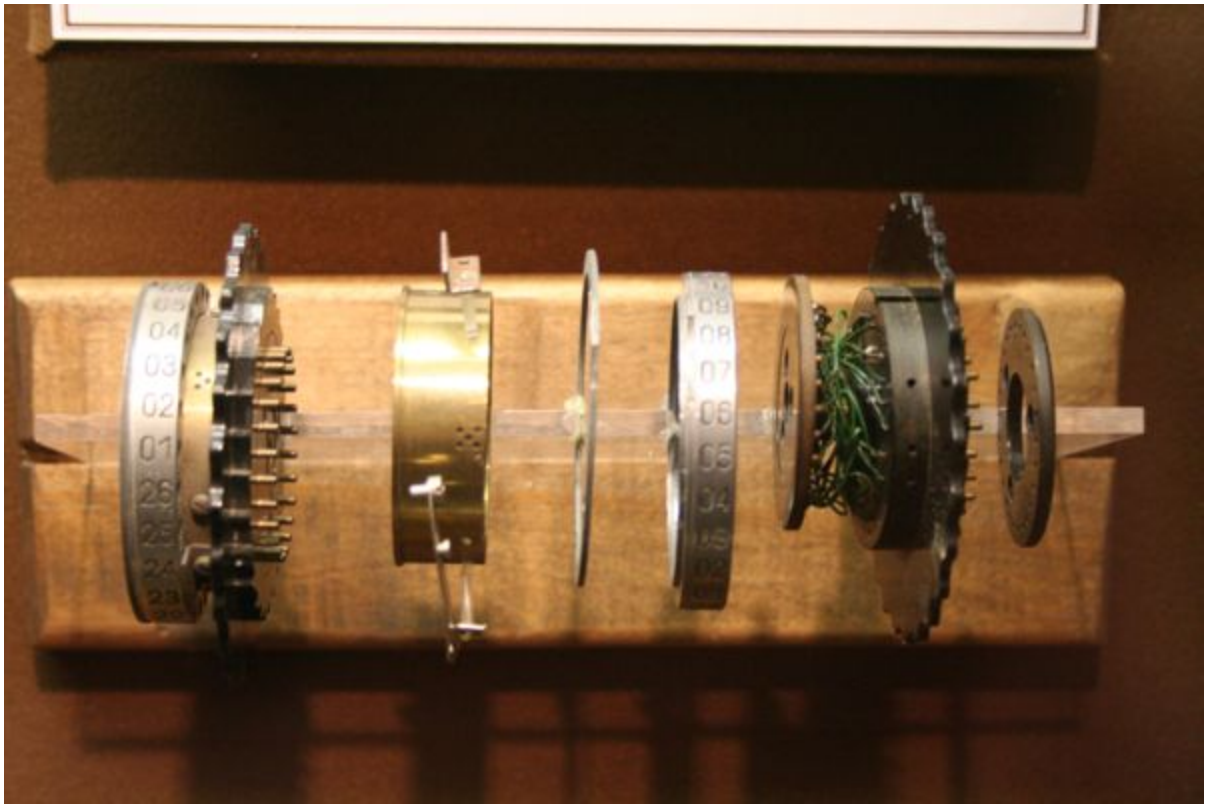


In cazul pozei de mai sus litera A este conectata cu litera J; si litera S este conectata cu litera O.

Daca presupunem ca am incepe criptarea literei A folosind aceasta masina, atunci la acest pas am transforma A in J.
Daca in schimb am presupune ca am incepe cu litera Q, la acest pas nu am face nimic pentru ca nu exista o legatura intre Q si alta litera.

Aceasta componenta este doar o schimbare evidenta, dar care adauga extrem de multe optiuni in setarile de decriptare si implicit de criptare.

Rotoare



Aceste componente constituie partea complexa de criptare. In poza de deasupra se poate observa cum arata un astfel de rotor.

Fiecare rotor are lista alfabetica de litere (ABCD...Z) si o lista cu conexiuni (FTG...K). Exista multe rotoare folosite in masinile enigma, acestea au fost schimbat destul de des pentru a schimba complet ce output poti sa primesti si au fost de asemenea folosite pe diverse domenii, spre exemplu aviatia folosea alte combinatii decat trupele de sol.

