UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Arqueología informática: la criptografía clásica con Scratch

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Le Danny Yang
Tutor: Xavier Molero Prieto

Curso 2016-2017

Agradecimientos

A mis padres, por el apoyo moral y económico. Y por supuesto a mis hermanos Wei Lesley, Shanshan y Tingting por vuestra confianza depositada en mí a lo largo de toda la carrera universitaria.

A mis profesores y en especial a mi tutor Xavier Molero Prieto, por todo lo que he aprendido gracias a vosotros.

A todos mis compañeros y amigos de la universidad, sobre todo a Guillem Matías Guillén y Franco Julián García Montero.

Gracias.

Resumen

El trabajo presentado se centra principalmente en la elaboración de cuatro cifrados clásicos de la historia de la criptografía (Escítala espartana, Tablero de Polibio, Cifrado de César y Disco de Alberti) mediante el uso de Scratch como lenguaje de programación. Además incluiremos un programa simple explicando el concepto de ataque de fuerza bruta, el cual se sigue usando actualmente dentro de las técnicas del criptoanálisis. Dichos cifrados se incluirán en el Museo de Informática de la Escola Tècnica Superior d'Enginyeria Informàtica con el propósito de enseñar al público adolescente de forma didáctica la importancia de la seguridad con ejemplos prácticos donde ellos sean quienes interactúen con el programa. En la primera parte del documento se contará cronológicamente la historia de la criptografía desde el primer cifrado conocido históricamente (Escítala espartana), hasta los algoritmos que existen hoy en día para mantener la seguridad de los mensajes que enviamos por las redes modernas, por ejemplo el algoritmo AES que es el estándar actualmente. Posteriormente se explicarán las bases de Scratch y cómo se adapta dicho lenguaje de programación a la creación de los cifrados propuestos. Por último, se explicará detalladamente el procedimiento a seguir para desarrollar los cinco programas (los cuatro cifrados y fuerza bruta) y así animar a nuestro público a usar Scratch para aprender a programar y asentar las bases que poseen todos los lenguajes de programación modernos (variables, lista de variables, bucles, envío de mensajes...).

Palabras clave: Scratch, criptografía, fuerza bruta, criptoanálisis, cifrado, algoritmo, lenguaje de programación, seguridad

Resum

El present treball es centra principalment en l'elaboració de quatre xifrats clàssics de la història de la criptografia (Escítala espartana, Tauler de Polibio, Xifrat de Cèsar i Disc d'Alberti) mitjançant l'ús de Scratch com a llenguatge de programació. A més, inclourem un programa senzill explicant el concepte d'atac de força bruta, el qual es seguix utilitzant actualment dins de les tècniques de criptoanàlisis. Aquestos xifrats s'inclouràn al Museu d'Informàtica de l'Escola Tècnica Superior d'Enginyeria Informàtica amb el propòsit d'ensenyar al públic adolescent de forma didàctica la importància de la seguretat amb exemples pràctics on ells són els que interactuen amb el programa. En la primera part del document es contarà cronològicament la història de la criptografia des del primer xifrat conegut històricament (Escítala espartana), fins als algoritmes que existixen hui en dia per a mantindre la seguretat dels missatges que enviem per les xarxes modernes, per exemple l'algoritme AES que és l'estàndard actualment. Més endavant s'explicaran les bases de Scratch i com s'adapta aquest llenguatge de programació a la creació dels xifrats propostos. Per últim, s'explicarà amb detall el procediment a seguir per a desenvolupar els cincs programes (quatre xifrats i força bruta) i d'aquesta manera animar el nostre públic a utilitzar Scratch per a aprendre a programar i assentar les bases que posseixen tots els llenguatges de programació moderns (variables, llista de variables, bucles, enviament de missatges...)

Paraules clau: Scratch, criptografia, força bruta, criptoanàlisis, xifrat, algoritme, llenguatge de programació, seguretat

Abstract

The current work is mainly focused on the elaboration of four classic ciphers in the history of criptography (Spartan scytale, Polybius square, Caesar cipher and Alberti's cipher wheel) using Scratch as the programming language. Furthermore we will include an easy program explaining the concept of Brute force which is still used at the present in the cryptanalysis techniques. These ciphers will be included in the Museo de Informática of Escola Tècnica Superior d'Ingenieria Informàtica in order to teach young public in a educational way how important the security is with practical examples where they will be the ones who interact with the program. First of all, the document will contain the history of the crytography chronologically from the very first known cipher (Spartan scytale), until the algorithms we know nowadays used to grant the safety of the messages we send on the Internet, for instance AES algorithm which is currently the standard. Afterwards, it will be explained how Scratch works and how it adapts to create the proposed ciphers. Finally, the development of each program will be described in detail aiming our target audience to use Scratch to start learning how to program and settle the fundaments that every modern programming language owns (variables, arrays, loops, message passing...).

Key words: Scratch, cryptography, brute force, cryptanalysis, cipher, algorithm, programming language, security

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En este capítulo expondremos las razones por las cuales hemos escogido el presente trabajo. Se explicará detalladamente los objetivos que se intentan cumplir y también describir al lector la estructura del proyecto que seguiremos a lo largo de todo el documento. Por otra parte, habrá un apartado donde se hablará de la bibliografía usada, centrándonos en cómo hemos aprovechado dichos recursos a la hora de crear nuestros programas con el lenguaje de programación Scratch.

1.1 Motivación

La principal motivación para realizar el proyecto es despertar el interés de los más jóvenes. Es por ello que el Museo de Informática de la Escola Tècnica Superior d'Ingenieria Informàtica de la Univesitat Politècnica de València ofrece un recorrido por la historia de la informática. El Museo de Informática realiza un trabajo incesante en cuanto a la difusión de las tecnologías, desde los artilugios clásicos como el Commodore Amiga, que fue un ordenador personal en la década de los 80, hasta los de hoy en día.

Dentro de las actividades que ofrece el museo, nosotros nos centraremos en el taller de Scratch, que intenta acercar a los visitantes al mundo de la programación con un lenguaje sencillo e interactivo como lo es Scratch. El presente trabajo servirá como ejemplo para conocer qué tipos de cifrados existen y qué técnicas han existido a lo largo de la historia.

Como punto final, el trabajo también pretende servir como motivación a estos usuarios a inicializarse en el mundo de la programación y no solamente describir el funcionamiento de los cifrados.

1.2 Objetivos

El trabajo realizado tiene como objetivo crear cuatro cifrados clásicos de la historia orientado a los jóvenes y un ejemplo de una técnica de criptoanálisis, concretamente el ataque mediante fuerza bruta. Los cifrados a crear son:

1. Escítala espartana
2. Tablero de Polibio
3. Cifrado de César
4. Disco de Alberti

Además de la creación de los cifrados, el trabajo tiene como objetivo dar a conocer la historia de los primeros cifrados existentes hasta los que se usan a día de hoy y sus características de modo que el lector pueda ver las diferencias y sacar sus propias conclusiones.

Cabe destacar que se puede encontrar el código de los programas realizados en la página del museo y de esta manera conocer cómo está programado internamente. Se trata de una manera interactiva para aprender.

1.3 Estructura de la memòria

El trabajo realizado se estructura en diez capítulos, a continuación describiremos brevemente su contenido:

- **Capítulo 1.** Se expone la motivación, objetivos y estructura de la memoria.
- **Capítulo 2.** Se mostrará una aproximación histórica de los primeros cifrados conocidos por el hombre hasta los cifrados que se usan en la actualidad.
- **Capítulo 3.** El entorno y lenguaje de programación Scratch. Conoceremos de manera detallada las herramientas que nos ofrece dicho entorno para la creación de programas sencillos.
- **Capítulos 4, 5, 6, 7, 8.** En cada capítulo se expondrá en detalle cada uno de los cifrados y el ataque de fuerza bruta. Se describirá el contexto histórico, el funcionamiento, las fortalezas y sus debilidades que hacen que se pueda romper el mensaje oculto (criptograma).
- **Capítulo 9.** Se describirá cómo está estructurada la página web y qué tecnologías hemos utilizado para su creación.
- **Capítulo 10.** En este último capítulo mostraremos las conclusiones finales del trabajo realizado, qué objetivos hemos cumplido y propondremos trabajos futuros.

1.4 Uso de la bibliografía

En esta sección de la memoria se hablará de la bibliografía utilizada para poder llevar a cabo la realización del trabajo. Nos centraremos en los recursos que hemos podido obtener de la web y así poder comenzar con el trabajo en el entorno de Scratch.

Para la redacción de la memoria se consultaron diversos trabajos anteriores de antiguos alumnos de la propia escuela centrándonos especialmente en dos de ellos, realizados por Samuel Villaescusa Vinader y Miguel Marqués Moros ([31], [21]).

Por otra parte, todo lo relacionado con Scratch se buscó dentro de la propia web [28]. Además, para ampliar la información sobre el entorno y el lenguaje de programación Scratch existe un foro dentro de la página donde cualquier usuario puede preguntar sus dudas [10].

Toda la información útil, lo podéis encontrar en las siguientes bibliografías [1, 2, 3, 4, 5, 6, 12, 13, 14, 15, 16, 17, 22, 23, 30]. En dichas bibliografías describen la historia de cada uno de los cifrados, los autores y el funcionamiento de estos.

1.5 Derechos de autor

Cabe destacar que la imagen del Disco de Alberti y Escítala espartana han sido creadas por Celia Soler Iranzo de la Universitat Jaume I, estudiante de Ingeniería de Diseño Industrial y Desarrollo de Productos. El resto de imágenes han sido obtenidas libres de derechos de autor en la web o han sido creadas mediante el editor de imágenes que nos ofrece Scratch.

Todos los sonidos que hemos utilizado pertenecen a la biblioteca de sonidos del entorno de Scratch.

CAPÍTULO 2

La criptografía clásica: una mirada retrospectiva

En este capítulo se pretende dar a conocer la historia de cuatro de los grandes cifrados clásicos en orden cronológico (Escítala espartana, Tablero de Polibio, Cifrado de César y Disco de Alberti), que marcaron un antes y un después en la historia de la criptografía. Así como su funcionamiento y la manera para romper el mensaje.

2.1 Escítala espartana (siglo V a. C.)

En la presente sección se presentará el contexto histórico en el que surgió el primer cifrado conocido históricamente. Además, se explicará el funcionamiento del cifrado así como el descifrado y finalizaremos con un ejemplo sencillo.

> The Spartans are the equal of any men when they fight as individuals; fighting together as a collective, they surpass all other men.
>
> *Damaratus a Xerxes*

2.1.1. Contexto histórico

El mundo de la criptografía comienza en la Antigua Grecia, donde el ejército formaba una parte fundamental del Estado. Este ejército no era ni más ni menos que el conocido ejército espartano, uno de los más importantes de toda la historia de Grecia.

En contrapartida, para llegar a ser el ejército más temible, en Esparta se practicaba la eugenesia. Por ello, cuando una mujer daba a luz a un niño, este era examinado por una comisión de ancianos que determinaba si estaba sano y bien formado. En el posible caso de que no lo estuviese, se le consideraba una carga para el propio Estado y se le abandonaba. Si el niño nacía sano se le confiaba a su familia para que lo criara hasta la edad de siete años, posteriormente pasaría a manos del Estado con el único fin de prepararlo para el ejército. Se dice que el ejército espartano era el más disciplinado, mejor entrenado y por supuesto, el más temible de la época. Además, un soldado espartano valía lo que varios hombres de cualquier estado.

Por otra parte, los espartanos eran astutos y lo aplicaban en el campo de batalla. Se inventaron un sistema de lo más simple con lo que se conocería como la primera he-

Figura 2.1: Ejército espartano con la formación hoplítica

rramienta de cifrado: la escítala espartana. En los tiempos que corría era fundamental encriptar la información para que el enemigo no la captara. Este sistema también lo utilizaban los éforos de la Antigua Grecia (véase la figura 2.2).

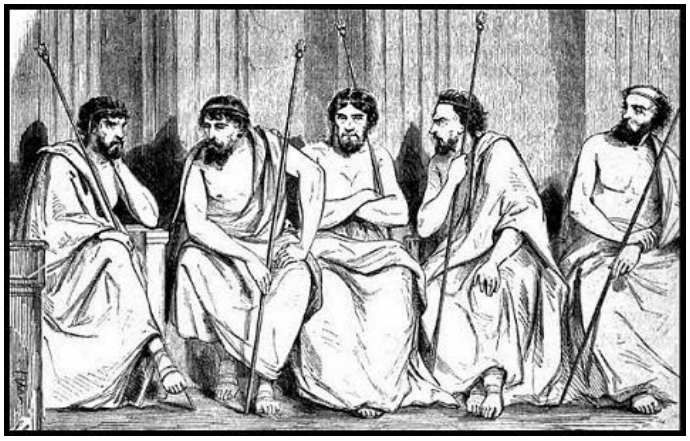

Figura 2.2: Éforos de la Antigua Grecia

Se trata del primer cifrado conocido por el hombre históricamente y la primera vez que se mencionó fue por un poeta griego llamado Archilochus, quien vivió en el siglo VII a. C. Este cifrado usaba el método de la transposición, que consistía básicamente en que siguiendo un esquema preestablecido se intercambiaban los caracteres del mensaje original. En este cifrado, este esquema es el diámetro de la vara. Al mismo tiempo, este cifrado pertenece a la familia de los cifrados de clave simétrica debido a que se usa la misma clave para cifrar y descifrar (diámetro de la escítala). Así pues, la escítala espartana (véase la figura 2.3) será el punto de partida en el mundo de la criptografía que conocemos hoy en día.

2.1.2. Cifrado

La escítala espartana consiste básicamente en enrollar una cinta alrededor de la propia escítala o vara. Una vez enrollada el emisor escribía horizontalmente sobre la vara y al desenrollar la cinta de la vara se creaba un mensaje oculto (véase la figura 2.4), más adelante veremos un claro ejemplo. La importancia del diámetro de la vara radica en que con una escítala de diferente tamaño de diámetro la reorganización de la cinta al enrollarse sobre la vara sería errónea y el mensaje seguiría oculto.

Para la época, este ingenioso sistema era realmente difícil de descifrar si no se sabía la medida del diámetro.

Figura 2.3: Escítala espartana

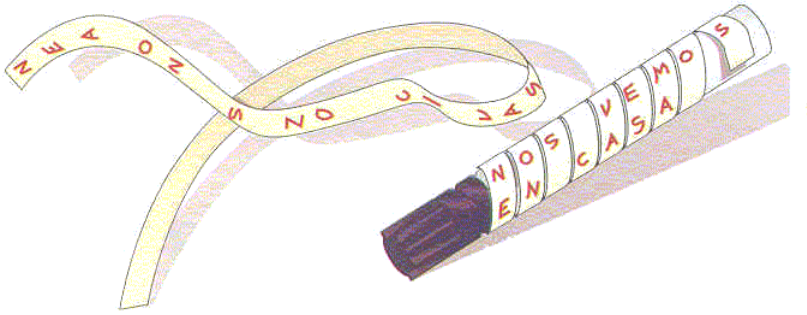

Figura 2.4: 'NOS VEMOS EN CASA' escrito sobre la cinta de la escítala espartana. A la izquierda se muestra el resultado de la cinta una vez desenrollada.

2.1.3. Descifrado

Para el desencriptado del mensaje, basta con conocer el diámetro de la escítala para reorganizar la cinta y obtener un texto legible. En caso contrario, el texto continuaría ilegible por parte del receptor.

2.1.4. Ejemplo

En este siguiente ejemplo tendremos una vara con una cinta enrollada dando como resultado ocho columnas donde escribiremos el siguiente mensaje: *kill king tomorrow midnight*.

El texto escrito en nuestra escítala que contiene ocho columnas se visualizaría de esta manera:

K	I	L	L	K	I	N	G
T	O	M	O	R	R	O	W
M	I	D	N	I	G	H	T

Tabla 2.1: Ejemplo de cómo quedaría el texto 'KILLKINGTOMORROWMORNING' en una matriz que representa la escítala.

Una vez desenrollamos la cinta, el criptograma obtenido es: *ktm ioi lmd lon kri irg noh gwt*.

2.2 Tablero de Polibio (siglo II a. C.)

En esta sección mostraremos la bibliografía y contexto histórico de la época en la que vivió Polibio. También describiremos detalladamente la aportación que realizó a la criptografía y su funcionamiento. Por otra parte, hablaremos de la técnica del criptoanálisis principal para descifrar cifrados monoalfabéticos, el análisis de frecuencias.

> If history is deprived of the Truth, we are left with nothing but an iddle, unprofitable tale.
>
> *Polybius*

2.2.1. Contexto histórico

Durante el siglo II a. C. nació en Megalópolis (Grecia) quien sería considerado uno de los historiadores más importantes debido a que fue el primero en escribir una historia universal.

El principal objetivo de Polibio era explicar cómo pudo imponerse la hegemonía romana en el Mediterráneo, describiendo para ello cómo se encadenan los sucesos políticos y militares. Toda esta información se recoge en su obra *Historiae* (véase la figura 2.6), que consta de un total de 40 volúmenes. Un dato curioso es que actualmente se conservan completos los cinco primeros volúmenes, del resto quedan solamente fragmentos.

Por otra parte, su trabajo a menudo se compara con el de Tucídides. Algunos de los críticos literarios de la época como Dioniso de Halicarnaso condenaron el estilo de Polibio por ser considerado áspero y vulgar, aunque él se mantenía firme con su idea de que el texto fuera lo más preciso posible más que por entretener a los lectores y así lo hizo.

Polibio vivió 20 años más después de que terminara su obra. Poco se sabe de la vida del historiador griego y se dice que se unió a Escipión Emiliano como asesor militar durante el asedio de Numancia en Hispania (134 - 132 a. C.). Posteriormente, escribiría sobre las experiencias vividas en la guerra aunque dicho trabajo se perdería para siempre. Finalmente, moriría a la edad de 82 años tras una caída a lomos de su caballo.

En cuanto a su aportación al mundo de la criptografía, alrededor del año 150 a. C. surge lo que se conocería como el primer cifrado que usa un algoritmo de sustitución y que recibe el nombre de Tablero de Polibio (ver tabla 2.2). Este algoritmo utiliza como base una matriz 5x5, es decir, de cinco filas y cinco columnas donde cada par fila-columna se corresponde a un carácter.

Cabe destacar que el alfabeto español contiene 27 caracteres por lo que se debe omitir dos en este caso. Para nuestro ejemplo hemos eliminado la 'Ñ' y la 'Z'. La organización de estos caracteres sobre la tabla da lugar a la clave del cifrado, a pesar de que en nuestro caso hemos ordenado dichos caracteres en orden alfabético para que sea más sencillo al lector seguir nuestros ejemplos.

Ahora bien, este cifrado pertenece a la familia de los cifrados simétricos al igual que la Escítala espartana (veáse la sección 2.1), que usaba los espartanos en la Antigua Grecia debido a que el emisor y el receptor deben compartir la misma clave, en este caso

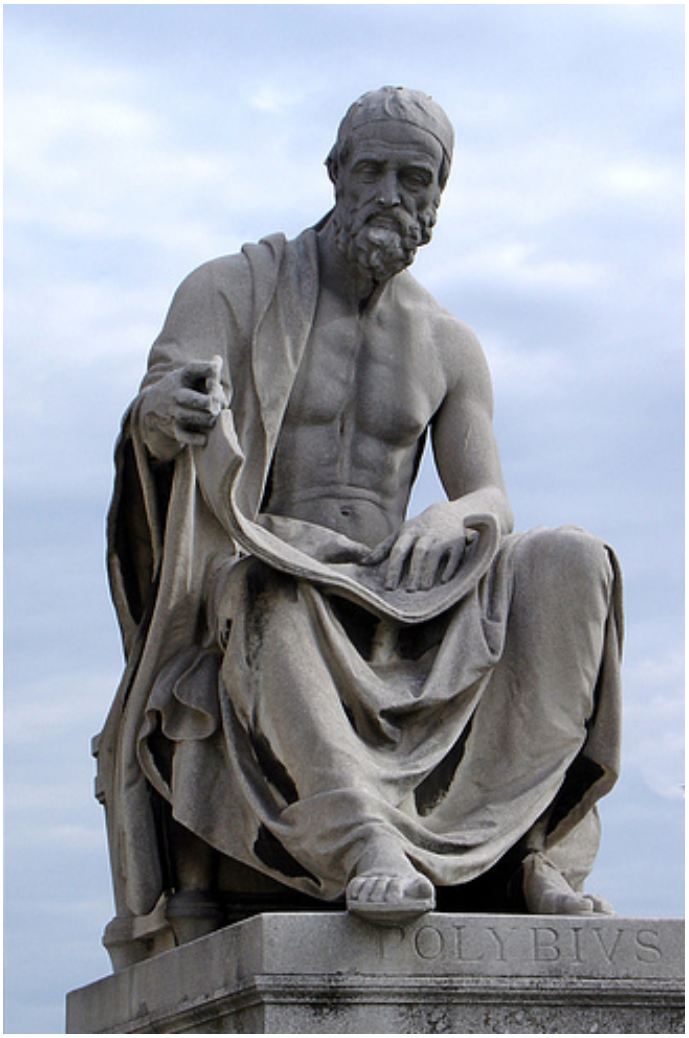

Figura 2.5: Estatua de Polibio en Viena, Austria.

la misma matriz, cuyos caracteres deben estar distribuidos de la misma forma. En caso contrario el receptor, al intentar descifrar el mensaje oculto, el texto continuaría siendo oculto.

2.2.2. Cifrado

Hemos visto que para la encriptación de un mensaje usando el método de Polibio se necesita una clave que será el tablero o matriz en cuestión. El siguiente paso será escribir el mensaje donde cada uno de los caracteres se sustituirá por la pareja correspondiente a la fila y columna de dicho carácter en la matriz, de este modo como entrada obtendremos una serie de caracteres y como salida nos generaría un conjunto de dígitos que van del uno al cinco (correspondientes a los índices del tablero). Si quisiéramos que nos generase una salida también de caracteres en vez de la tabla 2.2, podríamos usar la tabla 2.3.

Para el ejemplo de más adelante usaremos la tabla 2.2.