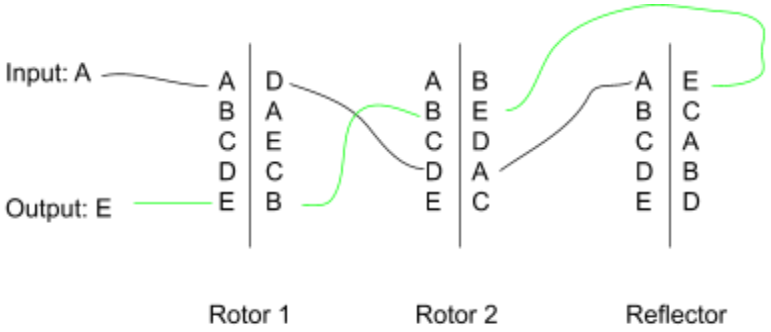
Rotoarele folosite in acest program sunt urmatoarele:

Rotorul unu `PEZUOHXSCVFMTBGLRINQJWAYDK`
Rotorul doi `ZOUESYDKFWPCIQXHMVBLGNJRAT`
Rotorul trei `EHRVXGAOBQUSIMZFLYNWKTPDJC`
Reflector `IMETCGFRAYSQBZXWLHKDVUPOJN`

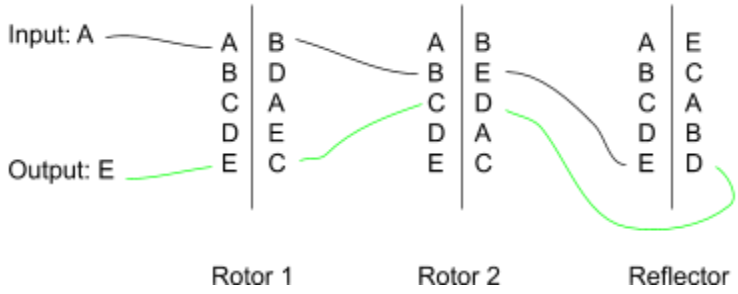
Aceste rotoare au fost folosite in 1939.

Dupa ce o litera a fost criptata primul rotor (cel din dreapta) se invarte cu o iteratie. Dupa valoarea 26 se intoarce la 1 si se invarte urmatorul rotor (cel din mijloc). Astfel daca ai vrea sa modifici primul rotor de la valoarea 2 la 1 trebuie sa faci un cerc complet si o sa modifici si valoarea rotorului doi. Pentru ca nu voiai sa modifici valoarea rotorului doi sa spunem ca din 3 in 4 cat a ajuns sa fie dupa ce primul rotor a fost modificat, trebuie sa il modifici si pe acesta cu o rotatie completa ca sa ajungi inapoi la 3, si astfel modifici si ultimul rotor fara sa vrei. Programul foloseşte acelaşi principiu, din acest motiv modificarea rotoarelor trebuie facuta de la dreapta la stanga şi completa.

Mai departe va fi prezentat modul de funcţionare printr-o reprezentare grafica exemplificativa.



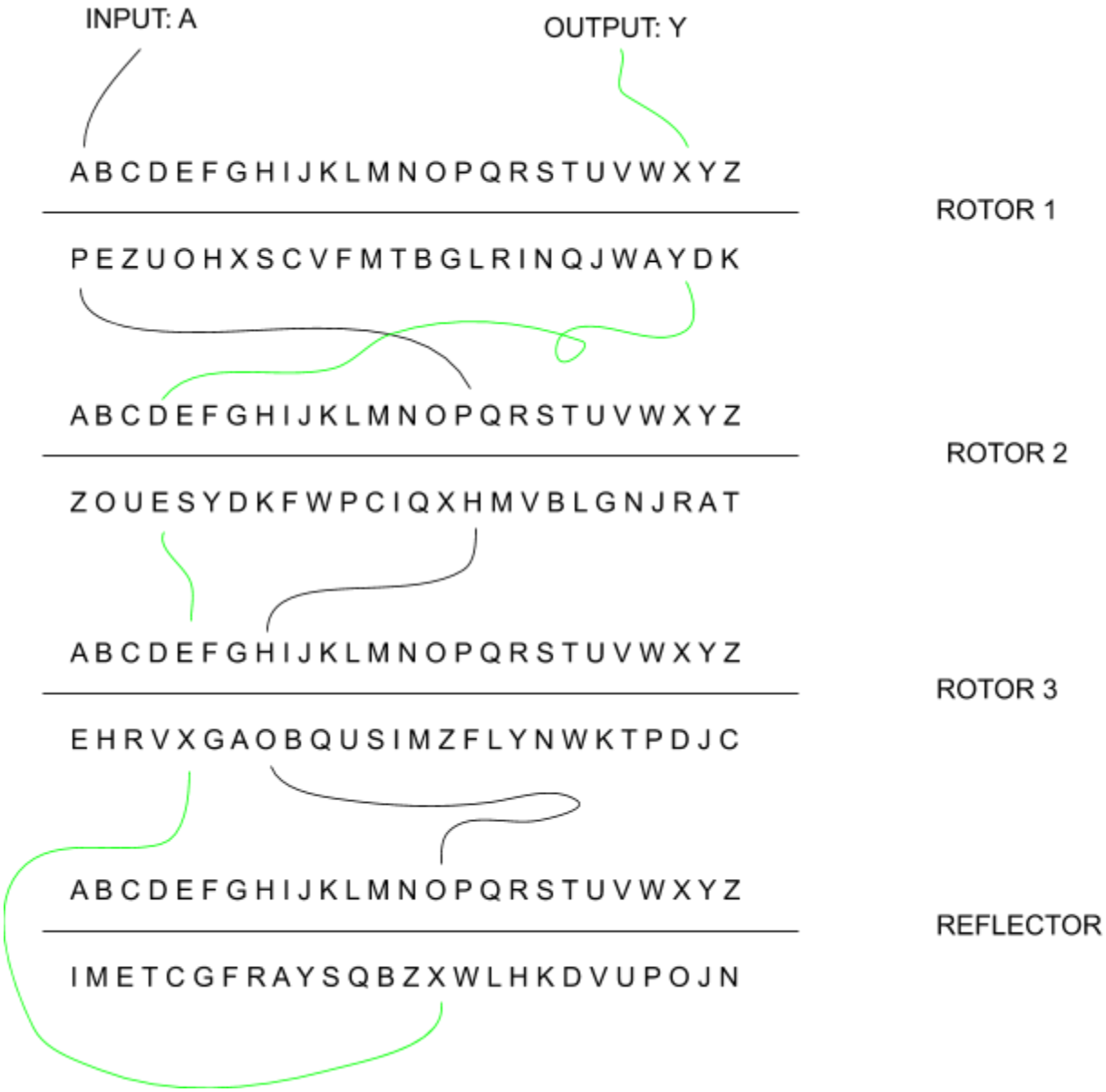
Prima iteratie



A doua iteratie
(primul rotor s-a rotit cu o litera)

Acesta este doar un exemplu pentru înţelegerea mecanismului, în realitate am fi obtinut un alt output.

Un exemplu cu setările inițiale 1|1|1 și fără mufe pe rotoarele folosite în program:



Procesul din imaginea de mai sus este:
Input: A;
A->P; P->H; H->O;
Intram in reflector cu O care devine X si ne intoarcem;
X->E; E->D; D->Y;
Output: Y;

Traducerea in cod

Variabile

```
//interactibles
[SerializeField]
GameObject[] lights;
[SerializeField]
Text input;
[SerializeField]
Text output;
[SerializeField]
GameObject inputText1; //input text for rotor 1
[SerializeField]
GameObject inputText2; //input text for rotor 2
[SerializeField]
GameObject inputText3; //input text for rotor 3
```

Acestea sunt variabilele care leaga elementele de UI de program.

Scurta descriere pentru tipurile folosite:
[SerializeField] = pastreaza variabila private dar ne da acces la ea din unity editor
GameObject = Tipul general de element prezent in scena din editorul Unity
Text = tipul de element folosit pentru UI Text.

Vectorul lights contine luminile care se “aprind” pentru output. “Aprinderea” si “stingerea” este de fapt activarea si dezactivarea obiectului.

Input, output sunt casutele de text unde scriem input si respectiv output.