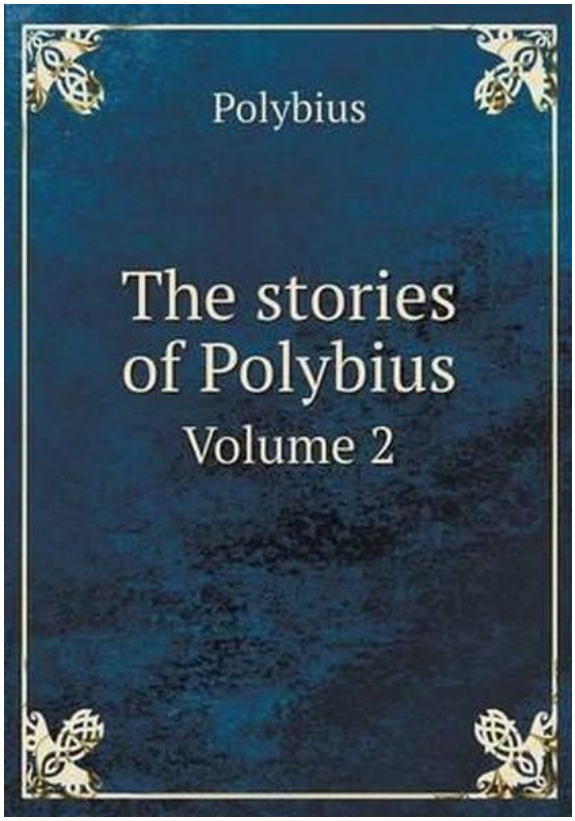

Figura 2.6: Volumen 2 de 'Las historias de Polibio' traducido al inglés.

2.2.3. Descifrado

En la parte del emisor, cuando se recibe un criptograma cifrado siguiendo este método bastará con sustituir cada pareja de dígitos por el correspondiente al carácter que se sitúa en el tablero.

La dificultad reside en que la clave nos es desconocida y para la época este cifrado era prácticamente indescifrable. Hoy en día existe una técnica de criptoanálisis que se denomina «análisis de frecuencias» que consiste en realizar un conteo de cada carácter del alfabeto y así establecer cuál es el que con más frecuencia aparece en el criptograma. La información recogida se contrasta con una tabla de frecuencias de cada idioma previamente elaborada (ver figura 2.7). Siguiendo la tabla en la lengua española el carácter que más aparece es la letra 'E' con una probabilidad del 14 % seguida de la 'A' con un 12,3 %.

Ahora bien, si en el texto encriptado realizamos dicha técnica del Análisis de frecuencias y el resultado que obtenemos es el de la tabla 2.9 vemos que el carácter 'H' es el que con más frecuencia aparece y de este modo sabemos que la 'H' del criptograma se sustituye por la 'E' del mensaje original.

Cabe mencionar que solo servirá para los cuatro caracteres más frecuentes debido a que existe un margen diferenciador bastante amplio. Una vez sustituidos los primeros caracteres, podemos realizar otra tabla que nos muestren los bigramas[1] y trigramas[2] con mayor índice de aparición. En el lenguaje español, los bigramas más frecuente son: *es*, *ue*, *en*, *de*, *qu*, *os*, *er*, *el*, *as*, *ra* y los trigramas: *que*, *est*, *ent*, *oqu*, *del*, *con*, *ien*, *ues*, *ade*, *aqu*. Esta técnica del criptoanálisis solo sirve para los cifrados de sustitución simple, es decir, que un carácter siempre se sustituirá por el mismo. En el caso de los cifrados polialfabéticos

[1] Secuencia de dos letras consecutivas o dos números
[2] Secuencia de tres letras consecutivas o tres números

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y

Tabla 2.2: Matriz que representa el tablero de Polibio. En él vemos que los caracteres están ordenados alfabéticamente donde excluimos los caracteres 'Ñ' y 'Z' ya que el alfabeto español contiene 27 letras mientras que solo hay 25 casillas. Las filas y columnas se representan mediante índices numéricos.

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

Tabla 2.3: Misma representación de los caracteres que la tabla 2.2 pero mostrando las filas y columnas mediantes caracteres desde la 'A' hasta la 'E'.

el análisis de frecuencias no servirá ya que un carácter se sustituye por varios caracteres durante el proceso de cifrado.

Las figuras 2.7, 2.8 y 2.9 se han extraído de [11].

2.2.4. Ejemplo

En el ejemplo vamos a escribir el texto *'killkingtomorrowmidnight'* y con la ayuda de la tabla 2.2 (nuestra clave) comenzaremos a cifrar el mensaje. Tal como dijimos anteriormente cada carácter del mensaje original se sustituye por el índice de la fila y la columna en el que esté. De este modo el criptograma generado es el siguiente:

'32243232314434224535333543433553332414342422 2345'

Para el descifrado bastaría realizar simplemente las operaciones inversas. Como es de esperar, la longitud del criptograma es el doble de largo que el mensaje original ya que un carácter se sustituye siempre por dos dígitos numéricos. Esto último podría ser una gran desventaja para textos muy extensos.

Frecuencia Alta		Frecuencia Media		Frecuencia Baja	
e	14.0	u	4.9	v	1.1
a	12.3	t	3.8	g	1.0
o	9.8	c	3.6	j	0.6
s	7.6	m	2.7	f	0.5
n	6.6	p	2.1	z	0.4
r	6.2	q	2.0	ñ	0.2
i	5.6	b	1.5	x	0.04
l	5.5	y	1.4	k	0.0004
d	5.3	h	1.2	w	0.0002

Figura 2.7: Tabla de frecuencias donde en ella podemos ver la frecuencia de aparición de cada una de las letras del alfabeto español, siendo el carácter 'e' el que más aparece con una probabilidad de 14 % y la 'w' con un 0.0002 %.

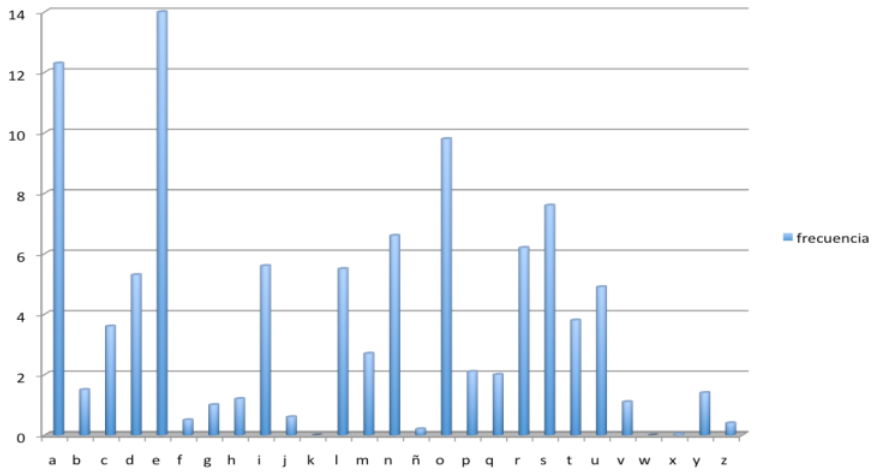

Figura 2.8: Datos de la figura 2.7 representados en diagrama de barras.

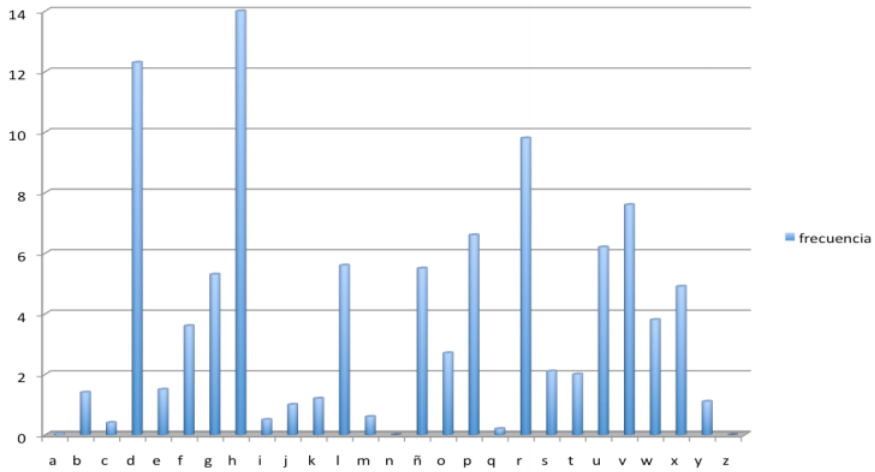

Figura 2.9: Frecuencia de aparición de los caracteres extraídos de un texto encriptado aleatorio.

2.3 Cifrado de César (siglo I a. C.)

En esta sección empezaremos con el contexto histórico del dictador romano, Julio César y de su conocido sistema de encriptación, el cifrado de César. Se hablará del funcionamiento de dicho cifrado y de cómo se puede romper la seguridad de los mensajes encriptados mediante este sistema. Finalizaremos con un ejemplo para asentar los conocimientos aprendidos durante este capítulo.

I came, I saw, I conquered.

Julius Caesar

2.3.1. Contexto histórico

Cayo Julio César (véase la figura 2.10) (Roma, 100 - 44 a. C.) fue un líder militar y político que gobernó la República de Roma años antes de la Era Cristiana. Procede de una de las familias más antiguas de Roma, los Julios, por lo que la educación que recibió fue esmerada por maestros griegos.

Desde muy joven se interesó por la política, con una ideología populista, donde le unía su relación con Mario (político y militar romano), que recibía el nombre de tercer fundador de Roma debido a sus éxitos militares. Con el paso del tiempo, César fue aumentando su prestigio en los distintos cargos que ocupó, desde cuestor (recaudador de impuestos) a propretor (magistrado romano) de la Hispania Ulterior a la edad de 39 años.

De regreso a Roma, consiguió una hazaña al reconciliar a los dos líderes rivales, Craso (ver figura 2.11) y Pompeyo (ver figura 2.12), mediante acuerdos privados para repartirse el poder formando un triunvirato.

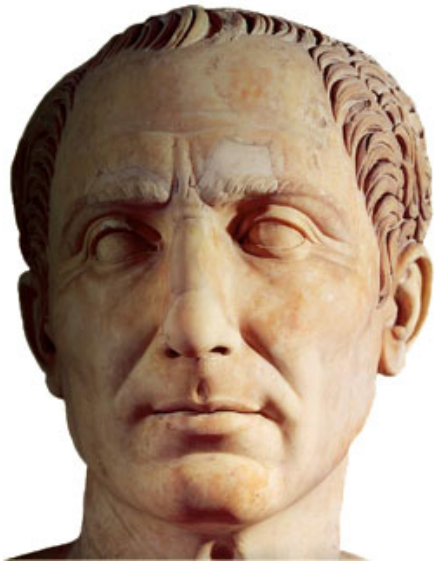

Figura 2.10: Busto del general Cayo Julio César.

Más adelante, tras la muerte de Craso en una expedición y también la muerte de la hija de César, que era hasta entonces la esposa de Pompeyo, afectaron seriamente la relación que mantenían ambos gobernantes.

Mientras tanto, el líder militar Julio César se lanzó a la conquista de nuevos territorios para Roma resultando airoso en todas sus expediciones. El prestigio y el poder que al-

canzó el general preocupó a Pompeyo por lo que más tarde se enfrentarían el uno contra el otro. Pompeyo moriría tras ser perseguido por César en Egipto. Posteriormente, el dictador Cayo Julio César seguiría acumulando victorias hasta el día de su muerte a la edad de 56 años. Fue asesinado por Casio y Bruto, que le impidieron completar sus reformas.

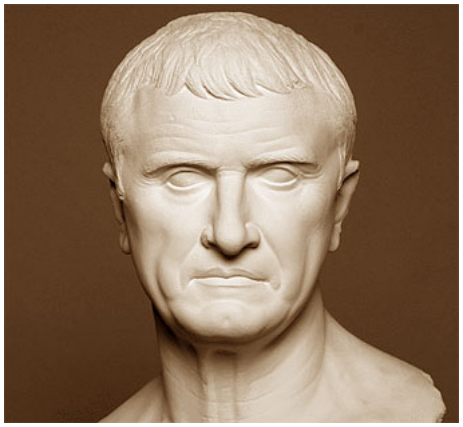

Figura 2.11: Busto de Marco Licinio Craso.

Figura 2.12: Busto de Cneo Pompeyo.

Finalmente, en el contexto de la criptografía, el dictador inventó un sistema muy sencillo para comunicarse con sus generales de forma segura, conocido como cifrado por desplazamiento o desplazamiento de César. Este mecanismo era considerado suficientemente seguro debido a que la mayoría de la población no sabía leer ni escribir.

2.3.2. Cifrado

Este sistema consistía en desplazar cada letra del mensaje original por otra que se encuentre un número de posiciones más adelante del alfabeto; dicho número se considera la clave del cifrado. A pesar de que este número puede ser cualquiera, el cifrado de César originalmente usa un desplazamiento de tres posiciones: una 'A' en el texto original se sustituiría por una 'D' en el mensaje oculto.

En la sección 2.3.4 veremos un ejemplo de cómo funciona.

2.3.3. Descifrado

Para el caso del descifrado, bastará con realizar la operación opuesta. Esto quiere decir que si tenemos una clave $k = 3$, la letra 'D' se descifraría como una 'A' debido a que volveríamos tres posiciones hacia atrás. Cabe destacar que como estamos ante un cifrado monoalfabético, podemos usar la técnica de Análisis de frecuencias tal como hemos visto anteriormente en la sección 2.2.3.

2.3.4. Ejemplo

En esta parte, mostraremos un pequeño ejemplo sobre el uso del cifrado de César. Para ello, tomaremos el siguiente texto: *ATACARALAMANECER* con un desplazamiento o clave $k = 3$.

Tras aplicar el cifrado sustituyendo cada letra del mensaje original tres posiciones a la derecha (explicado en la sección 2.3.2), obtenemos como resultado *DWDFDUDÑDODPHFHU*, un texto ilegible.

2.4 Disco de Alberti (1467)

En esta última sección se describirá la vida y contexto de quien podríamos denominar como una de las figuras más emblemáticas del Renacimiento del siglo XV. Se describirán sus numerosas aportaciones en los distintos campos como la arquitectura, el arte y sobre todo en la criptografía, con el primer cifrado polialfabético conocido históricamente.

> No art, however minor, demands less than total dedication if you want to excel in it.
>
> *León Battista Alberti*

2.4.1. Contexto histórico

León Battista Alberti (véase la figura 2.13), más conocido simplemente como Alberti, nació en 1404 en Génova. Alberti fue un arquitecto y escritor italiano y, junto a Leonardo Da Vinci fue una de las figuras más representativas del Renacimiento, ya que reunía todos los conocimientos de la época.

Figura 2.13: León Battista Alberti

Hijo natural de Lorenzo Alberti y de Bianca Fietschi. En cuanto a su educación recibió la mejor posible por un noble italiano. Desde 1414 a 1418 estudió clásicos [3] en el prestigioso colegio Gasparino Barzizza en Padua. Más adelante completaría sus estudios en la universidad de Bolonia donde estudió derecho.

[3] Estudio de las lenguas, cultura, historia y el pensamiento de la civilación de la Antigua Grecia y Roma.

Como anécdota de su juventud, se decía que Alberti era capaz de saltar por encima de una persona adulta. Aprendió música sin ningún maestro y sus composiciones musicales llegaron a ser admiradas por jurados profesionales.

Debido a su carácter polifacético, Alberti realizó varias contribuciones en distintos campos. Destacaremos algunas de las más importantes:

1. Alberti fue el creador de un teoría que se llama *Historia* explicado en su tratado De pictura (1435) [8].

2. Alberti fue el autor de un trabajo influyente en la arquitectura, *De Re Aedificatoria* (véase la figura 2.14). Durante el siglo XVIII, fue traducido al italiano, francés, español y al inglés.

3. En la obra de Francesco Colonna, *Hypnerotomachia Poliphili* (1499) (véase figura 2.15) se le atribuye a Alberti varias ilustraciones de xilografía[4].

4. En el contexto de la criptografía, Alberti también fue una figura destacada e inventó el primer cifrado polialfabético de la historia el cual se conoce hoy en día como el cifrado de Alberti.

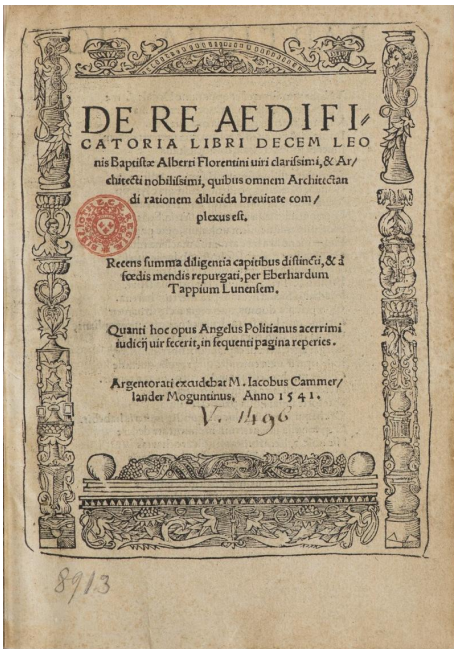
DE RE AEDIFI/
CATORIA LIBRI DECEM LEO
nis Baptistæ Alberti Florentini uiri clarissimi, & Ar/
chitecti nobilissimi, quibus omnem Architectan
di rationem dilucida breuitate com/
plexus est,
Recens summa diligentia capitibus distincti, & à
fœdis mendis repurgati, per Eberhardum
Tappium Lunensem,
Quanti hoc opus Angelus Politianus acerrimi
iudicij uir fecerit, in sequenti pagina reperies.
Argentorati excudebat M. Iacobus Cammer/
lander Moguntinus, Anno 1541.
V. 1496
8913

Figura 2.14: De Re Aedificatoria

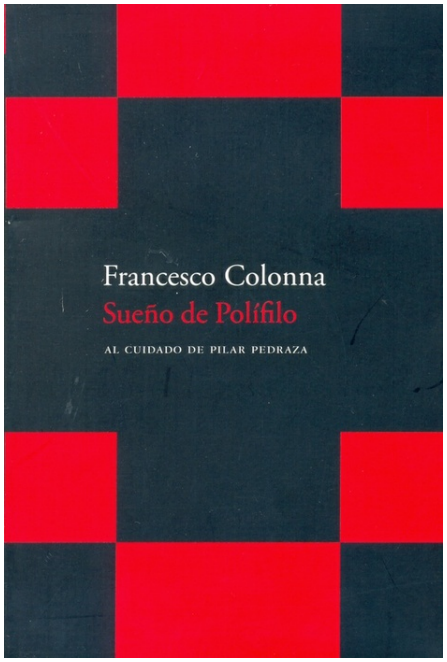

Figura 2.15: Hypnerotomachia Poliphili traducido al español

En las siguientes líneas nos centraremos en su aportación a la criptografía. Para el cifrado de Alberti se usa un disco que se compone de dos anillos concéntricos formado por 24 celdas cada uno. Este invento fue el primer avance significativo desde el cifrado de César (mencionado anteriormente en la sección 2.3). David Khan, un historiador de renombre y autor de la obra *Codebreaker* [19], le nombró como *"Father of Western Cryptography"*.

[4]Técnica de grabar imágenes en una plancha de madera vaciando las partes que en la reproducción o impresión deben quedar en blanco.

2.4.2. Cifrado

Para comenzar a explicar el funcionamiento del disco de Alberti, es fundamental introducir tres parámetros esenciales: el periodo, el incremento en cada periodo y por último la configuración inicial.

El periodo dentro de este contexto tiene una relación directa con la frecuencia de aparición de los caracteres del mensaje que queramos enviar al receptor. El incremento del disco de Alberti es el número de veces que se rota el disco (en sentido horario) por cada periodo. La configuración inicial describe en qué posiciones están ambos anillos, interno y externo, a la hora de cifrar el mensaje.

Ahora bien, una vez presentados todos los parámetros, la combinación de estos forman la clave del cifrado. Alberti, para complicarlo más todavía, añadió a su disco los dígitos del uno hasta el cuatro con el objetivo de formar códigos cuyos significados se guardan en un libro de códigos. Por ejemplo el 324 podría significar 'Las naves están listas para zarpar'. Alberti recomendó cifrar el código a su vez con el fin de que si se capturara el libro de códigos, el mensaje permaneciera seguro.

El funcionamiento del cifrado consistía en que una vez los anillos se situasen en la configuración inicial del disco, el carácter del anillo externo se sustituía por el inmediatamente inferior. La complejidad del cifrado viene dada en que en cada periodo establecido, el disco se rota un incremento (si el incremento es de tres, el disco se rota tres veces) y de esta manera el carácter anterior ahora se sustituye por otra consiguiendo un sistema polialfabético (véase la figura 2.16), se muestra que el carácter 'A' se sustituye en este caso por la 'M'.

Figura 2.16: Disco de Alberti, en la figura podemos ver que el carácter 'A' del anillo superior se sustituye por la 'm' que está en el anillo inferior.

Un dato importante es que como el anillo externo posee 24 celdas y cuatro de ellas son dígitos, algunos caracteres del alfabeto se representará como se indica en la tabla 2.4:

Debido a esto, se realizará una fase de preprocesado del mensaje sustituyendo los caracteres que no estén en el disco por el correspondiente en la tabla 2.4. En la sección 2.4.4 se mostrará un ejemplo del primer cifrado polialfabético.

H	FF
J	II
K	QQ
U	VV
W	XX
Y	ZZ

Tabla 2.4: Tabla de equivalencias donde podemos ver que la 'H' se sustituye por 'FF', debido a que en el disco de Alberti solo dispone de 24 celdas en cada anillo y el alfabeto español consta de 27 caracteres.

2.4.3. Descifrado

En cuanto al descifrado, el primer punto a comentar es que la técnica del análisis de frecuencias descrito en el punto 2.2.3 no sirve en este caso debido a que es polialfabético y un carácter en concreto se puede sustituir por cualquiera a lo largo del proceso de encriptación.

Durante la época, Alberti afirmó que su método era indescifrable y lo denominó con sus propias palabras como "digno de reyes".

En el caso de si se conociera la clave del cifrado, el proceso de descifrado basta con realizar las mismas operaciones que en el cifrado pero fijándonos en el anillo interno y sustituyendo el carácter por el inmediatamente superior.