InputText1, InputText2, InputText3 sunt casutele de input pentru rotaţia rotoarelor.

```
//Rotor settings
[SerializeField]
string r1 = "PEZUOHXSCVFMTBGLRINQJWAYDK";
[SerializeField]
string r2 = "ZOUESYDKFWPCIQXHMVBLGNJRAT";
[SerializeField]
string r3 = "EHRVXGAOBQUSIMZFLYNWKTPDJC";
[SerializeField]
string r4 = "IMETCGFRAYSQBZXWLHKDVUPOJN"; //reflector

//rotor vectors we will use
int[] rotor1;
int[] rotor2;
int[] rotor3;
int[] rotor4;

//the rotation for each rotor
int r1Rotation=1;
int r2Rotation=1;
int r3Rotation=1;
```

Acestea sunt variabilele pentru rotoare.

r1, r2, r3, r4 sunt stringuri care contin setarile pentru rotoare in varianta lor de tip text. Am folosit aceasta varianta pentru a usura testarea si schimbarea din editor a setarilor, aceasta se poate face simplu cu copy-paste de pe wiki. Trebuie precizat ca r4 este reflectorul.

rotor1, rotor2, rotor3, rotor4 sunt vectori care contin valorile int ale setarilor, transformate din r1...r4 folosind o functie care va fi prezentata mai tarziu.

r1Rotation, r2Rotation, r3Rotation sunt variabile care stocheaza rotatia rotoarelor.

```
//checks if another menu uses the alphabetical input
bool plugsOn = true;
0 references
public void changePlugsOn()
{
    if (plugsOn) plugsOn = false;
    else plugsOn = true;
}

//a matrix for the connected plugs
public int[,] plugs;
```

Acestea sunt variabilele pentru mufe

plugsOn verifica daca exista un meniu deschis care foloseste ca input litere, si este acompaniat de changePlugsOn(), o functie apelata de butoanele care deschid/inchid meniurile ce folosesc un astfel de input.

plugs este o matrice ce stocheaza 1 daca [i,j] are o conexiune, unde i si j sunt variantele numerice ale literelor; sau 0 daca nu exista o astfel de conexiune.

```
//the output letter should be a global variable because we use it in update and we cand initiate it each frame
int letterOutput = 0;        //numeric version
string letterOutputS = "a";  //string version
```

Acestea sunt variabilele de output

Nu putem sa le stocam local pentru ca se vor reintializa in fiecare frame.

letterOutput este varianta numerica

letterOutputS este varianta de tip string.

Initializarea datelor

```
//Start function is called when the user starts the program and here we initialize variables
Unity Message | 0 references
void Start()
{
    plugs = new int[26,26];
    inputText1.GetComponent<TMP_InputField>().text = "1";
    inputText2.GetComponent<TMP_InputField>().text = "1";
    inputText3.GetComponent<TMP_InputField>().text = "1";
    input.text = "Here is the input text";
    output.text = "Here is the output text";
    InitializeRotors();
}
```

Explicatie:
inputText1.GetComponent<TMP_InputField>().text accesează din obiectul (GameObject) inputText1 componenta (.GetComponent) de tipul (<TMP_InputField>()) pentru a schimba proprietatea text (.text).

In initializare folosim functia InitializeRotors() care transforma stringul de setari in vector de numere.

```
//transform the rotor settings to int so we an use it later as int instead of string
1 reference
void InitializeRotors()
{
    rotor1 = new int[26];
    rotor2 = new int[26];
    rotor3 = new int[26];
    rotor4 = new int[26];



    int count = 0;
    foreach(char letter in r1)
    {
        rotor1[count] = letterToNumber(letter);
        count++;
    }
    count = 0;
    foreach (char letter in r2)
    {
        rotor2[count] = letterToNumber(letter);
        count++;
    }
    count = 0;
    foreach (char letter in r3)
    {
        rotor3[count] = letterToNumber(letter);
        count++;
    }
    count = 0;
    foreach (char letter in r4)
    {
        rotor4[count] = letterToNumber(letter);
        count++;
    }

}
```

(pentru letterToNumber() vezi functii ajutatoare)

Aceasta functie doar parcurge stringurile r1, r2, r3, r4 si vectorii pentru rotoare si pune in vectori varianta numerica a char ului respectiv.

Algoritmul de criptare

Algoritmul o sa fie prezentat pe pasi si apoi ca intreg.

Pasul 1 Verificarea inputului

Pentru a cripta o litera meniurile ce folosesc input alfabetic trebuie sa fie inchis (!plugsOn).

Inainte de a incepe verificarea inputului tratam cazul special “ “ (space) pentru a permite utilizatorului sa puna spatiu in text.

Pentru a verifica daca utilizatorul a apasat o litera contruim un vector de litere si verificam la fiecare frame daca una din ele a fost apasata

```
void Update()
{
    char[] letters=new char[] {'q','w','e','r','t','y','u','i','o','p','a','s','d','f','g','h','j','k','l','z','x','c','v','b','n','m'};

    if(!plugsOn)
    {

        if (Input.GetKeyDown("space") == true)
        {
            if (input.text == "Here is the input text") input.text = " ";
            else input.text += " ";

            if (output.text == "Here is the output text") output.text = " ";
            else output.text += " ";
        }

        foreach (char letter in letters)
        {
            string letterS = letter.ToString();
            int letterN = letterToNumber(letter);


            if (Input.GetKeyDown(letterS) == true)
            {
```

Daca putem sa criptam si utilizatorul introduce spatiu doar scriem spatiu in input si output.

```
if (input.text == "Here is the input text") input.text = " ";
else input.text += " ";

if (output.text == "Here is the output text") output.text = " ";
else output.text += " ";
```

Aceasta bucata de cod verifica daca a mai fost ceva introdus inainte sau nu.

```
string letterS = letter.ToString();
int letterN = letterToNumber(letter);
```

letterS este varianta de tip string a literei pe care o verificam acum.
letterN este varianta numerica a literei pe care o verificam acum.

Construim vectorul de litere letters de tip char[], nu de string pentru ca avem nevoie sa transformam in int pentru calcule, iar transformarea din string in int este mai greu de realizat decat cea din char in int.

```
if (Input.GetKeyDown(letterS) == true)
```

Verificam daca litera curenta a fost apasata. Input.GetKeyDown se activeaza o singura data cand litera a fost apasata complet.

Pasul 2 Verificam daca exista o mufa conectata

```
letterN = CheckPlugs(letterN);

//check if there is a plug connected to the letter
2 references
int CheckPlugs(int letter)
{
    for(int i=0;i<26;i++)
    {
        if (plugs[letter, i] == 1) return i;
    }
    return letter;

}
```

Schimbam letterN cu CheckPlugs(letterN), astfel daca exista o mufa conectata letterN devine litera cu care este conectata, iar daca nu, ramana aceeaşi valoare.

Pasul 3 Folosim rotoarele

```
letterOutput = UseRotors(letterN, true);
```

Salvam in letterOutput valoarea obtinuta dupa folosirea rotoarelor cu valoarea initiala letterN si cu modificarea rotatiei rotoarelor din interfata (true).

```
//function that goes through the rotors to change the letter
1 reference
int UseRotors(int letter,bool update)
{
    letter = rotor1[letter];
    letter = rotor2[letter];
    letter = rotor3[letter];
    letter = rotor4[letter];


    int n = 0;
    foreach (int l in rotor3)
    {
        if (l == letter)
        {
            letter = n;
            break;
        }
        n++;
    }

    n = 0;
    foreach (int l in rotor2)
    {
        if (l == letter)
        {
            letter = n;
            break;
        }
        n++;
    }

    n = 0;
    foreach (int l in rotor1)
    {
        if (l == letter)
        {
            letter = n;
            break;
        }
        n++;
    }



    rotateRotors(update);
    return letter;
}
```

Sa analizam pe rand elementele din aceasta functie.

```
letter = rotor1[letter];
letter = rotor2[letter];
letter = rotor3[letter];
letter = rotor4[letter];
```

Letter devine pe rand valoarea din rotor aflata la litera curenta. Aceasta bucata de cod realizarea schimbarea prezentata aici cu liniile negre.

```
int n = 0;
foreach (int l in rotor3)
{
    if (l == letter)
    {
        letter = n;
        break;
    }
    n++;
}

n = 0;
foreach (int l in rotor2)
{
    if (l == letter)
    {
        letter = n;
        break;
    }
    n++;
}

n = 0;
foreach (int l in rotor1)
{
    if (l == letter)
    {
        letter = n;
        break;
    }
    n++;
}
```