2.4.4. Ejemplo

Como ejemplo a mostrar, tendremos los siguientes parámetros:

1. $periodo = 4$
2. $incremento = 2$
3. La configuración inicial será la de la figura 2.16
4. Mensaje original: *MATARALREY*

Como en el anillo externo no existe el carácter 'Y', dicha letra se sustituirá por 'ZZ' tal como se ha visto en la sección 2.4 con el que tras la fase de preprocesado el texto quedará como *'MATARALREZZ'*.

Una vez tenemos el mensaje preprocesado empezaremos a sustituir los caracteres por el correspondiente en el anillo interno. En el caso del primer carácter 'M', este se sustituirá por la 'E'. Seguiremos de la misma forma hasta llegar al cuarto carácter (debido al periodo) que rotaremos en este caso dos veces el disco (incremento).

El siguiente paso será volver a cifrar hasta que se cumpla de nuevo el periodo. El resultado se muestra en la tabla 2.5. Como se puede observar, el carácter se ha sustituido por una 'M' y una 'S' cuando en un cifrado monoalfabético siempre se sustituye por el mismo valor.

M	A	T	A	R	A	L	R	E	Z	Z
E	M	T	M	L	S	B	L	M	R	R

Tabla 2.5: Resultado del cifrado de Alberti al cifrar 'MATARALREY' con los parámetros mencionados en 2.4.4.

2.5 La criptografía moderna

En esta presente sección introduciremos al lector sobre los cifrados que existen en la actualidad y también su funcionamiento sin entrar en demasiados detalles. Para ello, hablaremos sobre al algoritmo DES y posteriormente sobre AES.

2.5.1. Algoritmo DES

En el año 1973 por parte de la NIST (*National Institute of Standards and Technology*) se realizó una convocatoria con el objetivo de crear un nuevo estándar para el cifrado de los mensajes. Poco tiempo más tarde se confirmaba que el ganador es el algoritmo DES creado por Horst Feistel y por tanto, este se establece como nuevo estándar en el ámbito de la criptografía.

El algoritmo consiste en que dada una clave k de 64 bits y un mensaje, a lo largo del cifrado existe una serie de permutaciones y un conjunto de tablas en las cuales, la más importante era la tabla S, cuya finalidad era reorganizar el mensaje inicial siguiendo un patrón establecido por dichas tablas. En el año 1998 DES deja de ser el estándar y se realiza otro convocatoria.

Para saber más acerca de este algoritmo se puede acceder a las diversas páginas que existen en Internet como el siguiente enlace: `http://page.math.tu-berlin.de/~kant/teaching/hess/krypto-ws2006/des.htm`.

Una de las características de este algoritmo es que es simétrico y por ello el emisor una vez recibe el criptograma, solo tendrá que usar la misma clave con la que se cifró para obtener el mensaje original. El algoritmo DES posee variaciones, cada uno con sus ventajas y desventajas. A continuación explicaremos una de las variaciones más conocidas:

- **Triple DES.** Esta variación consiste en realizar el cifrado de DES como su nombre indica tres veces secuencialmente con el fin de garantizar una mayor seguridad. Las tres claves que se van a usar para el cifrado del mensaje deben ser distintas si se quiere mantener una seguridad y el espacio de claves pasa de 2^{56} a 2^{168}. Este algoritmo se puede ver en la figura 2.17.

2.5.2. Algoritmo AES

Tras el paso del algoritmo de DES mencionado en el apartado anterior surge como nueva alternativa el algoritmo AES que proviene del inglés, *Advanced Encryption Standard*, creado por John Daemen y Vincent Rijmen. También conocido como sistema Rijndael, este algoritmo es simétrico y es más seguro que el triple DES además de ser más rápido.

Una de las razones por la que se sustituyó DES es debido al tamaño de la clave que, con los avances tecnológicos, se vuelve insegura, por lo que en AES esta clave se duplica a 128 bits.

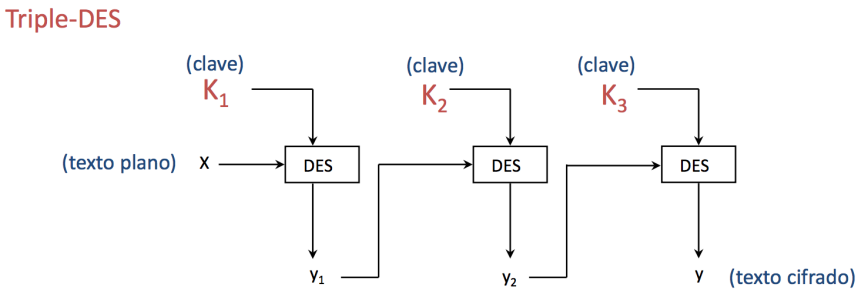

Figura 2.17: Esquema de funcionamiento del algoritmo triple DES. Partimos de un texto plano y tres claves k_1, k_2 y k_3 por lo que cuando cifremos el mensaje con la primera clave, luego realizamos la misma acción pero con la segunda clave y así sucesivamente.

Por otra parte, durante el periodo de DES había una constante paranoia por parte de las figuras importantes de la criptografía sobre el hecho de que los diseñadores de este algoritmo hayan introducido una puerta trasera o *backdoor*[5] con el fin de romper los mensajes encriptados siguiendo DES. Esto se debe a que durante la convocatoria que se realizó en 1973, el proceso de elección no fue transparente al público y existe una cierta desconfianza sobre la total seguridad de DES a pesar de que nunca se encontró ningún fallo dentro del algoritmo.

En el siguiente documento se encuentra un ejemplo explicado del algoritmo AES: https://kavaliro.com/wp-content/uploads/2014/03/AES.pdf.

[5]Secuencia especial dentro del programa con el fin de evitar los sistemas de seguridad y de esta manera acceder a él.

CAPÍTULO 3
El entorno de programación Scratch

En el presente capítulo se describirá el entorno de programación Scratch (véase la figura 3.1) además del porqué de dicha elección. Se hablará también de las herramientas que nos ofrece a la hora de crear programas.

3.1 ¿Por qué Scratch?

Las razones principales por las que hemos escogido este entorno de programación y no otros se describen a continuación:

1. **Entorno de programación**. Scratch fue concebido con el objetivo de que usuarios noveles sin apenas conocimientos de programación pudieran crear sus primeros programas desde cero o a partir del código de otros, ya que Scratch permite la compartición de proyectos de forma fácil y rápida (de ahí viene su lema presente en la figura 3.1). El hecho de que se pueda reutilizar el código debido al carácter libre ahorra una cantidad inmensa de tiempo y trabajo.

2. **Facilidad de uso**. A pesar de que Scratch no posee todas las características de un entorno de programación de alto nivel (corrección de sintaxis, importación de librerías, control de versiones...), Scratch es el lenguaje idóneo para los más jóvenes ya que se trata de un lenguaje sencillo cuya forma de programar se basa en los bloques (similar a los LEGO, véase figura 3.2) que se conectan entre ellos a la hora de añadir funcionalidad en el programa que estemos creando. La curva de aprendizaje es mínima.

3. **Compartición de código**. Todo y cada uno de los programas creados en Scratch se pueden compartir con el resto de usuarios, es decir, cualquier persona tiene la decisión de compartir o no sus proyecto Scratch. De esta manera el usuario puede acceder al código interno y si lo desea, importar el proyecto y realizar cambios para dar lugar a una nueva versión. Debido a esta funcionalidad de Scratch enriquecemos tanto al creador como al usuario que deseen empezar en el mundo de la programación.

4. **Curva de aprendizaje**. Como mencionamos anteriormente, el tiempo que se requiere para aprender Scratch es casi nulo. Una importante parte de esta razón es la inmensa cantidad de bibliografía que existe hoy en día en la web de Scratch y cursos externos que podemos encontrar en https://www.udemy.com. Actualmente Scratch se encuentra en la version 2.0.

5. **Editor gráfico online**. Cuenta con un editor donde el usuario puede crear imágenes para incorporarlo en su proyecto. En secciones posteriores se describirá con más detalle esta funcionalidad.

6. **Software libre**. Scratch es totalmente gratuito y está al alcance de cualquier usuario con ganas de comenzar en el entorno de programación. Por otra parte, Scratch está en más de 50 idiomas.

Figura 3.1: Logo de Scratch con su lema: «Imagina, programa, comparte».

Figura 3.2: Logo de Scratch construido mediante piezas de LEGO.

3.2 Proyecto Scratch

Scratch es una plataforma multimedia de programación con versión de escritorio y web. El público al cual va dirigido son niños y profesores de escuelas a pesar de que no existe ninguna restricción de uso con el fin de que los usuarios aprendan de manera sencilla crear programas que les ayuden a inicializarse en el mundillo de la programación.

Además Scratch ayuda a los usuarios a pensar de manera creativa, razonar y trabajar colaborativamente gracias a poder compartir proyectos. Todo el contenido de la plataforma lo puedes encontrar en su propia web (véase figura 3.3). [1]

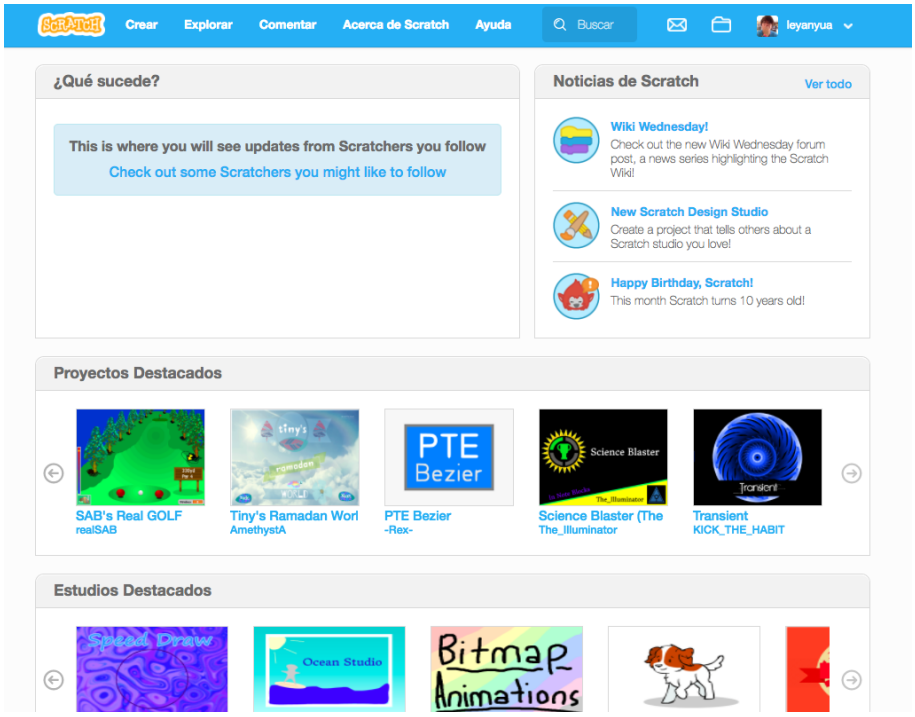

Figura 3.3: Pantalla inicial de la web de Scratch. En ella podemos ver en la parte superior derecha la opción de registrarse y en el centro de la figura, acceder a los diferentes proyectos realizados por otros usuarios.

El proyecto nació en 2003 de la iniciativa del grupo de investigadores *Lifelong Kindergarten* del MIT (*Massachusetts Institute of Technology*) liderado por Mitchell Resnick (véase figura 3.4). La idea de Scratch surgió por parte de los integrantes del grupo de investigación y se basó en el lenguaje de programación Logo[2] creado por Wally Feurzeig y Seymour Papert. Este lenguaje de programación era de alto nivel aunque de fácil aprendizaje por lo que Scratch adoptó esta característica.

Por otra parte, el término de Scratch viene dado por la técnica usada por los *disk jockeys* denominada *scratching*, que consiste en realizar modificaciones en las melodías moviendo el vinilo hacia delante y detrás. En el entorno de Scratch es algo similar debido a que mezcla animaciones, imágenes y sonidos. En palabras del líder de *Lifelong Kindergarten*:

> *We take the name "Scratch", from the way that hip-hop disk jockeys scratch with music. They take pieces of music and then combine them together in unexpected and creative ways.*

Otra connotación diferente que podemos darle al nombre es que el término *from scratch* significa desde cero en inglés. Lugar de donde proviene la mayoría de los usuarios con conocimientos nulos de programación.

Scratch es un proyecto de desarrollo cerrado y de código abierto. El hecho de que el desarrollo sea cerrado quiere decir que el grupo de investigación no intenta persuadir

[1] https://www.scratch.mit.edu/

[2] Guía realizada por Marisa Carro Rubiera. Dpto. de Tecnologías. I.E.S. "Elisa y Luis Villamil" (Vegadeo). CURSO 2008-2009. http://www.edu.xunta.gal/centros/iesricardomella/system/files/logo1_0.pdf

Figura 3.4: El creador de Scratch, Mitchell Resnick en un *Scratch Day*

a los usuarios a contribuir en la elaboración del proyecto sino que se ha realizado íntegramente por ellos. Que un proyecto sea de código abierto significa que el grupo de investigación libera la totalidad del proyecto para que la comunidad pueda modificarlo y añadir extensiones.

En marzo del año 2007 la página de Scratch fue completamente rediseñada coincidiendo con la salida de su segunda versión. Añadieron como novedad el componente social. A partir de aquel momento, todos los usuarios de Scratch, también llamado *scratchers*, podían compartir sus proyectos con la comunidad y visualizarlos. En la Figura 3.5 se muestra el espacio donde los *scratchers* tienen la posibilidad de comentar los proyectos del resto de usuarios.

3.3 Objetivo de Scratch

El grupo de investigación del MIT desarrolló Scratch con una única finalidad en mente, introducir a los más pequeños en el arte de la programación y de este modo enseñarles a programar de forma didáctica y divertida. Por ello, el componentes social juega un papel fundamental ya que ofrece la posibilidad a los usuarios a visualizar el código interno de programas ajenos.

Un detalle a destacar es que Scratch permite también el uso de comentarios en el código para enriquecer la información por parte del creador del programa.

3.4 Scratch y el pensamiento computacional

El pensamiento computación, proveniente del término inglés *Computational Thinking*, es un concepto definido por Jeannette M. Wing. Jeannette indica: «*El pensamiento computacional implica resolver problemas, diseñar sistemas y comprender el comportamiento humano, haciendo uso de los conceptos fundamentales de la informática*», es decir, se trata del proceso que implica la formulación de un problema y expresar la solución en secuencias en-

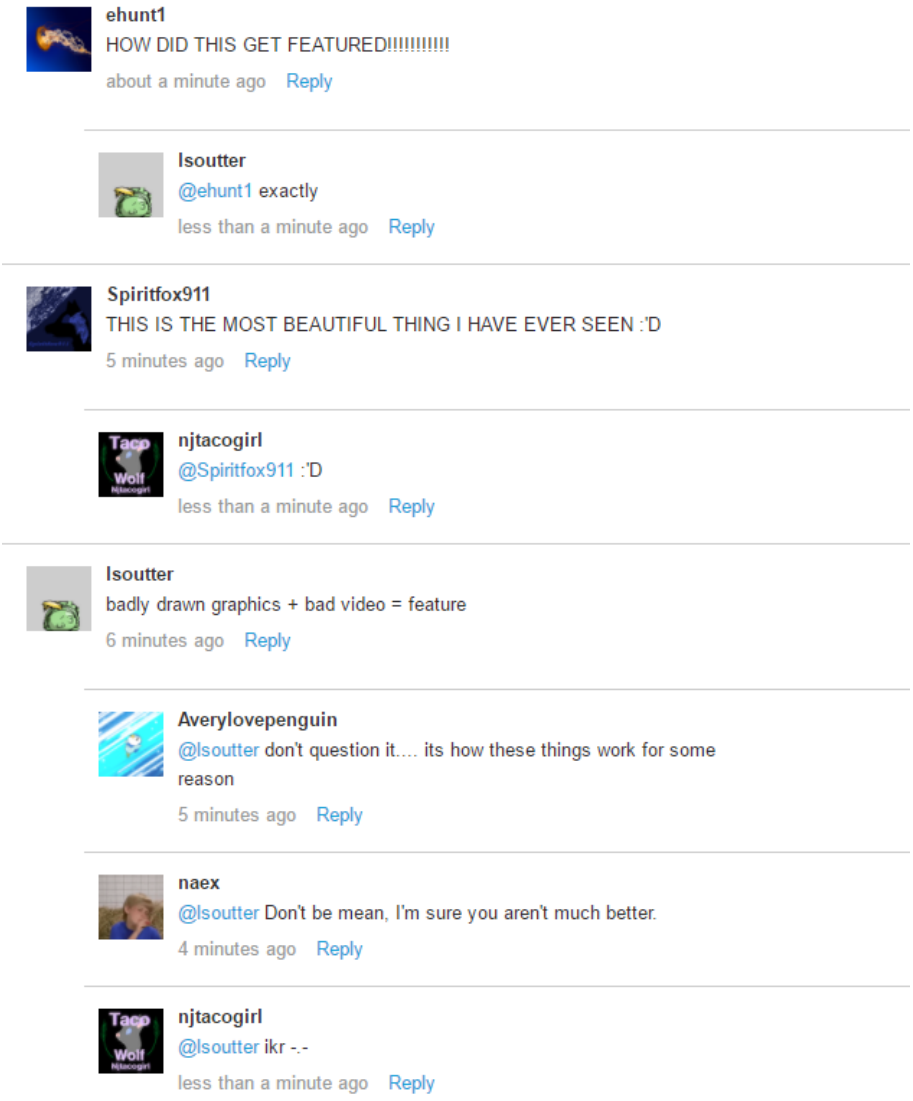

Figura 3.5: Sección de comentarios de un proyecto Scratch. Cada programa realizado en Scratch y compartido con la comunidad tiene esta sección donde todos los *scratchers* tienen la opción de valorar tu proyecto.

tendibles por una computadora. En [18] se puede consultar más información acerca del pensamiento computacional.

Ahora bien, dentro del entorno de Scratch se intenta abarcar este pensamiento estableciendo conceptos como: variables, operadores, métodos, eventos, variables condicionales, iteración y la secuencia de instrucciones. Por otra parte, Scratch intenta adaptar ciertas prácticas del pensamiento computacional:

1. Ser incremental en la búsqueda de soluciones: divide y vencerás.

2. Probar y depurar: nada sale a la primera, se cometen y se corrigen los errores.

3. Reusar código y modificarlo: no partimos de cero.

4. Abstraer, modelar y modularizar: creamos una estructura ordenada para gestionarla con mayor facilidad.

Estas prácticas son comunes en los lenguajes de programación de alto nivel por lo que una vez las dominamos, los conceptos aprendidos los podremos trasladar a lenguajes más complejos debido a que estas prácticas se enfocan principalmente en la manera que tiene una persona de pensar con el fin de llegar a una solución y no en la forma de programar de un lenguaje en concreto (sintaxis[3]).

3.5 Panel de objetos

Una de las características que ofrece Scratch a la hora de crear los programas es el panel de objetos. Este panel permite al usuario tener localizado de manera sencilla todos los objetos que vayamos creando a lo largo del proyecto y acceder al fragmento de código asociado a los objetos respectivamente que veremos más adelante en la sección 3.6. Esta característica ayuda a los usuarios noveles que no poseen conocimientos sobre programación a organizar todo el conjunto de objetos del escenario con la imagen asociada al objeto (véase la figura 3.6).

Figura 3.6: Panel de objetos en el cual se encuentran los objetos del proyecto. Esta captura de imagen ha sido extraída del cifrado de Alberti.

En la esquina superior de la imagen, vemos cuatro iconos que nos permite seleccionar una imagen de la biblioteca de Scratch, crear desde cero una imagen, subir desde nuestro ordenador una imagen o realizar una imagen desde la *webcam* del ordenador. Scratch ofrece un editor de imágenes (véase la figura 3.7) para que el usuario pueda dibujar o importar una imagen desde su ordenador personal. En el caso de que queramos importar una imagen a nuestro programa cabe mencionar dos conceptos muy importantes ligados al mundo de las imágenes: mapa de bits (*bitmap*) e imagen vectorizada (*Scalar Vector Graphics*). A continuación describiremos los conceptos y la ventajas de usar cada uno de ellos[4]:

[3] La sintaxis de un lenguaje de programación es la estructura en que se organizan los distintos elementos sintácticos, como espacios, identificadores, operadores, etc. Es decir, el orden que tienen unos con respecto a otros.

[4] Las ventajas y desventajas se han extraído del siguiente enlace: http://disenograficotco.blogspot.com.es/p/ventajas-y-desventajas_23.html

1. Mapa de bits: son imágenes son formadas a partir de puntos, llamados píxeles[5], que contiene información del color que representa. El conjunto de píxeles da lugar a la imagen en sí y define la resolución de la propia imagen. Muchos habrán oído hablar de una imagen cuya resolución es de 1920x1080 y esto quiere decir nada más y nada menos que dicha imagen se compone de 1920 píxeles de ancho por 1080 píxeles de alto.

   Las ventajas que tienen estas imágenes frente a las vectorizadas es que ocupan un menor tamaño y se requiere un menor tiempo a la hora de transferir dichas imágenes por la web.

2. Imagen vectorizada: las imágenes vectorizadas son representaciones de figuras geométricas tales como círculos, cuadrados o segmentos. Además, están representadas por fórmulas matemáticas (por ejemplo, un círculo está definido por un punto y un radio) donde un procesador «traducirá» estas fórmulas y las convertirá en figuras que el ser humano pueda visualizar y entender.

   La ventaja que posee este tipo de imágenes frente a los mapa de bits es que como no tienen una dimensión absoluta o asociada, al aumentar el tamaño de las imágenes no perderán calidad de imagen, cosa que los bitmaps sí. Se guardan en ficheros con extensión .svg.