Aceasta bucata de cod face schimbaarile prezentate la functionarea rotoarelor cu linia verde, adica intoarcerea dupa ce valoarea a fost cea din reflector. Pentru fiecare litera intr-un rotor verificam ce index are litera curenta. Acel index este litera din ordinea alfabetica care corespunde cu litera curenta.

```
rotateRotors(update);
return letter;
```

La final doar rotim rotoarele si returnam litera dupa ce a fost schimbata. Acest return duce valoarea in letterOutputN.

rotateRotors(update), in cazul nostru update este true, adica dorim sa schimbam partea de UI care arata rotatia.

```
void rotateRotors(bool update)
{
    int aux;
    if (r1Rotation < 26)
    {
        r1Rotation++;
        aux = rotor1[0];
        for(int i=1;i<26;i++)
        {
            rotor1[i - 1] = rotor1[i];
        }
        rotor1[25] = aux;
    }
    else
    {
        r1Rotation = 1;
        aux = rotor1[0];
        for (int i = 1; i < 26; i++)
        {
            rotor1[i - 1] = rotor1[i];
        }
        rotor1[25] = aux;

        //----------R2-----------
        if (r2Rotation < 26)
        {
            r2Rotation++;
            aux = rotor2[0];
            for (int i = 1; i < 26; i++)
            {
                rotor2[i - 1] = rotor2[i];
            }
            rotor2[25] = aux;
        }
        else
        {
            r2Rotation = 1;
            aux = rotor2[0];
            for (int i = 1; i < 26; i++)
            {
                rotor2[i - 1] = rotor2[i];
            }
            rotor2[25] = aux;
```

```
            //--------R3---------
            if (r3Rotation < 26)
            {
                r3Rotation++;
                aux = rotor3[0];
                for (int i = 1; i < 26; i++)
                {
                    rotor3[i - 1] = rotor3[i];
                }
                rotor3[25] = aux;
            }
            else
            {
                r3Rotation = 1;
                aux = rotor3[0];
                for (int i = 1; i < 26; i++)
                {
                    rotor3[i - 1] = rotor3[i];
                }
                rotor3[25] = aux;
            }
        }
    }

    if (update)
    {
        inputText1.GetComponent<TMP_InputField>().text = r1Rotation.ToString();
        inputText2.GetComponent<TMP_InputField>().text = r2Rotation.ToString();
        inputText3.GetComponent<TMP_InputField>().text = r3Rotation.ToString();
    }

}
```

In aceasta functie rotim primul rotor, si daca acesta ajunge inapoi la 1, il rotim si pe al doilea cu o iteratie, daca acea iteratie ajunge inapoi la 1 pentru al doilea rotor atunci rotim si al treilea rotor cu o iteratie.
La final schimbam partea de UI daca aceasta trebuie schimbata

Pasul 4 Verificam iar mufele si aprindem beculetul

```
letterOutput = CheckPlugs(letterOutput);
```

Verificam iar daca exista o mufa conectata, dar de data asta la output, si, dacă este cazul schimbat outputul cu litera conectata.
Aprindem beculetul de la litera obtinuta.

```
lights[letterOutput].SetActive(true);
```

Pasul 5 Actualizam elementele de UI

```
if (Input.GetKeyUp(letterS) == true)
{

    letterOutputS = numberToLetter(letterOutput);
    lights[letterOutput].SetActive(false);

    if (input.text == "Here is the input text") input.text = letterS;
    else input.text += letterS;

    if (output.text == "Here is the output text") output.text = letterOutputS;
    else output.text += letterOutputS;

}
```

Daca utilizaatorul ridica tasta apasata anterior convertim rezultatul numeric in string si il afisam in casuta de output, cu inputul adaugat in casuta de input. De asemenea aici oprim beculetul aprins anterior.