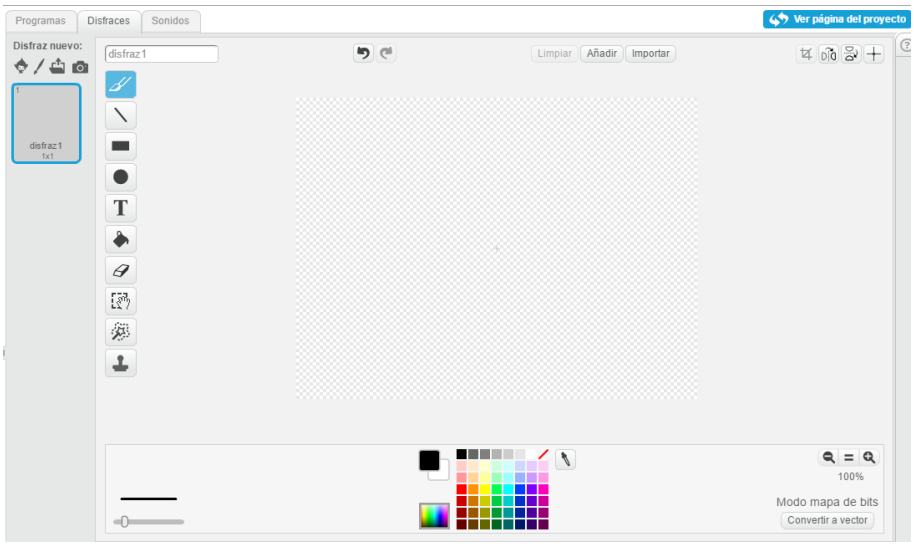

Figura 3.7: Editor de imágenes en Scratch. Podemos ver en la parte izquierda el conjunto de herramientas que ofrece Scratch para la creación de nuestras propias imágenes. En la esquina inferior derecha tenemos la opción de convertir la imagen a vector.

Así pues, una vez presentadas las diferencias de las imágenes que se pueden observar en la figura 3.8, el usuario será quien decida qué tipos de imágenes elegir para su proyecto, en nuestro caso para todos los proyectos que hemos realizado hemos hecho uso de imágenes vectorizadas. Para una información más detallada, consulte en la propia web de Scratch [27].

Por otra parte, en el panel de objetos que se puede ver en la figura 3.9, se permite cambiar el nombre al objeto en cuestión, ver en qué posición se encuentra, variar la rotación y esconder la imagen.

Finalmente, en el panel de objetos se encuentra un objeto que se define por defecto en Scratch llamado «Escenario», el cual maneja lo que ocurre en el programa y es quien se

[5]Unidad básica de una imagen digital asociado con un color formado por colores primarios.

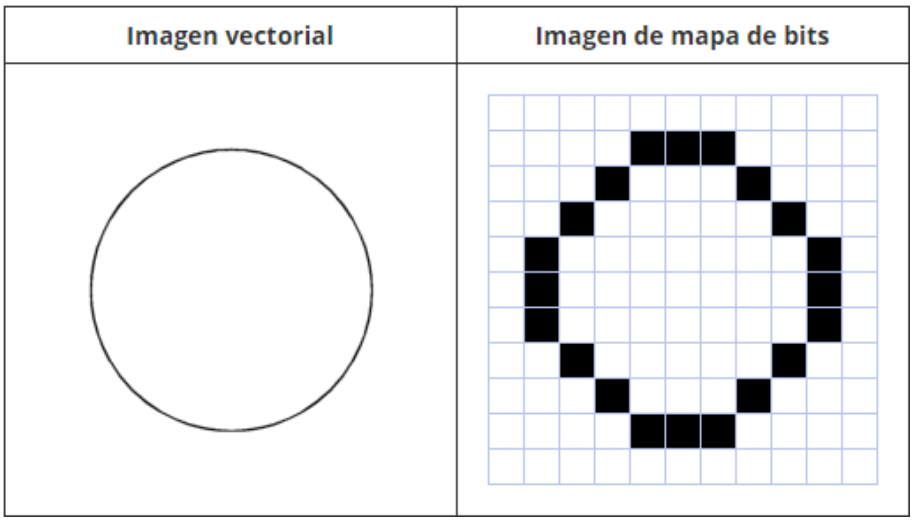

Figura 3.8: Diferencias entre los dos tipos de imágenes: a la izquierda la imagen vectorizada y a su derecha una imagen *pixelada*.

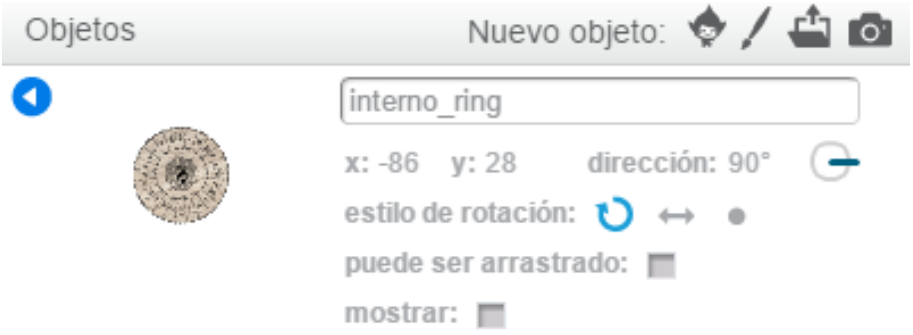

Figura 3.9: Información que se muestra al pulsar sobre un objeto del panel de objetos. Lo primero que nos encontramos es el nombre identificativo del objetivo, la posición de dicho objeto que en este caso se encuentra en el eje $x = -86$ y eje $y = 28$. Luego tenemos la opción de decidir si dicho objeto puede ser arrastrado o mostrado mediante un *checkbutton*.

encarga de establecer el fondo del mismo. Una función que se le suele encargar a este objeto es que sea el responsable de establecer las variables de inicialización[6] del programa.

3.6 Panel de programas

Cada objeto del entorno Scratch posee su propio panel de programas o *scripts* y en este panel como su propio nombre indica, es donde los usuarios de Scratch van a ir añadiendo funcionalidad al programa. Una de las principales razones que caracteriza Scratch frente a otros lenguajes de programación es que, en Scratch, no se escriben líneas de código, sino que se va arrastrando bloques predefinidos por el lenguaje con bastante parecido a las piezas de LEGO, las cuales simulan las características básicas de un lenguaje de programación.

En la esquina superior izquierda de la figura 3.10 se puede observar todos los bloques que ofrece Scratch. A continuación profundizaremos sobre los distintos bloques:

[6] Valores que se establecen al ejecutar un programa.

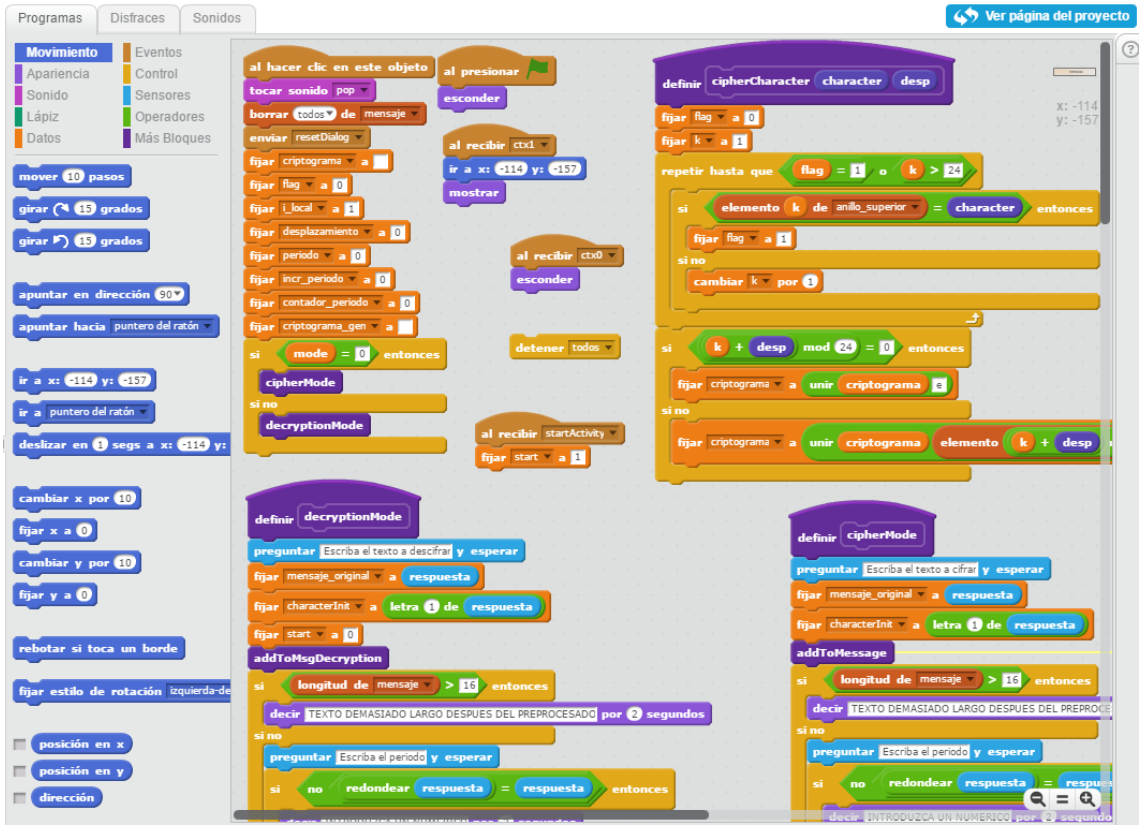

Figura 3.10: Ejemplo de código que se muestra en el «Panel de programas» cuando un usuario pulsa sobre un objeto. Este fragmento de código está extraído del cifrado del disco de Alberti.

- **Bloques de Movimiento.** Son los responsables de desplazar y detectar en qué posición se encuentra el objeto dentro del escenario del programa. Con ellos podremos mover un objeto desde una posición inicial a uno final. Normalmente se suele fijar la posición de las imágenes en una posición en concreto una vez se arranca el programa.
- **Bloques de Apariencia.** Dentro de este bloque nos encontramos con la opción de manipular la imagen a nuestro antojo. Por ejemplo, podemos aumentar el tamaño un 10 %, esconder dicha imagen, cambiar de disfraz, mostrar un texto...
- **Bloques de Sonido.** Este bloque permite la manipulación de los canales de audio. Con ello podremos aumentar el volumen, detener el sonido e incluso generar sonidos ya sea de la biblioteca del propio entorno de Scratch o un sonido importado desde una fuente externa.
- **Bloques de Lápiz.** Son los encargados de dibujar sobre el escenario desde el centro del objeto actual. Permite desde bajar y subir un lápiz de dibujo hasta cambiar el color de las líneas dibujadas.
- **Bloques de Datos.** Este bloque permite al usuario crear variables simples o más complejas como los vectores (*arrays* en inglés). Una característica que posee es que dichas variables se pueden declarar como locales de un objeto o globales. La diferencia es que si se declara como local, dicha variable solo podrá ser modificada por el objeto que ha creado la variable. En caso contrario, cualquier objeto del programa podrá modificar el valor de la variable.
- **Bloques de Eventos.** Son los encargados de enviar mensajes (o señales) al resto de objetos o recibir. Cuando se envía una señal existe un objeto que la genera y un

objeto u objetos que están esperando a ella para realizar la tarea predefinida, en caso de que se reciba la señal en concreto, como puede ser un fragmento de código que se encargue de mover el objeto, etc.

- **Bloques de Control.** Proporcionan las instrucciones de iteración y condición. Con ellos podemos generar fragmentos de códigos que incluyan una decisión (fragmentos si-entonces-si no), bucles o también fragmentos de espera, en los que la ejecución del objeto se detiene hasta que se cumpla una condición indicada por el usuario. Una funcionalidad que permite Scratch es el hecho de crear copias de un objetos, denominadas clones.
- **Bloques de Sensores.** Los sensores permiten detectar ciertas situaciones dentro del programa como identificar cuándo un objeto colisiona contra otro, qué teclas estamos pulsando en el momento actual y activar un cronómetro si así lo quisiéramos. Las funcionalidades que ofrece este bloque facilitan en gran medida la creación de cifrados.
- **Bloques de Operadores.** Los operadores permiten que los usuarios puedan verificar situaciones donde englobe que una variable tenga que tener un cierto valor en concreto para continuar la ejecución del programa, que un valor sea menor que otro, suma de variables, generación de números aleatorios comprendidos entre dos valores, acceder al carácter i-ésimo de una variable, etc.
- **Bloques de Más Bloques.** Scratch permite la generación de bloques adicionales donde estos bloques están formados por los bloque mencionados anteriormente, tal como se puede ver en la figura 3.11. Scratch tiene como objetivo simular los llamados métodos que poseen los lenguajes de alto nivel.

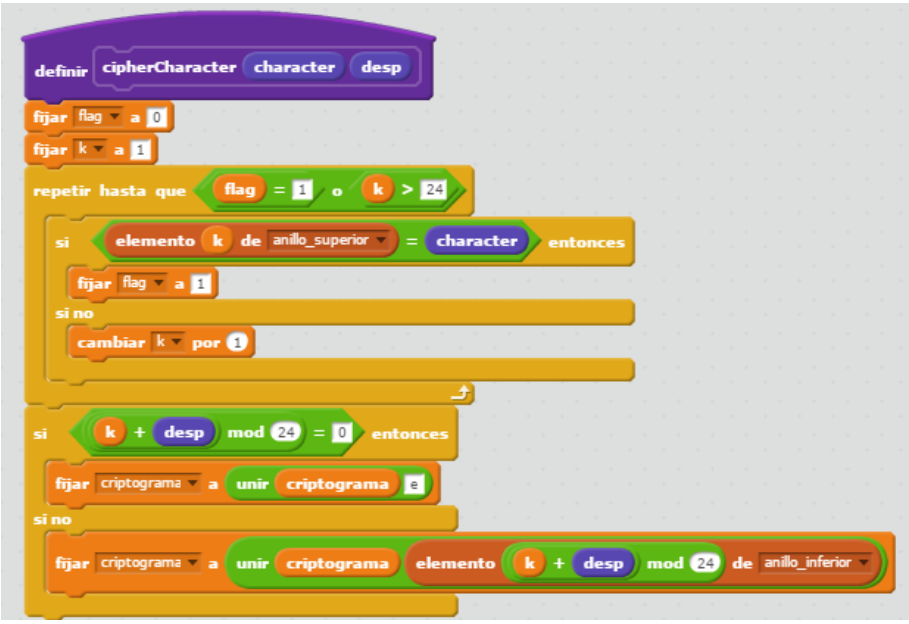

Figura 3.11: Bloque definido en Scratch con la función de sustituir un carácter por el carácter cifrado en el cifrado de Disco de Alberti.

3.7 Panel de disfraces

El panel de disfraces es el encargado de gestionar las distintas imágenes que puede tener un objeto de nuestro programa. Scratch, para simular la animación de los objetos, lo

que hace es cambiar de manera sucesiva los disfraces y así darle al usuario una sensación de movimiento. Por ejemplo, en la figura 3.12 se puede observar que el botón «Comenzar» tiene dos disfraces con el fin de que cuando el usuario pase el ratón sobre dicho botón (programado previamente), este cambie el color de fondo. Se ha implementado esta función para provocar una cierta retroalimentación a quien esté utilizando nuestro programa.

Figura 3.12: Panel de disfraces del botón «Comenzar» del cifrado de Alberti.

3.8 Panel de sonidos

El último de los paneles que vamos a mencionar es el panel de sonidos (véase la figura 3.13). Al igual que el panel de programas y de disfraces, cada objeto que estemos creando en Scratch posee a su vez su propio panel de sonidos. La tarea de este panel consistirá en controlar los distintos sonidos que cada objeto reproducirá al ejecutar el programa en cuestión.

A la hora de importar un sonido, ya sea desde la biblioteca de Scratch o de vuestro propio ordenador, cabe destacar que Scratch admite diversos formatos de audio como wav[7] o mp3[8]. Además de poder importar audios, existe la posibilidad de recortar los audios, añadir efectos o incluso grabar desde un micrófono su propio audio. Posteriormente, estos sonidos se pueden controlar desde los bloques de sonido.

3.9 Metodología en el desarrollo de los cifrados

En esta sección presentaremos los pasos que hemos seguido para realizar los cinco programas que trata el presente trabajo (Escítala espartana, Tablero de Polibio, Cifrado de César, Disco de Alberto y Ataque por fuerza bruta). Comenzaremos con el porqué

[7] Se conoce como wav al formato de audio sin compresión de datos.
[8] Formato de audio cuyos datos se encuentra comprimidos.

Figura 3.13: Panel de sonidos donde podremos importar sonidos desde la biblioteca de Scratch o de fuentes externas. También tenemos la opción de editar y añadir efectos a nuestros audios.

de la elección de los cifrados hasta la corrección de los fallos una vez implementados en Scratch.

El primer paso como cabe de esperar fue seleccionar qué cifrados clásicos de la historia íbamos a implementar en este trabajo. La selección no fue sencilla ya que queríamos que fueran cifrados que los jóvenes pudieran entender con cierta facilidad y además que abarcáramos diversos tipos de cifrado como los monoalfabéticos y polialfabéticos.

En segundo lugar se ha realizado la recopilación de toda la documentación posible, de sus características, de la complejidad y de las dificultades de los cifrados que podrían surgir al estar programando en un entorno limitado como Scratch. Para ello, previamente hemos tenido que aprender el lenguaje, aunque no ha sido especialmente complicado debido a que existe una gran cantidad de documentación en la propia web y en tutoriales que podemos encontrar en Internet.

El proceso de recopilación fue costoso debido a que se ha tenido que investigar cada uno de los cifrados, así como sus peculiaridades tanto en la forma de cifrar el mensaje y en el cómo descifrar el mensaje.

En cuanto al diseño de los objetos que dan lugar al programa, hemos tenido que empezar desde cero ya que no existían imágenes con una resolución aceptable y al importarlas al entorno Scratch estas se veían muy *pixeladas*. Por ejemplo, a la hora de desarrollar el disco de Alberti nosotros queríamos unas imágenes vectorizadas debido a la gran calidad de imagen que nos ofrece este formato. Para ello con la ayuda de una alumna de la Universitat Jaume I, Celia Soler Iranzo hemos tenido que diseñar estas imágenes en Adobe Illustrator como se puede ver en la figura 3.14.

En cuanto a la programación con Scratch, el principal objetivo es que los mensajes se cifrasen y se descifrasen correctamente. Para ello, antes de diseñar ninguna imagen en concreto, se trabajaba únicamente con las variables (véase la figura 3.15) para ver de este modo si el mensaje que vamos a encriptar se hacía tal como queríamos. Una vez dado el visto bueno comenzaríamos a incluir la interfaz definitiva para que el usuario final pueda usar el programa. En la figura 3.16 se puede ver el aspecto final.

Una vez tenemos la funcionalidad deseada, se diseña una pantalla de inicio para todos nuestros programas. Se ha intentado que sea común en todos los cifrados para que el usuario final una vez pruebe los programas se sienta cómodo. En nuestros menús principales tendremos un apartado de «Comenzar», «Instrucciones»,«Historia» tal como se

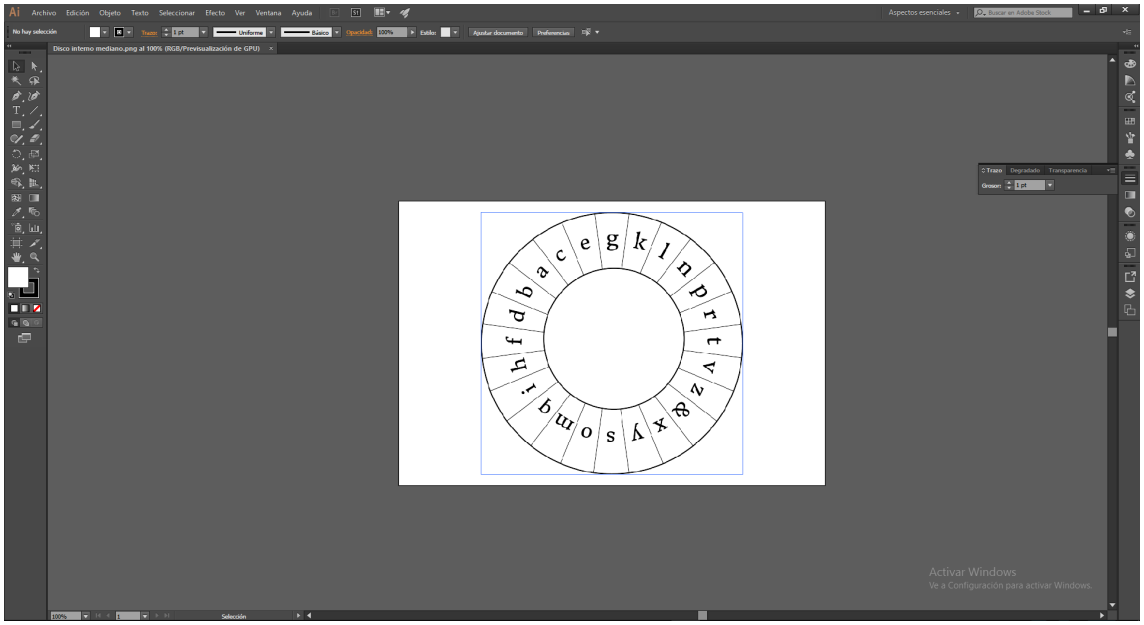

Figura 3.14: Creación del anillo interno en *Adobe Illustrator* para el cifrado del Disco de Alberti.

puede ver en la figura 3.17 junto al logo del Museo de Informática, de la ETSINF[9] y de la Universitat Politècnica de València.

Por último, una vez tenemos el programa con la interfaz se pasa a una fase de pruebas o de testeo con el objetivo de eliminar cualquier tipo de *bug*[10] ya que tenemos que asegurarnos que estos *bugs* desaparezcan y que los programas queden robustos antes de ser publicados en la web del Museo de Informática. Al no disponer de un equipo de *testers* como en las grandes empresas, hemos pedido a unos cuantos amigos que probaran los programas en busca de posibles errores.

3.10 Otros detalles

Cabe mencionar un detalle muy importante en Scratch y que nos afectará de cara a la creación de nuestros distintos cifrados. Debido a que Scratch está bastante limitado en cuanto a comparación con un lenguaje de más alto nivel como puede ser Java, el tratamiento tanto de las mayúsculas y minúsculas puede resultar muy engorroso y por ello hemos decidido que para nuestros programas este hecho puede estar dentro de las posibles ampliaciones.

Por otra parte, los caracteres como @, $, %, &, # entre otros, tampoco los trataremos y nos ceñiremos a los caracteres en minúsculas y números enteros reales debido a la dificultad extra que nos supondría, ya que el objetivo de este trabajo es mostrar los cifrados al público adolescente principalmente de manera didáctica.

[9] Escuela Técnica Superior de Ingeniería Informática

[10] Un *bug* es un error o fallo en el *Software* que provoca en el programa un resultado no deseado por el usuario.