Codul complet pentru criptarea unei litere

```
// Update is called once per frame
Unity Message | 0 references
void Update()
{
    char[] letters=new char[] {'q','w','e','r','t','y','u','i','o','p','a','s','d','f','g','h','j','k','l','z','x','c','v','b','n','m'};

    if(!plugsOn)
    {

        if (Input.GetKeyDown("space") == true)
        {
            if (input.text == "Here is the input text") input.text = " ";
            else input.text += " ";

            if (output.text == "Here is the output text") output.text = " ";
            else output.text += " ";
        }

        foreach (char letter in letters)
        {
            string letterS = letter.ToString();
            int letterN = letterToNumber(letter);


            if (Input.GetKeyDown(letterS) == true)
            {
                letterN = CheckPlugs(letterN);

                letterOutput = UseRotors(letterN, true);

                letterOutput = CheckPlugs(letterOutput);

                lights[letterOutput].SetActive(true);
            }
            if (Input.GetKeyUp(letterS) == true)
            {

                letterOutputS = numberToLetter(letterOutput);
                lights[letterOutput].SetActive(false);

                if (input.text == "Here is the input text") input.text = letterS;
                else input.text += letterS;

                if (output.text == "Here is the output text") output.text = letterOutputS;
                else output.text += letterOutputS;

            }
        }
    }


}
```

Setari

Rotoare

```
//function used to update the rotors after the user changes the value using the input box in UI
//we dont update the UI with the new version because the user changed it
0 references
public void updateRotors()
{

    while (inputText3.GetComponent<TMP_InputField>().text != r3Rotation.ToString())
    {
        rotateRotors(false);
    }
    while (inputText2.GetComponent<TMP_InputField>().text != r2Rotation.ToString())
    {
        rotateRotors(false);
    }
    while (inputText1.GetComponent<TMP_InputField>().text != r1Rotation.ToString())
    {
        rotateRotors(false);
    }
}
```

Chemam rotateRotors(false) (fara sa actualizam si UI-ul) pana cand rotatia este egala cu valoarea introdusa.

Mufe

Setarile de la mufe se afla pe fiecare mufa in parte si actualizeaza din exterior variabila plugs[,].

```
[SerializeField]
GameObject enigma=null;
[SerializeField]
GameObject connectedTo = null;
```

Fiecare capat de mufa are un capat conectat (connectedTo) si o referinta la enigma care stocheaza plugs[,].

```
this.GetComponent<TMP_InputField>().text = " ";
```

Initializam textul din acest capat de mufa cu “ “.

De fiecare data cand utilizatorul incearca sa schimbe valoarea din casuta de input pentru o litera se cheama urmatoarea functie:

```
0 references
public void TryToChange()
{
    if(connectedTo.GetComponent<TMP_InputField>().text!=" ")
    {
        if (this.GetComponent<TMP_InputField>().text.ToCharArray().Length == 1)
        {
            int j = letterToNumber(this.GetComponent<TMP_InputField>().text.ToCharArray()[0]);
            int k = letterToNumber(connectedTo.GetComponent<TMP_InputField>().text.ToCharArray()[0]);

            bool check = true;

            if (enigma.GetComponent<readInputScript>().plugs[j, k] == 1) check = false;
            else if (j == k) check = false;
            else if (this.GetComponent<TMP_InputField>().text.ToCharArray().Length != 1) check = false;
            else if (connectedTo.GetComponent<TMP_InputField>().text.ToCharArray().Length != 1) check = false;
            else check = true;

            if (check == true)
            {
                enigma.GetComponent<readInputScript>().plugs[j, k] = 1;
                enigma.GetComponent<readInputScript>().plugs[k, j] = 1;
            }
            else
            {
                this.GetComponent<TMP_InputField>().text = " ";
                connectedTo.GetComponent<TMP_InputField>().text = " ";
            }
        }
    }
}
```

Sa analizam pe parti.

```
if(connectedTo.GetComponent<TMP_InputField>().text!=" ")
```

Verifica daca litera introdusa are pe campul partener ceva introdus. Daca are putem sa verificam daacaa pastram inputul sau nu, daca nu are atunci nu facem nimic.

```
if (this.GetComponent<TMP_InputField>().text.ToCharArray().Length == 1)
```

Verificam daca a fost introdusa o singura litera sau mai multe. Daca este una singura putem sa continuam.

```
int j = letterToNumber(this.GetComponent<TMP_InputField>().text.ToCharArray()[0]);
int k = letterToNumber(connectedTo.GetComponent<TMP_InputField>().text.ToCharArray()[0]);
```

J si k sunt valorile numerice ale literelor introduse.

```
bool check = true;
```

Verificam daca putem sa pastram acest input.

```
if (enigma.GetComponent<readInputScript>().plugs[j, k] == 1) check = false;
else if (j == k) check = false;
else if (this.GetComponent<TMP_InputField>().text.ToCharArray().Length != 1) check = false;
else if (connectedTo.GetComponent<TMP_InputField>().text.ToCharArray().Length != 1) check = false;
else check = true;
```

Verificam daca perechea a mai fost introdusa, daca a fost introdusa o litera de doua ori si daca ambele au ca format o singura litera. Daca inputul respecta toate conditiile necesare atunci check o sa fie true, daca incalca oricare dintre ele, atunci o sa fie false.

```
if (check == true)
{
    enigma.GetComponent<readInputScript>().plugs[j, k] = 1;
    enigma.GetComponent<readInputScript>().plugs[k, j] = 1;
}
else
{
    this.GetComponent<TMP_InputField>().text = " ";
    connectedTo.GetComponent<TMP_InputField>().text = " ";
}
```

Daca este true atunci schimbam valoarea din plugs[j,k]=plugs[k,j] in 1. Matriceea este simetrica cu diagonala secundara linie de simetrie. Diagonala principala o sa fie mereu 0.
Daca nu sunt indeplinite conditiile atunci doar scriem “ “ in locurile in care inputul nu a fost bun.

Pentru mufe avem si optiunea de a reseta valorile:

```
0 references
public void ResetPlugs()
{
    enigma.GetComponent<readInputScript>().plugs = new int[26, 26];
    foreach (GameObject plug in plugs)
    {
        plug.GetComponent<TMP_InputField>().text = " ";
    }
}
```

Aici reinitializam plugs[,] si scriem peste tot “ “.

Functii ajutatoare

```
//simple conversion from int to string for a letter
1 reference
string numberToLetter(int n)
{
    const string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

    return letters[n].ToString();
}
```

Schimbare din int in string.

```
//simple conversion from char to int forr a letter
5 references
int letterToNumber(char c)
{
    return ((int)c % 32)-1;
}
```

Schimbare din char in int. Aici folosim -1 pentru a incepe cu a=0 nu a=1.

Note

Daca la rotoare se introduce o alta valoare care nu este cuprinsa intre 1 si 26 programul o sa intre într-un loop infinit pentru ca rnRotation cu n={1,2,3} are valori doar in intervalul [1,26].

```
while (inputText3.GetComponent<TMP_InputField>().text != r3Rotation.ToString())
{
    rotateRotors(false);
}
```

Aceasta bucata de cod exemplifica situatia unui loop infinit.

La meniul de mufe nu exista o modalitata usoara de a schimba conexiunile. O modalitate mai buna ar putea fi implementata.
Daca utilizatorul ar incerca sa schimbe o conexiune deja scrisa atunci conexiunea deja scrisa ar ramane activa si ar fi adaugata si conexiunea noua depasind astfel numarul de 10 conexiuni si blocand litere in conexiuni pe care nu poate sa le mai vada.

Ar putea fi adaugata o modalitate de resetare a casutelor de input si output.

Daca sunt apasate mai multe taste simultan atunci outputul o sa fie gresit iar luminile o sa se blocheze pe pozitia activa. Aceest lucru se datoreaza faptului ca testele sunt facute in functia Update() care se updateaza la fiecare frame, astfel daca doua taste sunt apasate valoarea de output de la una dintre ele o sa fie pierduta.

Bibliografie

Programul folosit pentru realizarea aplicatiei: unity
Modelul 3d folosit: Enigma
Poza folosita pentru mufe: https://calculate.org.au/2015/02/03/crack-enigma-code/
Poza folosita pentru rotoare: https://en.wikipedia.org/wiki/Enigma_rotor_details

Linkuri folosite pentru intelegerea mecanismului si traducerea sa in aplicatie:
https://en.wikipedia.org/wiki/Enigma_rotor_details
https://en.wikipedia.org/wiki/Enigma_machine
https://www.youtube.com/watch?v=G2_Q9FoD-oQ&ab_channel=Numberphile
https://www.youtube.com/watch?v=ASfAPOiq_eQ&t=323s&ab_channel=WorldScienceFestival
https://youtu.be/2D2bJWHvqJo

Materiale recomandate pentru folosirea programului Unity:
https://docs.unity3d.com/Manual/index.html
https://www.youtube.com/user/Brackeys
https://www.youtube.com/watch?v=GhQdlIFylQ8&ab_channel=freeCodeCamp.org