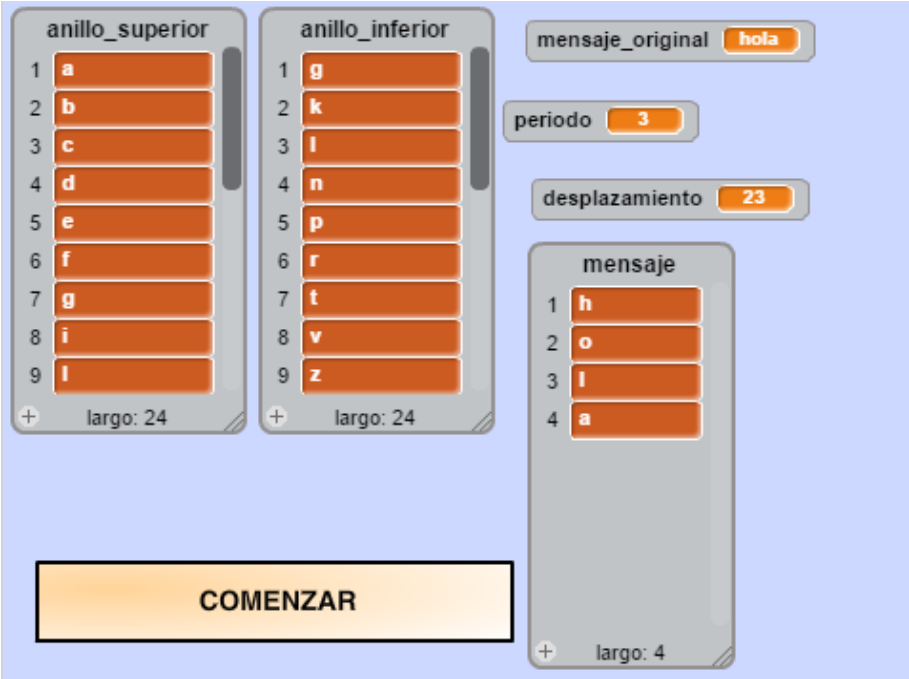

Figura 3.15: Captura del proceso de programación anterior a los diseños finales.

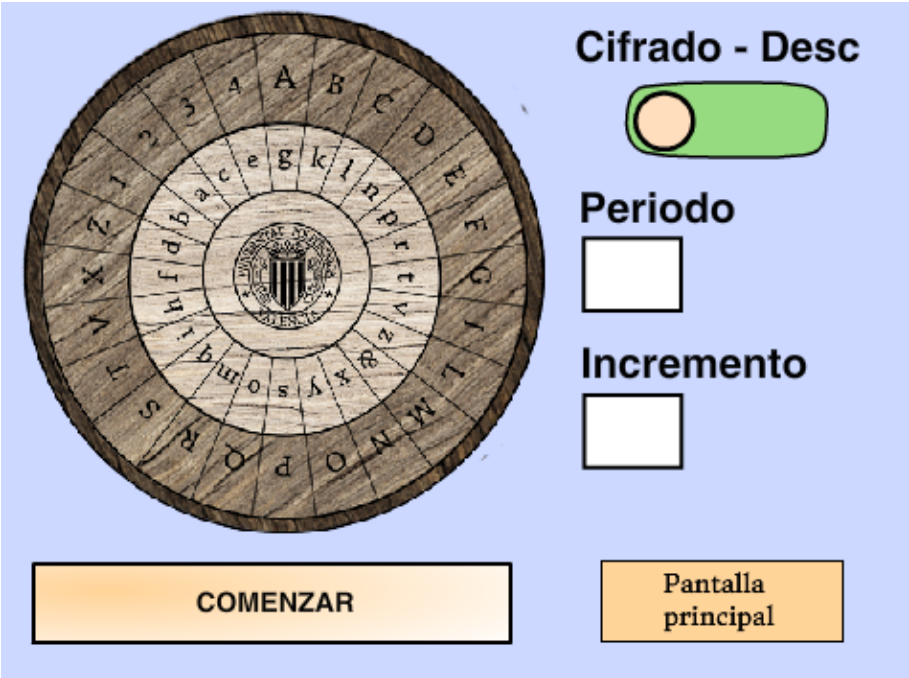

Figura 3.16: Captura de pantalla del Disco de Alberti.

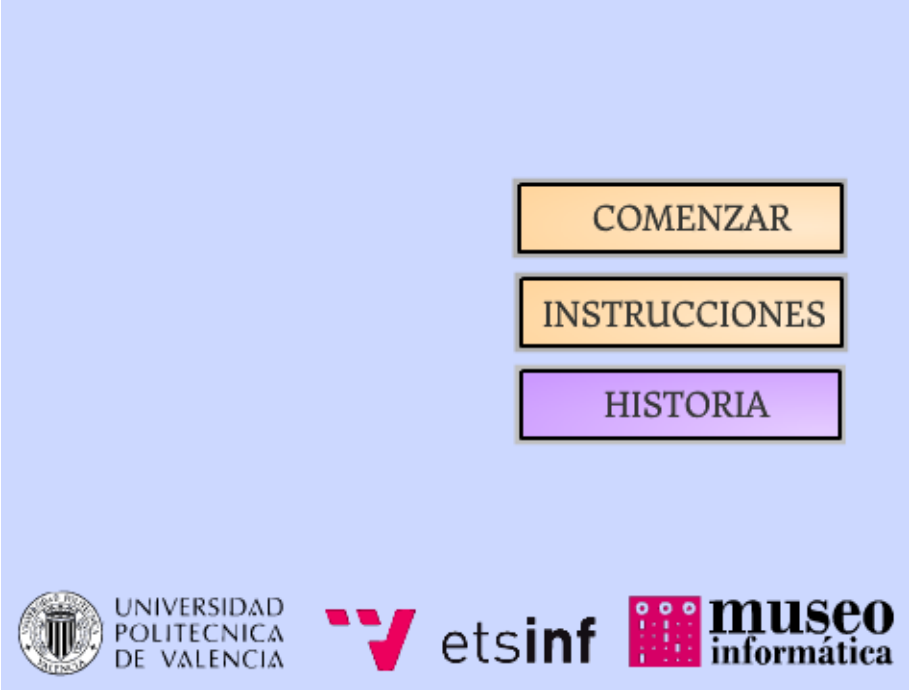

Figura 3.17: Menú principal generalizado de nuestros programas. En la figura podemos observar la estructura que va a seguir cada uno de nuestros cifrados implementados en Scratch. En el espacio vacío se añadirá una imagen representativa del cifrado en cuestión, en la parte inferior se encontrará el logo de la universidad, de la escuela de informática, del Museo de Informática y finalmente también tendremos las secciones que va a contener cada programa.

CAPÍTULO 4

Diseño e implementación de la Escítala espartana en Scratch

El primer cifrado que tiene lugar en este trabajo es la Escítala espartana. Este cifrado, como hemos mencionado anteriormente en el capítulo 2, fue el primer sistema criptográfico conocido históricamente. En este capítulo se describirá detalladamente todo el proceso seguido para poder implementar el cifrado.

4.1 Organización del menú principal

Al acceder al programa de la escítala espartana lo primero que se puede observar es el sencillo menú principal (véase la figura 4.1), que consta de tres apartados: «Comenzar», «Instrucciones» e «Historia». A continuación mostraremos qué funcionalidad ofrece cada uno de estos apartados:

Figura 4.1: Menú principal de la Escítala espartana. Tenemos los tres apartados bien diferenciados junto a la imagen del cifrado en cuestión y los logos.

- **Comenzar.** El primero y más importante, ya que es el que accede a la pantalla donde el usuario puede empezar a cifrar y descifrar sus mensajes. En la sección 4.2 se comentará con detalle el funcionamiento de todos los objetos que lo componen. El escenario será el de la figura 4.2.
- **Instrucciones.** Al pulsar sobre este botón nos llevará a un nuevo escenario donde brevemente se le explicará cómo funciona el programa dentro de Scratch.
- **Historia.** Se mostrará la historia y cómo surgió el cifrado.

4.2 Objetos del cifrado

El presente cifrado se compone de 27 objetos de los cuales seis serán los encargados de darle la funcionalidad al programa. A continuación se describirá su funcionamiento.

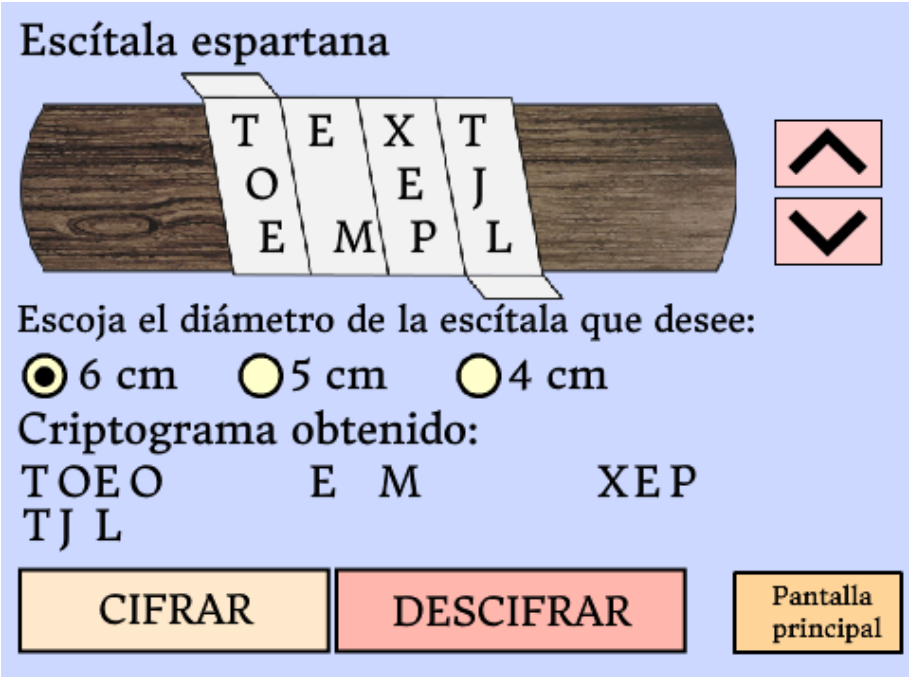

Figura 4.2: Captura de pantalla de la escítala espartana en el momento en el que se cifra 'TEXTO EJEMPLO' junto a su resultado.

1. **La escítala espartana.** La principal tarea de la escítala es mostrar el texto en la tela que hay sobre la vara. Con el fin de que el usuario pueda ver todo el texto que le introduciremos, hemos implementado la funcionalidad de rotar la vara (véase la figura 4.3). Básicamente cuando reciba la señal de «rotarEscitala» cambiaremos el disfraz por otro que no se vea la tela de la parte superior ni inferior creando una sensación de movimiento. Algunos os estaréis preguntando cómo rota el texto junto a la vara y bien, de esto se encargará el botón de «cifrar» que explicaremos unas cuantas líneas más abajo.
2. **Botón de selección.** También llamado *radio buttons* en inglés. Los *radio buttons* son los encargados de seleccionar el diámetro de la escítala y lo que se consigue es determinar el número de columnas y de caras que tendrá la vara, ya que con un diámetro más grande se puede interpretar que la tela se enrollará un menor número de veces (menos columnas).

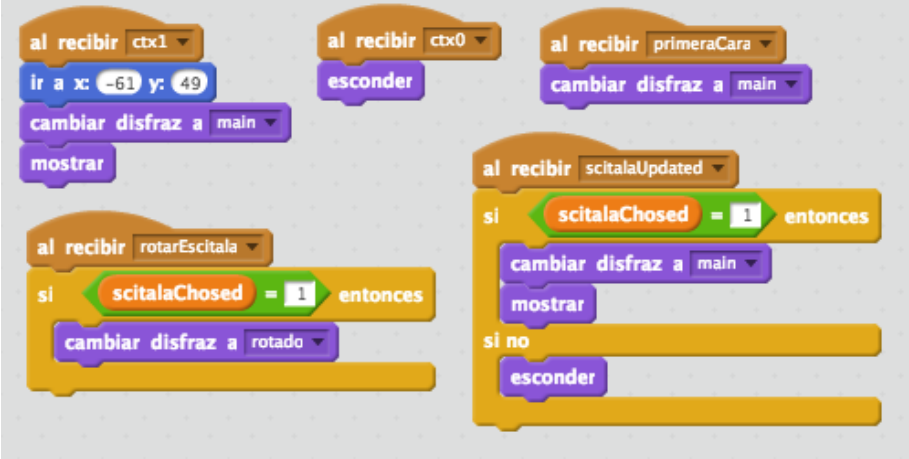

Figura 4.3: Fragmento de código asociado a la escítala espartana. Cuando el objeto reciba el mensaje «rotarEscitala», cambia el disfraz a otro con el fin de simular el movimiento de la rotación.

En nuestro programa lo hemos implementado (véase en la figura 4.4) tal que al pulsar sobre una opción, se actualizan las variables encargadas de determinar en qué diámetro estamos actualmente. Se puede apreciar en la figura que una vez pulsada una opción, establecemos el número de columnas correspondiente y también el número de caras, después cambiamos el disfraz de tal forma que se muestre el haberse pulsado el botón y enviamos además señales al resto de objetos para notificar sobre este hecho.

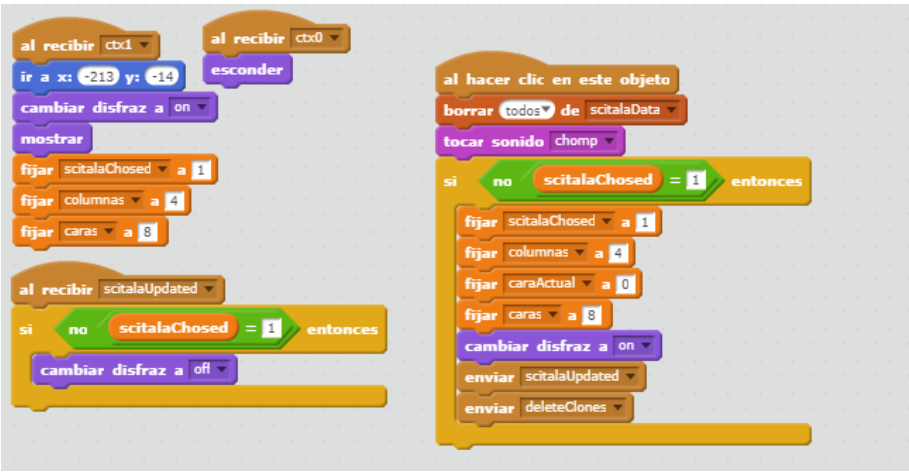

Figura 4.4: Fragmento de código asociado al objeto *radio button*. El usuario cuando pulsa dicho objeto, este actualiza las variables *scitalaChosed*, *columnas*, *caraActual* y posteriormente envía mensajes al resto de objetos tal como se puede ver en la figura.

3. **Botones de rotación.** La implementación de este objeto es muy sencilla. Ahora bien, al pulsar sobre este botón el programa realizará un sonido y dependiendo de qué botón hayamos pulsado (arriba o abajo) sumaremos o restaremos respectivamente la variable «caraActual», y posteriormente notificaremos al resto de los objetos de nuestra acción.

4. **Botón de cifrar.** A simple vista este botón puede parecer sencillo de implementar como los anteriores, pero aquí es donde reside la mayor complejidad de la implementación de este cifrado.

   El primer paso consistirá en, dado un mensaje, dividir este mensaje en tantas partes como caras tenga la escítala seleccionada y con una longitud determinada por la

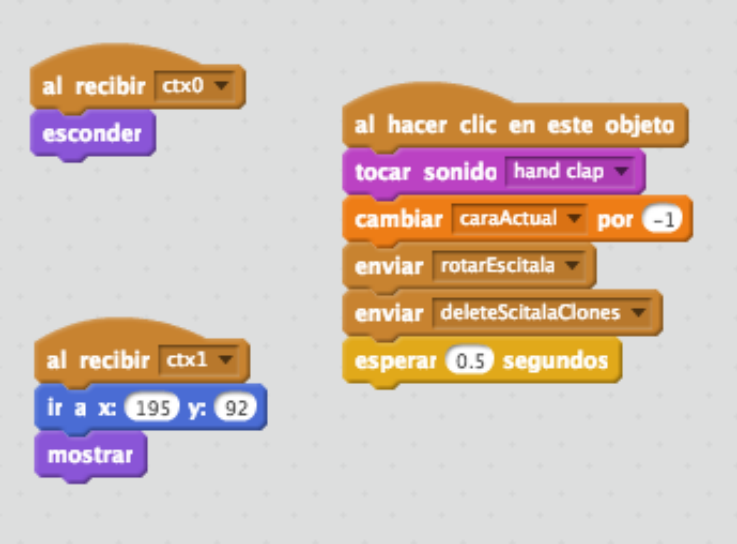

Figura 4.5: Fragmento de código asociado al botón de rotar la escítala. Tiene como objetivo sumar o restar la variable «caraActual».

variable «columnas». El resultado de este proceso lo podéis encontrar en la tabla 4.1, en este caso se ha optado por un total de siete caras. Dentro del fragmento de código, este proceso está implementado en el bloque «substring» de la figura 4.6.

El siguiente paso será cifrar el mensaje manipulando la matriz «scitalaData» e ir guardando los caracteres en la variable «res». Para ello accedemos a la primera posición de cada fila y una vez acabemos con la fila seguiremos con la segunda columna hasta que no queden más. El resultado final del cifrado, si nos fijamos en la tabla 4.1 y leyéndola de arriba hacia abajo, será *'EE OEL SST JO T EDE OUXEM NT P '*.

E	S	T	O	
E	S		U	N
	T	E	X	T
O		D	E	
E	J	E	M	P
L	O			

Tabla 4.1: Resultado del mensaje original en una matriz de siete elemento con cinco columnas. Así es como se representaría internamente la matriz en el cifrado de Alberti. El texto generado será el consiguiente al leer la matriz de arriba hacia abajo.

Por último, una vez tenemos el mensaje cifrado nos quedará únicamente escribir los caracteres sobre la tela. Para conseguir esto, por cada carácter del resultado se crea un clon y dicho clon lleva un identificador que, dependiendo de él, se posicionará en un lugar de la tela o en otro. En la figura 4.7 se puede visualizar este fragmento de código.

5. **Botón de descifrar.** Este botón implementa las operaciones inversas que el de «cifrar» (véase la figura 4.8). El primer paso que deberemos realizar será recuperar el

mensaje original desde un criptograma. En el proceso de recuperación guardaremos en una variable «mensajeOriginal» el mensaje original tras realizar el descifrado.

Este descifrado consiste en tomar el primer carácter del criptograma y concatenarlo con el carácter que está en la posición desplazado un número de veces correspondiente a la variable «caras» del criptograma.

Así pues, dado el criptograma *'EE OEL SST JO T EDE OUXEM NT P '* (del apartado de cifrar), el primer carácter es la 'E' y como sabemos que la escítala tiene siete caras en este caso, el mensaje original corresponderá los caracteres que se encuentren en las posiciones 1, 8, 15, 22, 29, 2, 9, 16... y así sucesivamente (cada iteración se realizará el número correspondiente a la variable «columnas») formando el mensaje 'E', 'S', 'T', 'O', ' ', 'E', 'S', ' '....

El siguiente paso será almacenarlo en forma de matriz dentro de la variable «scitalaData». Este paso es sencillo ya que, si por ejemplo, el número de columnas que hayamos escogido es cinco, bastará con fragmentar el mensaje original en ocho partes de cinco caracteres cada una. Posteriormente, teniendo la matriz, nos quedará escribir los caracteres sobre la escítala al igual que en el cifrado.

6. **Pantalla principal.** Este botón realiza una función muy sencilla y es la de volver a la pantalla principal ilustrada en la figura 3.17. Antes que nada tendremos que introducir la variable «ctx» (abreviatura de contexto). Esta variable indica en qué parte estamos del programa; por ejemplo, en la pantalla principal el contexto será $ctx = 0$, si estuviéramos en la pantalla de cifrado el contexto sería $ctx = 1$. En el caso del escenario de «Instrucciones» e «Historia» es prácticamente igual siendo el contexto 2 y 3 respectivamente.

   Explicado lo anterior en la figura 4.9, cuando se pulsa dicho botón cambia la variable «ctx» por un cero y se le notifica al resto de objetos.

4.3 Posibles ampliaciones

El cifrado ha sido prácticamente finalizado aunque como futura ampliación se podría simular el 'desenrollar' la tela y ver el criptograma directamente de la tela.

Otra opción que podríamos incluir en nuestro programa es ofrecer al usuario que introduzca él mismo el número de columnas y no restringirla en tan solo tres opciones (cuatro, cinco y seis columnas) como está implementado en este caso.

En cuanto al aspecto del sonido, se puede introducir una música de fondo de la Antigua Grecia, época de donde pertenece la escítala espartana. Finalmente, el apartado 3.10 habla de la dificultad de trabajar con tildes y mayúsculas en Scratch; así pues, esta parte la podría ampliar cualquier *scratcher* que tenga tiempo y ganas, ya que Scratch permite la reutilización de los programas.

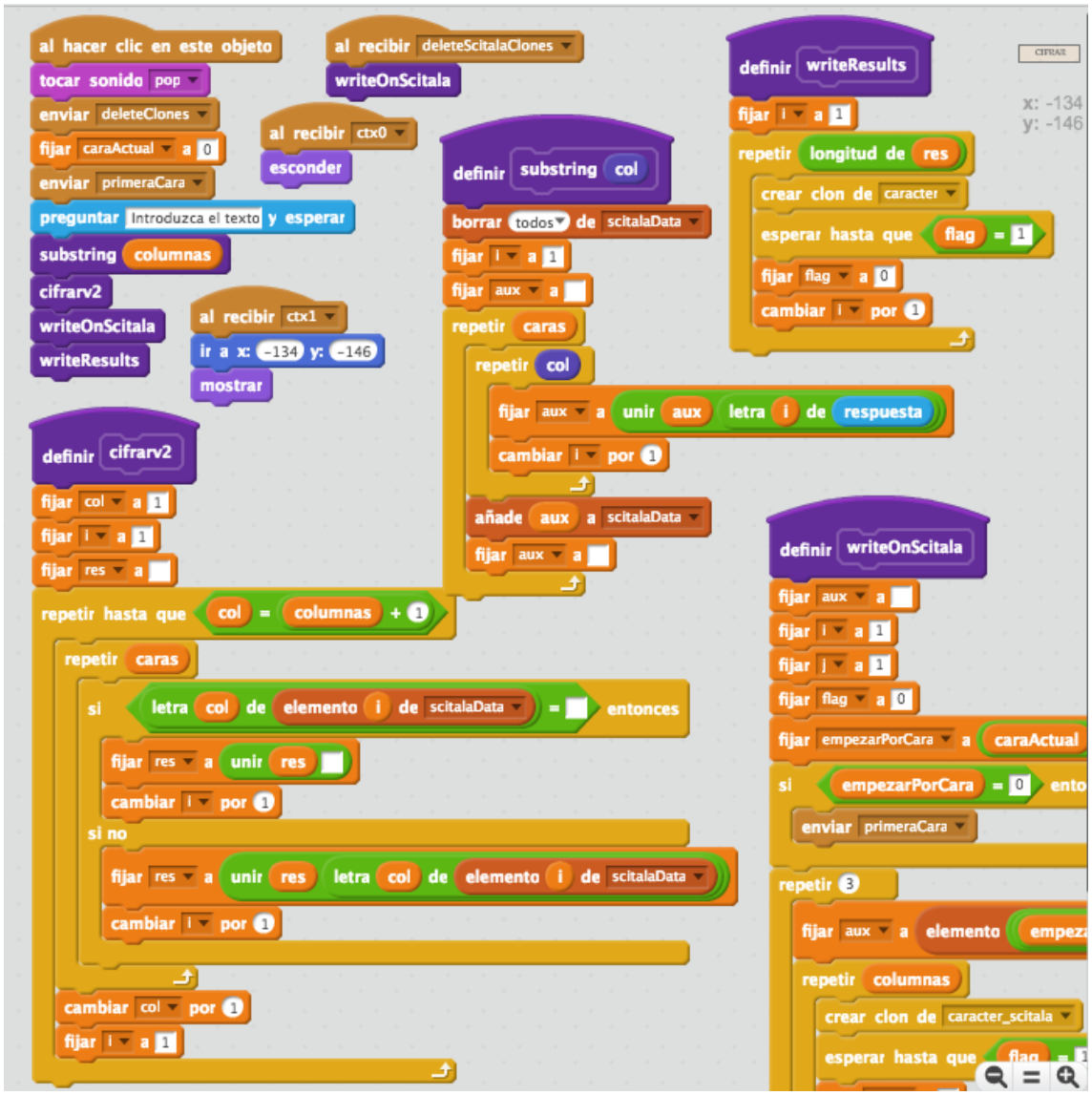

Figura 4.6: Fragmento de código asociado al botón cifrar. Como podemos observar en la figura, en el momento que pulsemos el objeto se eliminan los caracteres que previamente hayamos representado con «deleteClones» y se reinicia también la variable «caraActual» a cero. Con el método «substring» representamos el texto introducido en una matriz como la que tenemos la tabla 4.1 y el método «cifrarv2» se encarga de realizar el proceso iterativo de manera que se guarda el criptograma en la variable «res». Finalmente con «writeOnScitala» y «writeResults» mostramos el resultado en pantalla creando tantos clones hagan falta.

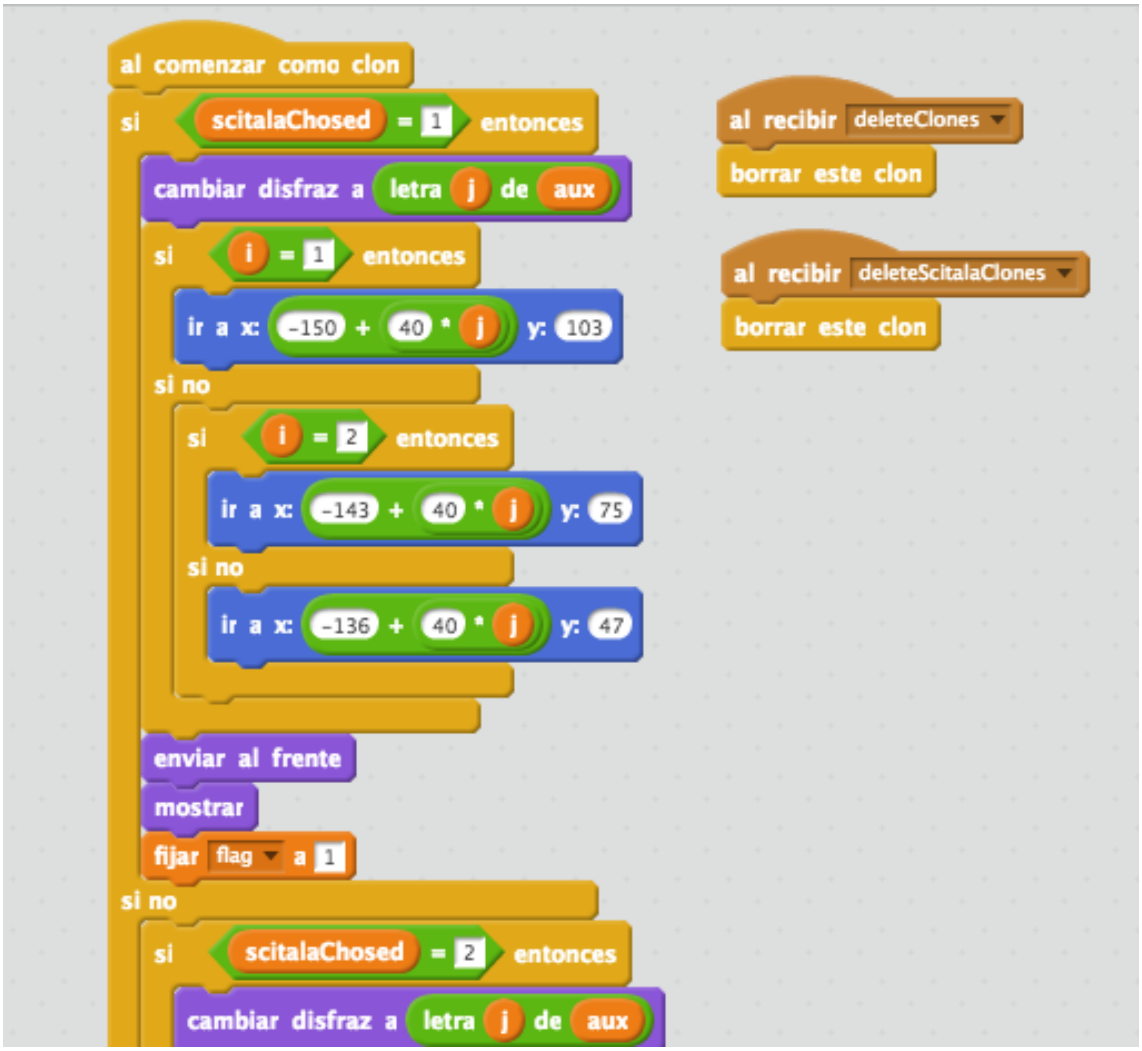

Figura 4.7: Fragmento de código asociado a los clones de la Escítala espartana. En el instante en el que se crea un clon, este cambiará el disfraz a la letra que le pertenece según la variable «aux». Por ejemplo, si la variable «aux» contiene el texto 'HOLA', el primer clon cambiará el disfraz a la imagen de una 'H' y posteriormente se situará en la posición correspondiente.

Figura 4.8: Fragmento de código asociado al botón descifrar. El proceso es casi idéntico al explicado en la figura 4.6 pero con la diferencia de que en este fragmento tenemos dos nuevos métodos: «recuperarMensajeOriginal» y «txtToArray» . El método «recuperarMensajeOriginal» se encarga de leer el criptograma de manera que reordene de manera correcta. Para ello se empieza desde el primer carácter de dicho criptograma y se guarda en la variable «mensajeOriginal», después se desplaza ocho posiciones para extraer el segundo carácter y asi hasta finalizar la primera iteración. El número de iteraciones corresponde con el número de columnas seleecionado previamente por el usuario.

Figura 4.9: Fragmento de código asociado al botón de «Pantalla principal». El único objetivo que tiene este objeto es volver al menú principal. Para ello, cuando se pulsa, fija la variable «ctx» a cero y envía «ctx0» para que el resto de objeto realice las acciones pertinentes (esconder el objeto, mostrarse...).

CAPÍTULO 5
Diseño e implementación del Tablero de Polibio en Scratch

En este capítulo se va a describir el proceso seguido para elaborar los distintos objetos que componen el programa con el objetivo de simular el cifrado del Tablero de Polibio.

5.1 Organización del menú

El menú (véase la figura 5.1) consta de tres opciones al igual que la implementación del cifrado de la escítala espartana vista en el capítulo 4.1. Estas opciones son: «Comenzar», «Instrucciones» e «Historia» y se describirán a continuación:

- **Comenzar.** Al pulsar sobre este botón, llevará al usuario a la pantalla que se puede ver en la figura 5.2, en dicha pantalla el usuario podrá encriptar un mensaje con una longitud máxima de 11 caracteres debido a que, con las limitaciones de Scratch, el criptograma generado se saldría de la pantalla (el criptograma en el cifrado de Polibio se duplica ya que un carácter del mensaje original se sustituye por dos dígitos correspondiente a la fila y columna de la matriz). Por esta razón hemos establecido un máximo de 11 caracteres para el mensaje original y un máximo de 22 para el criptograma a la hora de descifrar.
- **Instrucciones.** Al seleccionar esta opción se dará al usuario una breve explicación sobre el funcionamiento de este cifrado en el entorno de Scratch.
- **Historia.** Dentro de «Historia», se introducirá al usuario en el cifrado de Polibio y cuando surgió en la historia.

5.2 Objetos del cifrado

Este cifrado se compone de 20 objetos que dan lugar el resultado final del programa. De los 20 objetos, dos son los objetos principales que dan funcionalidad al programa (véase la figura 5.2) que los presentaremos a continuación:

- **Botón de conmutación o *toggle switch*.** Antes de comenzar, se debe saber que dentro de nuestro programa tenemos una variable denominada «modo» que se encarga de decirnos en qué modo se encuentra el programa en el momento actual. Un valor '0' corresponde al modo de cifrar y un valor '1' en el de descifrado. En la figura

Figura 5.1: Menú principal del cifrado de Polibio.

5.3 se puede apreciar el fragmento de código del objeto, que tiene como único fin actualizar en el momento que el usuario pulsa dicho botón.

- **Botón de comenzar.** Se trata del objeto más importante del programa ya que es el principal encargado de dar funcionalidad al programa. Una vez pulsemos el botón nos pedirá que introduzcamos un texto de longitud máximo de 11 ya que en caso contrario el criptograma se desbordaría de la pantalla del programa. Esto es uno de los inconveniente de trabajar en Scratch ya que está bastante limitado.

  En el caso de que quisiéramos descifrar un mensaje, se nos pedirá que introduzcamos una cadena de dígitos cuya longitud máxima sea de 22 y, esto es importante, que sea par, ya que de otro modo el último par de dígitos se quedaría sin el componente columna y no podríamos asociar a ningún carácter del tablero de Polibio ya que, como sabemos, un carácter se genera mediante una fila y una columna.

  En la figura 5.4 podemos ver una parte de código del objeto «Comenzar»; al pulsar el botón se eliminan todos los espacios en blanco del mensaje inicial con el bloque «removeBlankSpace». Más adelante realizamos una comprobación para ver si el texto supera los 11 o 22 caracteres, dependiendo de si estamos cifrando o descifrando. El siguiente paso será hallar en qué fila y columna se encuentra el carácter que estemos cifrando en ese momento y se guardará en una variable «i» y «j» que se puede apreciar en el método «contains». Así pues, el último paso que deberemos realizar en Scratch es almacenar el carácter cifrado en una variable, en este caso, en «criptograma».

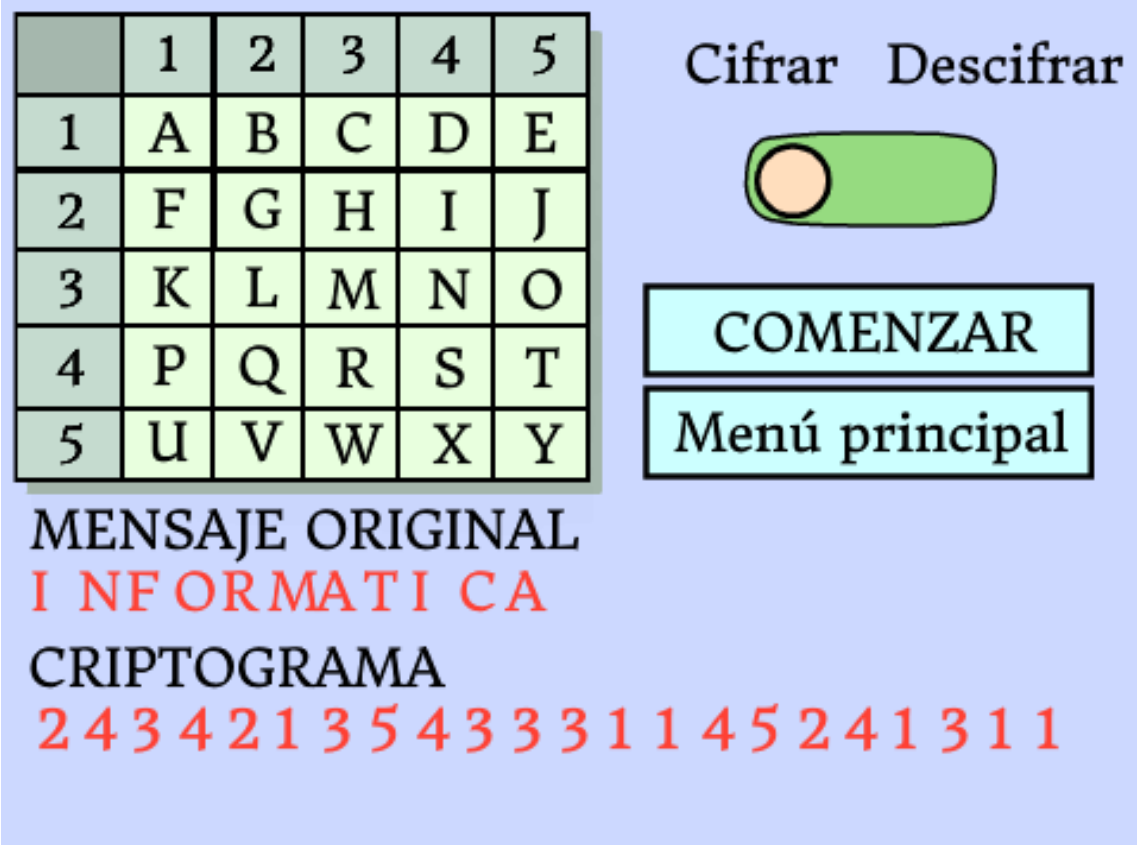

Figura 5.2: Pantalla del cifrado de Polibio. En ella podemos observar que el texto 'INFORMATICA' se encripta de la siguiente manera. Por ejemplo, la 'I' se ha sustituido por un '24' correspondiente a la segunda fila y cuarta columna, posición donde se encuentra la 'I'.

5.3 Posibles ampliaciones

Una funcionalidad que podemos ampliar en un futuro será dejar a los usuario la elección de la clave del cifrado, es decir, la organización de los carácteres del alfabeto en el tablero. Viendo la figura 5.5 la idea sería dejar libertad al usuario de poder pulsar sobre cada una de las casilla y añadir el carácter que quisiese.

Por otra parte una mejora también podría ser permitir al usuario el escribir mensajes más largos y aceptando carácteres en mayúsculas aunque siendo el proyecto acotado en un tiempo fijo, el hecho de añadir esta funcionalidad extra no salía a cuenta debido al tiempo que se requería en implementarla y la poca mejora que supondría de cara al usuario final.

El objetivo inicial que perseguíamos con nuestro programa es enseñar el funcionamiento de este cifrado a los «*scratchers*» o usuarios así pues la opción de generar la clave del cifrado sería una ampliación que añadir a nuestro programa.

Figura 5.3: Captura de pantalla correspondiente al fragmento de código del objeto «toggle switch». Cuando pulsamos sobre dicho objeto, dependiendo del modo en el que estemos el programa fijará la variable «modo» al valor '1' o al '0', tal como se puede ver en la figura.

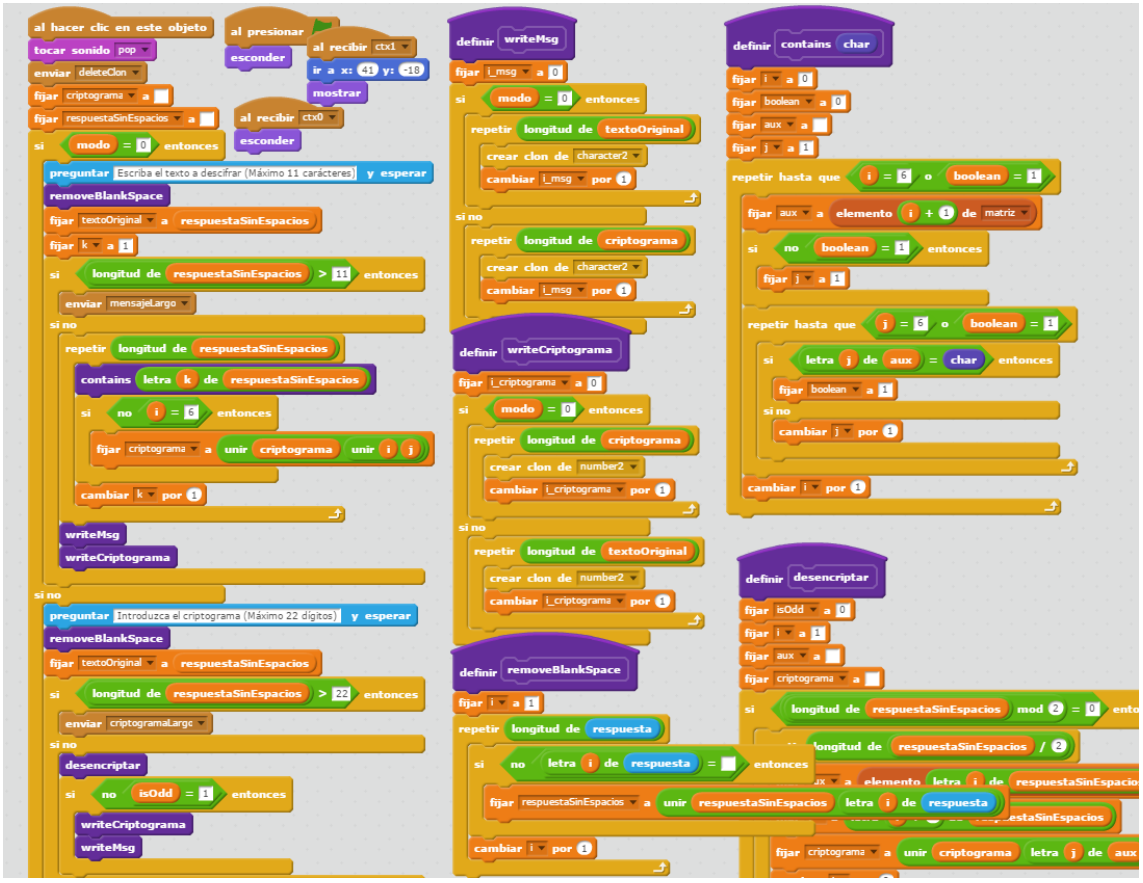

Figura 5.4: Captura de pantalla del código asociado al objeto «Comenzar». En primer lugar, el programa realiza una comprobación para ver en qué modo se encuentra actualmente, esto se puede observar en la quinta línea del código. Si estamos en el modo de cifrar, el programa eliminará los espacios en blanco con el método «removeBlankSpace» y posteriormente se nos restringirá que el tamaño del texto introducido no supere las 11 letras. El método «contains» se encarga de buscar en que posición se encuentra cada carácter del texto. Así pues, si el primer carácter es una 'A', viendo nuestro tablero de Polibio el método «contains» nos devolverá $i = 1$ y $j = 1$. Por último, se añade la i y la j a nuestra variable «criptograma» para ser mostrado al usuario.

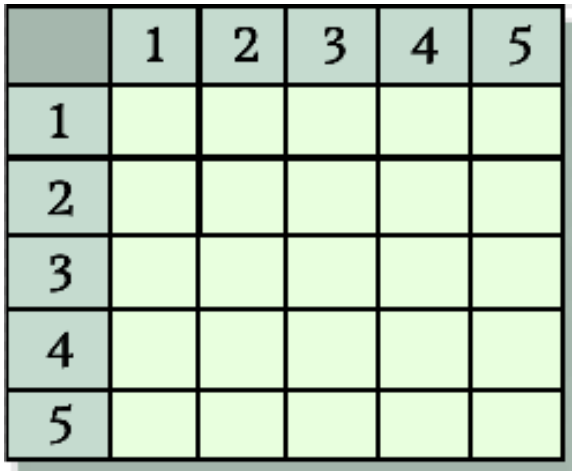

Figura 5.5: Tablero de Polibio con los espacios vacíos. Posible implementación para futuras versiones del programa, con el fin de que el usuario pueda introducir los caracteres que desee a la hora de establecer una clave para el cifrado.

CAPÍTULO 6

Diseño e implementación del Cifrado de César en Scratch

En este capítulo se describirá, tal como hemos hecho hasta ahora en los capítulos anteriores, el procedimiento seguido para la elaboración del cifrado de César, el que usaba el general de Roma en el campo de batalla para realizar sus conquistas.

Cabe destacar que debido a las limitaciones que posee Scratch en el aspecto de las mayúsculas y las tildes (mencionado en el apartado 3.10), el programa que hemos implementado en Scratch solo aceptan letras minúsculas y sin tilde.

6.1 Organización del menú

Dentro del programa en cuestión nos encontraremos con un total de 29 objetos entre los cuales se encuentran desde objetos simples hasta complejos.

El menú del cifrado de César (véase la figura 6.1) consta de cuatro secciones («Cifrar», «Descifrar», «Instrucciones» e «Historia»), en el que cada una de estas secciones se han creado un objeto con el fin de simular un botón. A continuación os mostraremos la funcionalidad que implementa cada botón en el programa:

- **Botón de cifrar.** Este es el más importante junto al de descifrar ya que son los encargados de ofrecer funcionalidad al programa.

  En el primer caso (véase la figura 6.2) el usuario podrá introducir el mensaje que desee encriptar con una longitud máxima de 20, ya que de no restringirse el tamaño del texto, este se desbordaría de la pantalla. Por otra parte, permitiremos al usuario introducir una clave de cifrado comprendida entre 1 y 26.

  Cabe destacar que hemos implementado diálogos de información con el objetivo de que el usuario no cometa errores innecesarios. Por ejemplo, en la figura 6.2 se puede ver que si pasamos con el ratón sobre la «i» se muestra un cuadro informándonos del tamaño máximo permitido.

- **Botón de descifrar.** Cuando presionamos sobre este botón se nos envía a un escenario similar a la figura 6.2 pero con una diferencia y es que en el texto del botón inferior está escrito 'Descifrar' en vez de 'Cifrar'. Hemos diseñado este escenario de tal manera para que el usuario se sienta familiarizado con el entorno y el proceso de aprendizaje sea mínimo.

  La funcionalidad del descifrado es igual que la de cifrado con la peculiaridad de que realiza el proceso inverso, es decir, si en el proceso de cifrar una 'A' se sustituye

Figura 6.1: Pantalla inicial del cifrado de César.

por una H, en el descifrado una 'H' se sustituiría por una 'A' si introdujéramos la clave correctamente.

- **Botón de instrucciones.** Tal como se vio en capítulos anteriores se repasará brevemente el funcionamiento de los objetos existentes en el programa, es decir, los pasos que se han de seguir.
- **Botón de historia.** La sección de «Historia» muestra al usuario, como su propio nombre indica, un recorrido por la historia del cifrado de César.

6.2 Objetos del cifrado

En este apartado se hablará de los objetos más importantes que componen el programa así como la manera en la que están implementados en el entorno de Scratch mostrando al lector el fragmento de código asociado a los distintos objetos. A continuación se describirá con detalle cada objeto:

- **Botón de «Escribir texto».** Para comenzar a cifrar el primer paso que se debe realizar es pulsar este objeto. Una vez que lo pulsemos se nos abrirá un cuadro donde introduciremos nuestro mensaje. En el momento que finalicemos, el texto se visualizará en pantalla tal como se ve en la figura 6.5. Si nos centramos en el cómo está implementado dicho objeto en Scratch podemos ver en la figura 6.3 que una vez pulsemos el objeto, el programa realizará una comprobación de la longitud del texto introducido y posteriormente se añadirá en un *array* «mensaje» y de esta manera, a la hora de crear clones de cada carácter del mensaje original será mucho más sencillo ya que cada clon se encargará de representar el carácter asociado a su identificador como clon.
- **Clave.** La clave en el cifrado de César sirve para saber cuántas posiciones se desplaza cada carácter en el mensaje original.

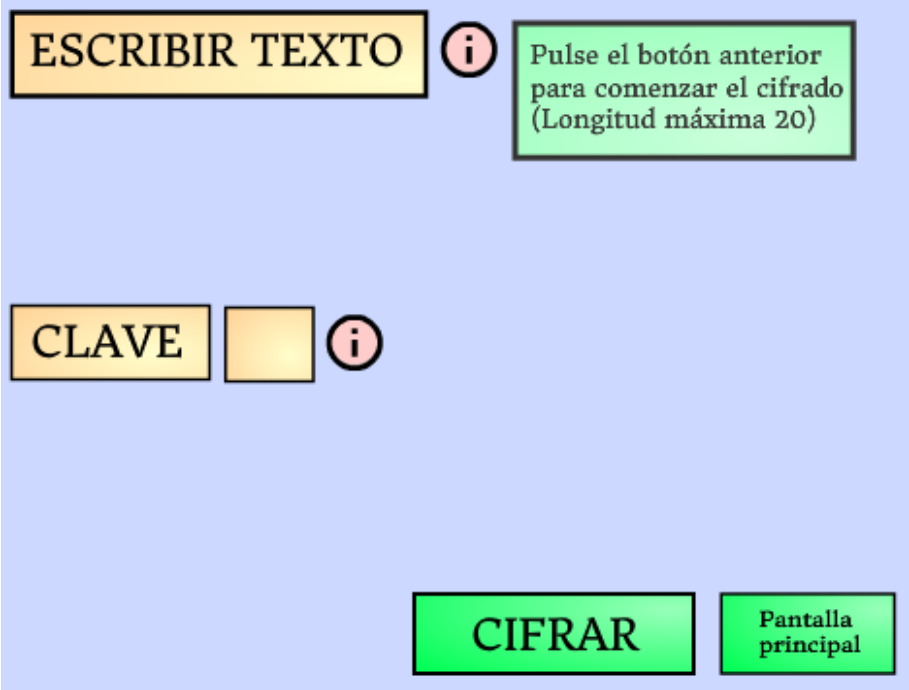

Figura 6.2: Escenario principal del cifrado de César. Accederemos a esta pantalla una vez el usuario haya pulsado el botón «Cifrar» del menú principal.

En el programa, el objeto «Clave» posee una comprobación de datos; esto quiere decir que si se introduce un carácter que no sea numérico y que no esté dentro del rango permitido (0..26), el programa mostrará un diálogo de advertencia avisándonos del problema (véase la figura 6.5). La comprobación de datos se puede ver en la figura 6.4.

- **Cifrar.** Si hemos seleccionado la opción de «Cifrar» en nuestro menú principal, este objeto aparecerá en la parte inferior del escenario. Es el encargado de dar comienzo al proceso de cifrado una vez tengamos el mensaje original y la clave correctamente introducidos.

  En la figura 6.6 podemos ver que, el primer paso que realizamos es la comprobación de los datos introducidos por parte del usuario. Existen dos bucles dentro del código, el primero de ellos toma el *array* «mensaje» para acceder a cada uno de sus elementos realizando el desplazamiento según un número k correspondiente a la clave del cifrado; esto se almacena a su vez en la variable «criptograma». Por ejemplo si el primer elemento de «mensaje» es la 'A' y tenemos una clave $k = 2$, entonces en el primer elemento de «criptograma» habremos añadido el carácter 'C' debido a que es el carácter 'A' desplazado dos posiciones en el alfabeto. El último paso será crear tantos clones como caracteres haya y mostrarlos en pantalla.

  Una característica que hemos implementado en Scratch es que durante el proceso se muestra una señal de «Procesando...» con el objetivo de darle una retroalimentación al usuario y saber que el programa se está ejecutando. De otro modo podrá pensar que ha habido un error en caso de no producirse nada por pantalla.

- **Descifrar.** Este objeto aparecerá únicamente si presionamos el botón de «Descifrado» en el menú principal. La funcionalidad es idéntica a la del cifrado pero realizando el proceso inverso así pues, el fragmento de código es igual exceptuando algunas diferencias (realizar el desplazamiento inverso).

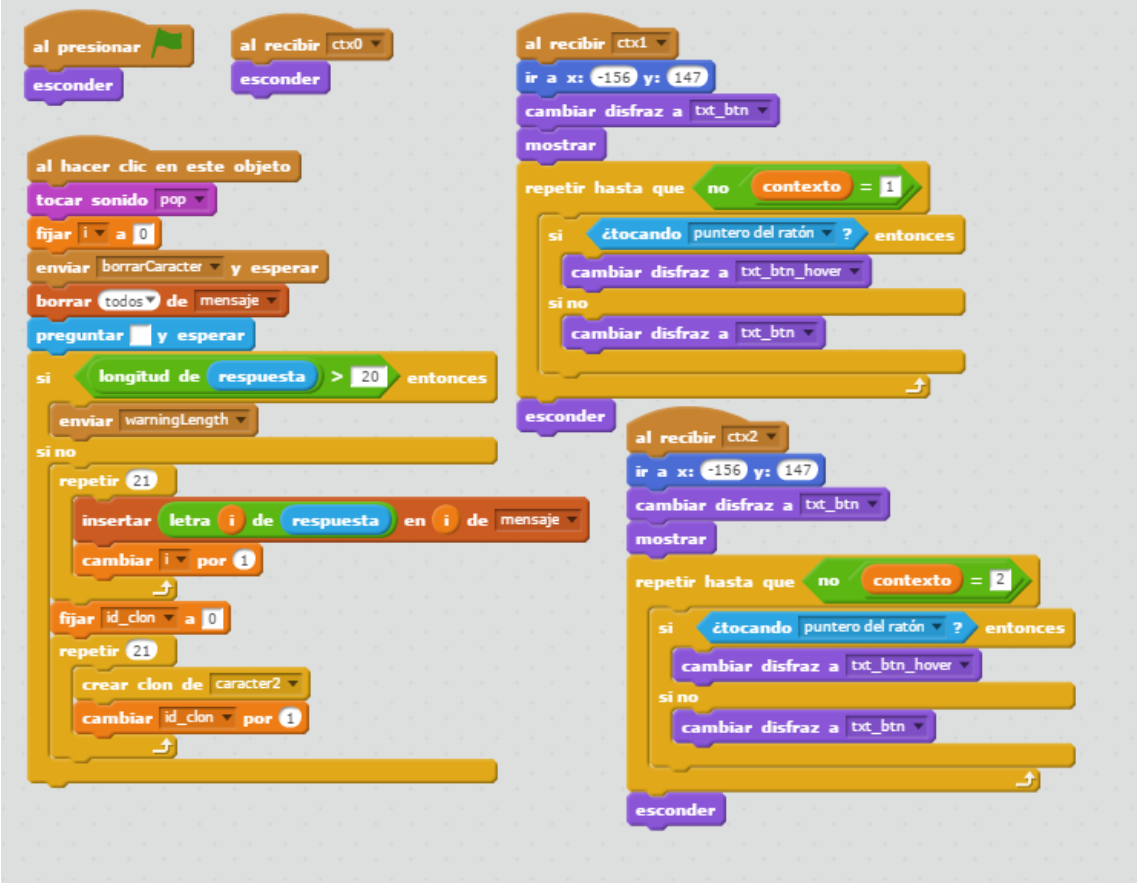

Figura 6.3: Fragmento de código asociado al botón «Escribir texto». En el momento que el usuario pulsa este objeto, reiniciaremos todas las instancias (objetos) que hayamos creado en el programa a su valor inicial. El siguiente paso es comprobar la talla del texto introducido que en el caso que supere los 20 caracteres se mostrará un diálogo de aviso. Si todo esta correcto, el programa realizará un proceso iterativo donde se creará cada uno de los clones que se encargan de mostrar el texto en pantalla.

- **Cuadro de diálogo.** Este objeto (véase la figura 6.5) se ha creado con la intención de proveer retroalimentación al usuario. La implementación de este objeto consiste en que cuando dicho objeto reciba una señal se muestre por pantalla gracias a la instrucción «mostrar» que ofrece Scratch dentro del bloque «Apariencia» que ya hemos descrito en el apartado 3.6.

- **Pantalla principal.** Este objeto no proporciona ninguna funcionalidad extra al programa y se encarga principalmente de la navegación, ya que gracias a él, el usuario se podrá mover por los distintos escenarios de los que se componen el programa.

6.3 Posibles ampliaciones

Como hemos mencionado en el comienzo del capítulo, el programa posee ciertas limitaciones en cuanto a las mayúsculas, si se intenta escribir un mensaje que contengan mayúsculas, este se ignorará y donde se tendría que escribir 'Hola' se escribirá '_ola'

Por otra parte, en nuestro programa hemos restringido también el tamaño máximo de texto a introducir y la idea de una posible ampliación sería crear clones con un tamaño menor con el fin de poder escribir un número mayor de caracteres dentro del cifrado. De

Figura 6.4: Fragmento de código asociado al objeto «Clave». Si nos fijamos en el código podemos observar que realiza una comprobación de datos en la instrucción condicional (si «range» contiene «respuesta»? entonces). Esta instrucción será verdadera si la clave introducida es un número comprendido entre 0 y 26. En caso contrario mostrará al usuario un diálogo de advertencia que se puede ver en la figura 6.5.

este modo, en vez de limitarse a 20 caracteres, si redujéramos a la mitad cada carácter se duplicaría el texto y podríamos formar un mensaje con más contenido en él.

Por último, aunque no menos importante, quien quiera puede acceder a nuestro código interno del programa para rediseñar la interfaz y realizarlo de modo que incorpore objetos con una temática más característica de la época romana.

Para futuras versiones del programa se podrían incluir estas características siempre y cuando no afecten al rendimiento.

Figura 6.5: Diálogo de aviso en el cifrado de César. Se muestra cuando el usuario introducie una clave no válida, es decir, cualquier carácter que no esté comprendido en el rango [0-26]

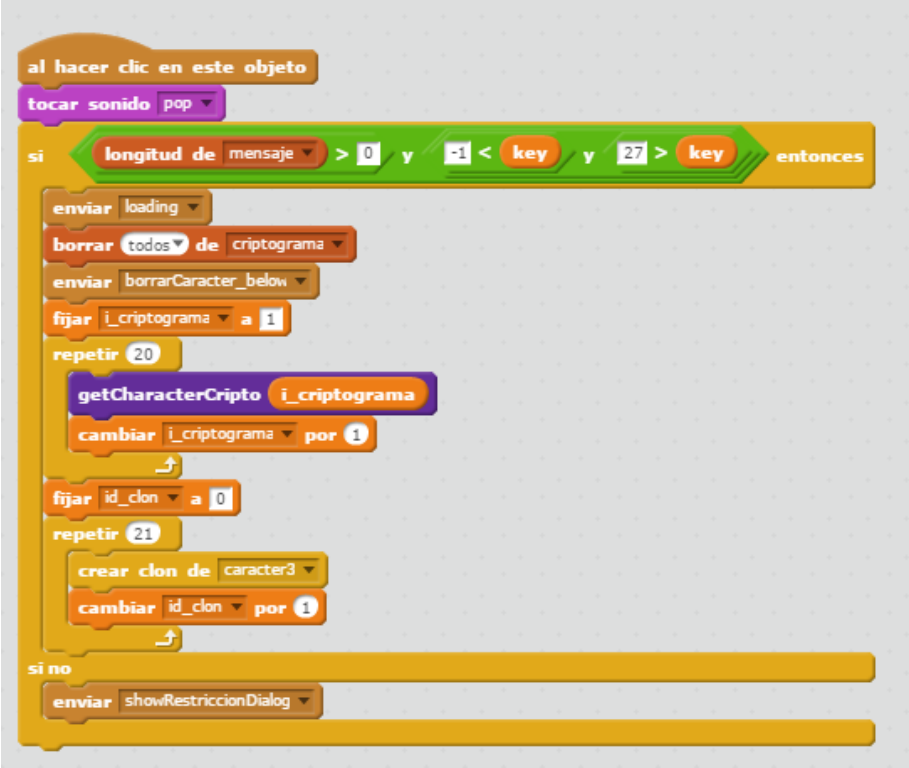

Figura 6.6: Fragmento de código asociado al objeto «Cifrar». Una vez tenemos el texto y la clave, el siguiente paso será empezar a cifrar el mensaje. En la figura se puede observar que realiza una iteración de 20 con el objetivo de cifrar cada carácter del mensaje siguiendo el método «getCharacterCripto». Posteriormente se crean tantos clones hagan falta, tal como hemos estado haciendo hasta ahora.

CAPÍTULO 7

Diseño e implementación del Disco de Alberti en Scratch

En este capítulo se hablará sobre el proceso de implementación del último de los cifrados realizados en Scratch, el disco de Alberti. Cabe mencionar que este ha sido el cifrado más complejo de implementar ya que, a diferencia del resto, este cifrado es polialfabético. Además del funcionamiento del cifrado, una complejidad añadida ha sido la creación del disco desde cero y la simulación de este en Scratch. En las siguientes secciones se describirán los objetos de nuestro menú principal, los objetos del escenario del cifrado y por último hablaremos sobre las posibles mejoras que se puede incorporar al programa.

7.1 Organización del menú

El menú principal del disco de Alberti consta de tres apartados bien diferenciados: «Comenzar», «Instrucciones» e «Historia», además de los logos de la UPV, ETSINF y el museo de informática. A continuación se va a describir cada uno de ellos:

- **Comenzar.** Este objeto será el encargado de que el usuario pueda acceder a la pantalla de cifrado (véase la figura 3.16). Esta pantalla consta del disco con el que cifraremos y descifraremos los mensajes, un *toggle switch* y una serie de parámetros (periodo e incremento). En la sección 7.2 se explicará el funcionamiento de estos objetos.
- **Instrucciones.** Al pulsar sobre este objeto se lanza la señal «ctx2» que se encarga de avisar al resto de objetos de que se ha actualizado el escenario y pueda realizar las acciones correspondientes.
- **Historia.** En el caso de pulsar este objeto se lanzará la señal «ctx3» y cambiará el escenario donde se mostrará la historia del cifrado. Se describirá el contexto histórico.

7.2 Objetos del cifrado

En la figura 7.1 se compone de siete objetos importantes que darán lugar al cifrado y que os describiremos a continuación:

- **Disco de Alberti.** Se trata de un disco compuesto por dos anillos de 24 celdas. Cada una de las celdas contiene un carácter que usaremos a la hora de cifrar o descifrar el

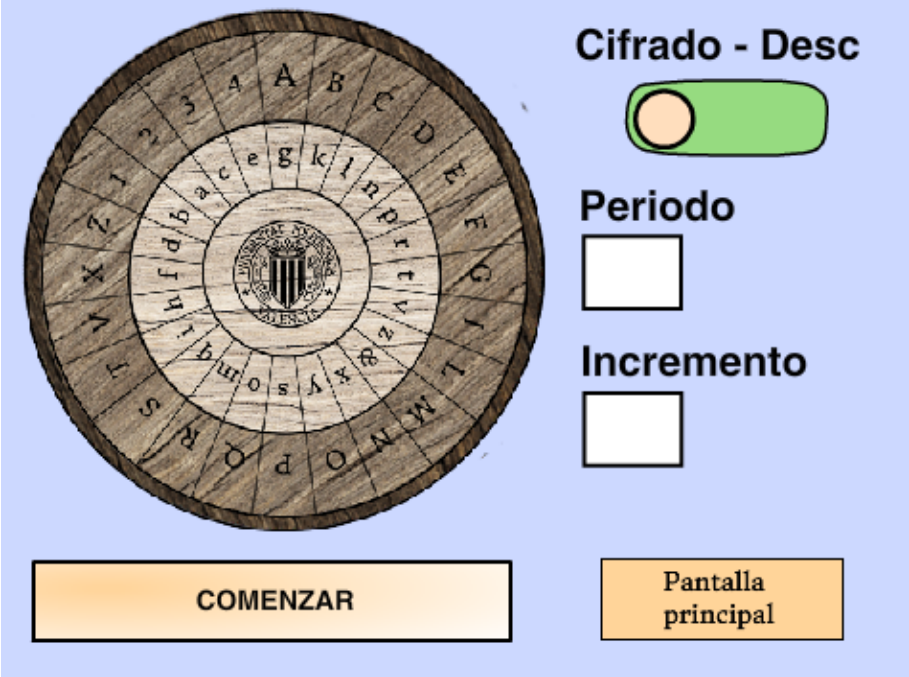

Figura 7.1: Menú de cifrado y descifrado del Disco de Alberti.

mensaje. Este objeto será quien se encargue de rotar el disco cuando reciba la señal «rotarAnillo» tal como se ve en la figura 7.2.

- **Botón de conmutación o *toggle switch*.** Este objeto sirve para seleccionar en qué modo estamos dentro del programa. Por ejemplo, una vez accedemos a la pantalla de cifrado nos encontraremos en el modo de cifrado representado por la variable «mode» ('0' = cifrado, '1' = descifrado); este es el modo por defecto (véase la figura 7.3). Al pulsar sobre este objeto actualizaremos la variable y además cambiaremos el disfraz de este objeto para mantener la coherencia.

- **Periodo.** Tal como se explicó el concepto de periodo en el apartado 2.4.2, en el programa este objeto es el encargado de mostrar este dato. Cuando introduzcamos el periodo, este recibirá una señal «updatePeriodo» que consiste en que el número correspondiente al introducido se muestre en el escenario.

- **Incremento.** Al igual que con el periodo, el cual se explicó en el mismo apartado, su implementación es idéntica.

- **Botón de comenzar.** Este objeto es quien posee la mayor parte de la funcionalidad del programa. En el momento que pulsemos este objeto, dependiendo del modo en el que estemos, se seleccionará un bloque u otro. Esto se ha implementado gracias a las instrucciones condicionales que proporciona Scratch.

  Pongamos por ejemplo que estamos cifrando un mensaje, el programa a continuación nos pedirá el mensaje, luego el periodo e incremento y finalmente la configuración del disco de Alberti, todo con su comprobación de datos. Esta información se guarda en «mensaje_original», «periodo» e «incr_periodo», respectivamente.

- **Diálogo de resultados.** En la figura 7.5 se muestra el resultado del proceso de cifrado. El fragmento de código lo tenemos en la figura 7.4.

- **Pantalla principal.** En la esquina inferior de la figura 7.1 se encuentra este botón. Se encarga de poder volver al menú principal.

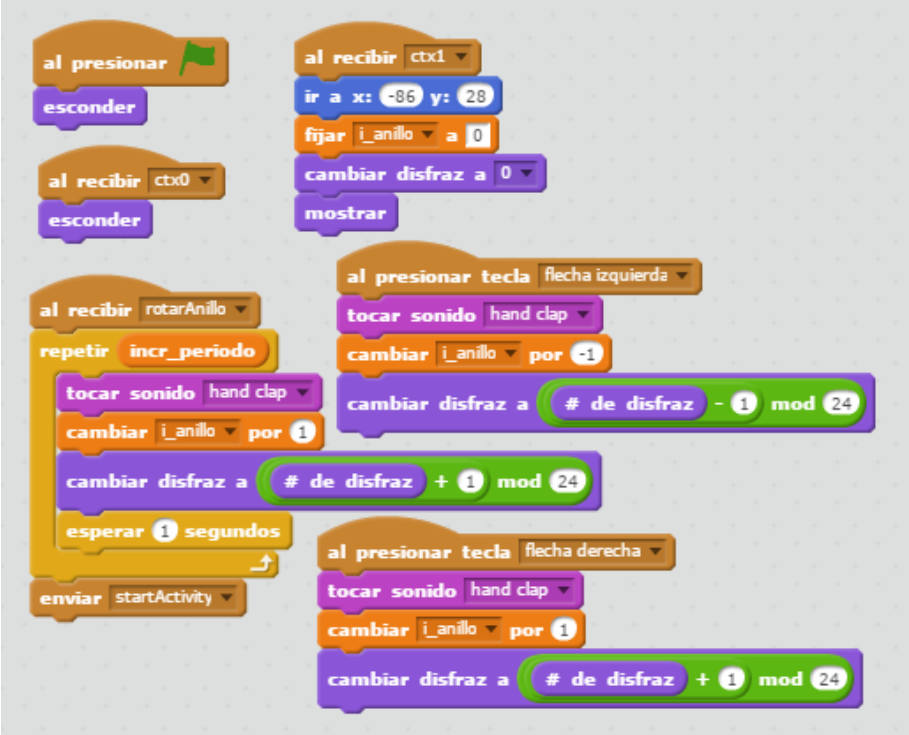

Figura 7.2: Fragmento de código asociado al disco de Alberti. Cuando este objeto recibe la señal «rotarAnillo», realizará un proceso iterativo que rotará el disco un número de veces correspondiente a la variable «incr_periodo». En cada una de estas iteraciones se aumenta en una unidad la variable «i_anillo», que representa en qué estado se encuentran los anillos.

Cuando se pulsa sobre el objeto, este actualiza el contexto en el que estamos a '0', que es el del menú principal enviando la señal correspondiente.

7.3 Posibles ampliaciones

Aunque hemos completado la funcionalidad básica del cifrado, en el apartado 2.4.2 mencionamos que junto al disco de Alberti, existía un libro de códigos (*codebook*) que contenía un sinfín de códigos con un significado en concreto.

A pesar de que el programa permite cifrar y descifrar dígitos numéricos de la misma manera que las cadenas de texto, la idea para una futura mejora sería desarrollar en Scratch un libro de códigos sencillo donde se mostrase una serie de códigos con un significado para que el usuario, una vez descifrara un texto que contenga un código, pudiera consultarlo y hallar el significado. Por ello, el hecho de implementar este libro de códigos podría formar parte como una futura implementación de este cifrado.

La tabla 7.1 muestra cómo podría quedar el resultado en Scratch:

G	Z	E	G	F	H	F
D	O	C	E	3	2	4

Tabla 7.1: Resultado de un criptograma con código.

Un receptor que tuviera un libro de códigos en su mano podría interpretar este criptograma como 'A las 12 horas las naves están listas para zarpar'. En caso de que no tuviera

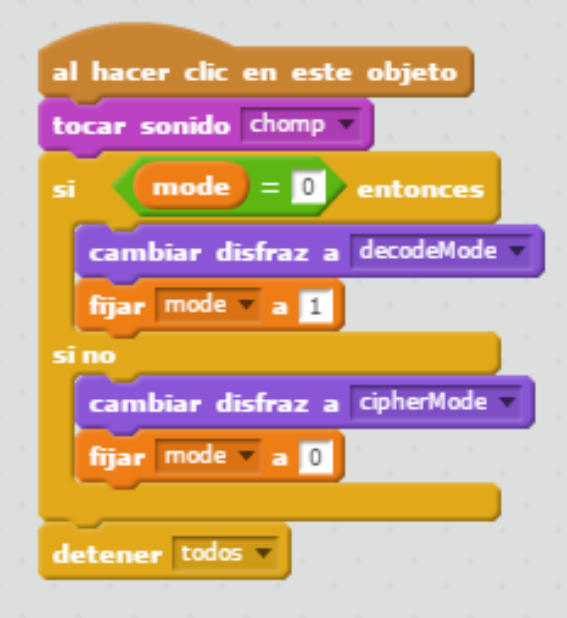

Figura 7.3: Fragmento de código asociado al objeto «*toggle button*». En la figura podemos observar que una vez el usuario pulsa dicho objeto, el programa comprobará el modo en el que estamos y dependiendo de ello, actualiza la variable «mode» a un valor o a otro.

dicho libro, el mensaje se mantendría oculto ya que no se sabe el significado de este aunque hubieran roto el mensaje.

Una característica del cifrado de Alberti es que existen varios métodos a la hora de cifrar un mensaje. En nuestro programa, el periodo es constante y también el incremento, es decir, no varía con el tiempo y el valor que le hemos introducido es el que tiene hasta el final del cifrado.

Existe otro método donde el incremento se produce cada vez que se encuentra un espacio en blanco; por ejemplo, en el texto 'MATAR AL REY' el incremento se realiza cada vez que nos encontramos con un espacio en blanco, es decir, los cinco primeros caracteres se sustituyen de la misma manera y una vez llegamos al espacio en blanco rotamos el disco x posiciones y así sucesivamente.

Dicho esto, la implementación de este método se dejaría para futuras mejoras al igual que el *codebook* mencionado al principio de esta sección.

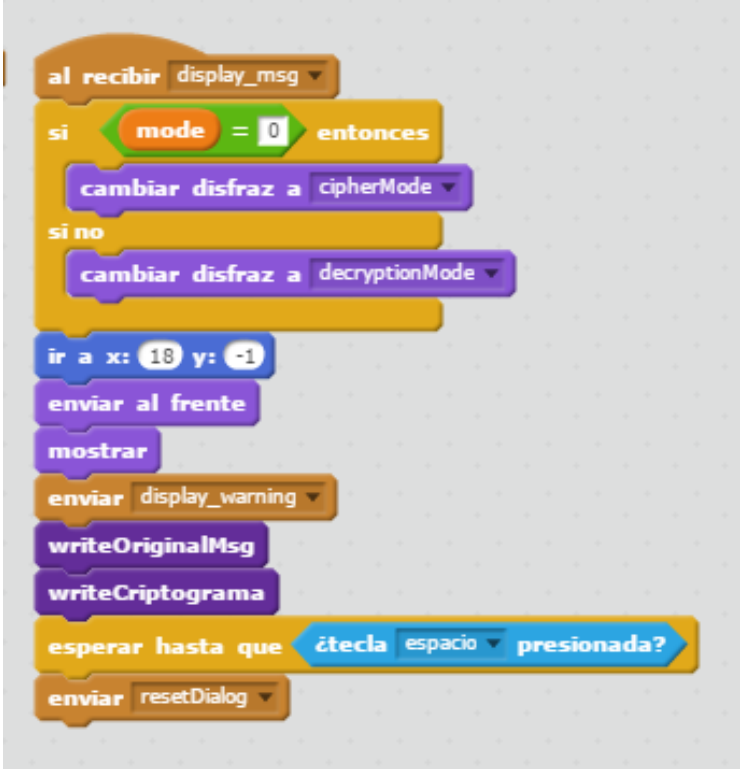

Figura 7.4: Fragmento de código asociado al objeto «Diálogo». Cuando el programa finaliza con el cifrado o descifrado, envía la señal «display_msg» con el fin de notificar al resto de objetos. El objeto «Diálogo» recibe dicha señal y muestra el diálogo que se puede observar en la figura 7.5.

Figura 7.5: Resultado del proceso del cifrado de Alberti. Se puede ver que la letra 'Y' de 'REY' se sustituye por una 'ZZ' debido a la explicación de la tabla 2.4.

CAPÍTULO 8

Diseño e implementación del Ataque por fuerza bruta en Scratch

En este capítulo se hablará de una técnica del criptoanálisis: el ataque por fuerza bruta. Esta técnica consiste en explorar todo el espacio de combinaciones posibles de un criptograma y hallar el mensaje original sin conocer la clave de descifrado. En nuestro programa se ha implementado a nivel de dígitos numéricos simulando romper una contraseña del usuario. A modo de resumen, la funcionalidad que ofrece el programa es que el usuario introduzca una cadena de dígitos de ocho cifras y mostrar la complejidad temporal de esta técnica, que es nuestro objetivo principal. A continuación se presentará cómo está organizado el menú del programa, en qué objetos se componen el programa y finalmente sobre qué posibles mejoras podríamos incluir en versiones futuras de este programa.

8.1 Organización del menú

El primer escenario es el menú (véase la figura 8.1) donde podremos acceder a las distintas funcionalidades que ofrece el programa representando estos objetos mediante botones. Estos son: «Comenzar», «Instrucciones» y «¿Qué es?», que se explicarán a continuación con más detalle.

- **Comenzar.** Al presionar sobre este objeto y tal como se ha venido haciendo hasta ahora en los capítulos anteriores, existe una variables «ctx» que dependiendo de en qué escenario estemos su valor cambiará. Así pues, si se pulsa en «Comenzar» este valor se modificará por un '1' que indica que estamos en el escenario principal donde el usuario de Scratch puede «jugar» con el programa.

- **Instrucciones.** Al pulsar sobre este objeto, la variable «ctx» se actualizará a '2' y accederemos al escenario de instrucciones. En dicho escenario se muestra al usuario las pautas para utilizar esta técnica de criptoanálisis en Scratch.

- **¿Qué es?** En este apartado se explica a los *scratchers* en qué se basa esta técnica y cómo podríamos mejorar la contraseña para que el tiempo de descifrado aumente considerablemente y de esta manera mantener segura nuestra contraseña a manos de los intrusos.

Figura 8.1: Menú principal del ataque mediante fuerza bruta.

8.2 Objetos del programa

En esta sección describiremos qué objetos del programa son los más importantes y cómo se han diseñado e implementado. Adjuntaremos el fragmento de código correspondiente de cada objeto con el fin de que se entienda mejor la implementación.

El programa (véase la figura 8.2) se compone de 28 objetos y entre estos objetos los más característicos y fundamentales para dar funcionalidad son:

- **Campo numérico.** En el escenario existen ocho campos numéricos. Cuando damos comienzo al programa, este objeto posee un método llamado «getNumberClicked» (véase la figura 8.3) cuya función es esperar hasta que el usuario introduzca un valor numérico. En caso de que el usuario quisiese introducir una letra del alfabeto, este se ignoraría ya que nuestro programa solo permite el uso de dígitos numéricos.

  Una vez introduzcamos un valor válido, este enviará una señal al siguiente campo numérico hasta haber introducido la contraseña de tamaño ocho. Otro detalle es que si nos fijamos en la figura 8.3 el objeto pasa por cuatro diferentes disfraces o etapas que son los de la figura 8.4. Para saber más cuándo ocurre cada fase del objeto se explicará a continuación:

  - **Primer disfraz.** Cuando accedemos a la pantalla 8.2 vemos que todos los campos numéricos se encuentran en la primera etapa a excepción del primero que está preparado para que el usuario introduzca el dígito numérico.
  - **Segundo disfraz.** Cada campo numérico estará en esta fase cuando esté esperando al usuario que introduzca un carácter numérico.
  - **Tercer disfraz.** Una vez se haya introducido un número cambiaremos a este tercer disfraz, esperaremos durante 0,2 segundos e inmediatamente pasaremos al cuarto disfraz. Esto se ha implementado con el fin de que el usuario tenga una retroalimentación constante ya que se le notifica cuando ha pulsado un botón.

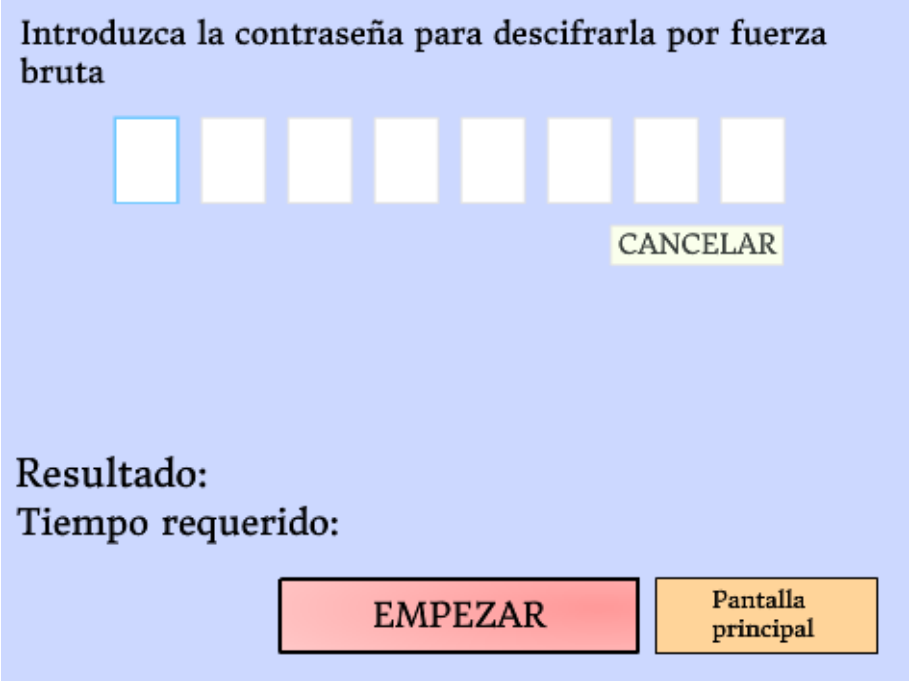

Figura 8.2: Pantalla que se visualiza cuando pulsamos sobre el objeto «Comenzar».

  - **Cuarto disfraz.** Como hemos mencionado líneas más arriba, cuando el usuario introduce un número, este pasa al tercer estado y posteriormente finaliza en este último estado dando como finalizado las diferentes fases del objeto en cuestión.

- **Cancelar.** En el caso de que nos hayamos equivocado introduciendo la contraseña o quisiéramos introducir una diferente, el programa permite la opción de cancelar el proceso actual y volver a comenzar. Este objeto, cuando es pulsado, reinicia las variables y posteriormente envía una señal «deleteClones» y «unlock_numberKeys» dejando el programa en un estado igual al inicial. El código de este objeto se encuentra en la figura 8.5.

- **Resultado.** La implementación de este objeto es sencilla. Una vez que se ha descifrado la contraseña, se crean tanto clones como la longitud de la contraseña y se muestra por pantalla el contenido de la variable «i», que es donde se almacena la contraseña descifrada.

- **Tiempo de ejecución.** La implementación es simular al objeto «Resultado». En el momento que el usuario haya introducido la contraseña correctamente se reinicia el cronómetro (funcionalidad dada por Scratch dentro del bloque de sensores) y una vez encontremos la contraseña introducida por el usuario, se guarda el tiempo requerido en la variables «time». El siguiente paso será crear los clones y mostrarlos por pantalla.

- **Empezar.** Este objeto es el responsable de dar comienzo al descifrado. Una vez se pulsa el botón, este envía una señal «unlock_numberKeys» que se encargará de desbloquear los campos numéricos, es decir, permite al usuario introducir la contraseña, cosa que antes de enviar la señal no estaba permitida. Cuando el usuario haya introducido la contraseña, este recibirá la señal «password_finished» y comenzaremos a realizar el ataque por fuerza bruta.

  En la figura 8.6 se ve que una vez se recibe esta señal se ejecutan los siguientes métodos: «executeDecryption», «writeResults» y «writeTime». El primer método

Figura 8.3: Fragmento de código asociado al objeto «campo numérico». Cuando el campo numérico recibe la señal «unlock_numberKeys», este a lo largo del proceso pasará por todos los disfraces mostrados en la figura 8.4. En nuestro código la transición de cada uno de estos disfraces se representa con la instrucción «cambiar disfraz a [selectedX]», siendo 'X' el número del disfraz. Cabe destacar que en el código existe un método «getNumberClicked» que se encarga de que el usuario tenga que introducir dígitos numéricos obligatoriamente.

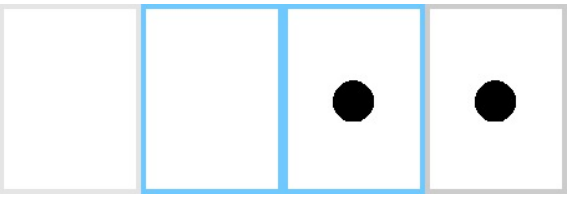

Figura 8.4: Etapas del objeto «campo numérico».

se encarga de realizar una iteración desde '0' hasta '99999999' (máximo valor permitido) que finalizará cuando $flag = 1$. El segundo y tercer método se encarga de mostrar el resultado y el tiempo requerido para romper la contraseña creando los clones que hagan falta para poder representar estos resultado por pantalla.

- **Texto «Procesando...».** Una vez el usuario haya terminado de introducir los valores correspondientes, se crea un clon de este objeto con el único fin de mostrar *feedback* al usuario.

- **Pantalla principal** Este objeto es el encargado de poder volver al menú principal desde cualquiera de los escenarios en el que estemos. Actualizará la variable «ctx» a '0' y enviará a su vez una señal al resto de los objetos para que puedan realizar las operaciones pertinentes.

8.3 Posibles ampliaciones

Una de las características que podríamos incorporar en futuras versiones de este programa sería la opción de poder introducir caracteres del alfabeto español junto a los dígitos numéricos.

Figura 8.5: Fragmento de código asociado al objeto «Cancelar». Al pulsar sobre este objeto, el programa se para y el resto de los objetos vuelven a su estado inicial.

El inconveniente de esto es que si se hubiera introducido esta característica en la primera versión, el tiempo requerido para romper la contraseña crecería exponencialmente y debido a las limitaciones de Scratch el programa tardaría horas e incluso días en finalizar a pesar de que nuestro principal objetivo es enseñar el funcionamiento de esta técnica del criptoanálisis a nuestros usuarios.

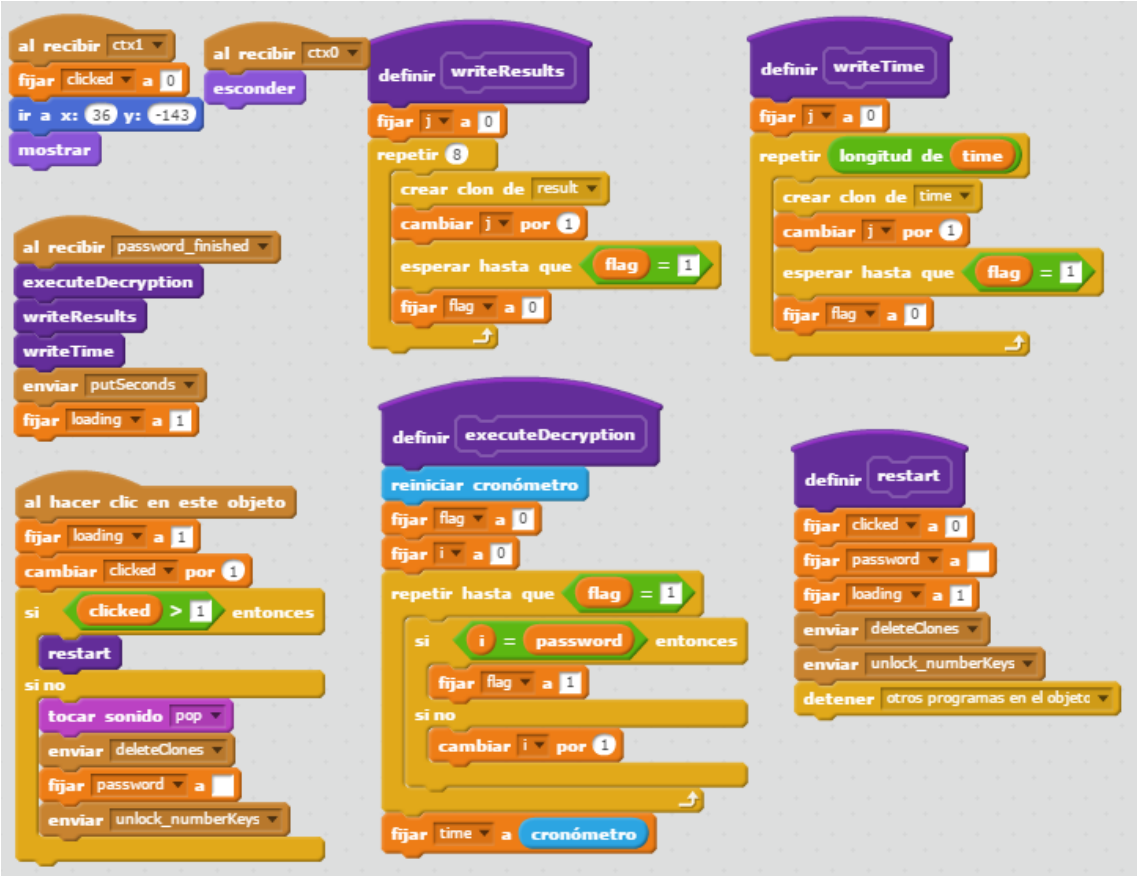

Figura 8.6: Fragmento de código asociado al objeto «Empezar». En el momento en el que introducimos todos los dígitos de la contraseña, el programa envía la señal «password_finished» donde nuestro objeto la recibirá. Así pues, una vez le llega la señal ejecutaremos tres métodos: «executeDecryption», «writeResults» y «writeTime». El primer método se encarga de realizar una iteración desde 1 a 99999999 hasta dar con la contraseña correcta. Durante el proceso habrá un cronómetro contando el tiempo que se tarda en romper la contraseña. Posteriormente, el programa tendrá que mostrar la contraseña del usuario y el tiempo que se ha tardado en hallarse.

CAPÍTULO 9
Diseño e implementación de la página web

El siguiente capítulo se centra en el diseño y la implementación de la página web del Museo de Informática de l'Escola Tècnica Superior d'Enginyeria Informàtica como su propio título indica, con los cinco programas que hemos creado en el entorno de Scratch. Así pues, la página web contendrá una breve descripción de cada uno de los programas, las instrucciones que se deberá seguir en Scratch y el programa mismo junto al enlace que nos permitirá acceder a la página de Scratch en la que podremos ver los comentarios de otros usuarios y la opción de importar el proyecto correspondiente para poder añadir más características.

9.1 Estructura de la página web

Como ya se vio en el primer capítulo, el objetivo de este trabajo es la creación de cuatro de los principales cifrados clásicos de la historia mediante el lenguaje de programación Scratch. Además de estos cifrados se ha creado un sencillo programa donde se muestra la utilización de un ataque por fuerza bruta, muy usado actualmente en el mundo del criptoanálisis. Una vez finalizados se han incluido en la web del Museo de Informática de l'Escola Tècnica Superior d'Enginyeria Informàtica con el fin de que las personas que visiten el museo puedan disfrutar conociendo una parte de la historia de la criptografía y de esta manera animar a los visitantes a dar sus primeros pasos en la programación de la mano de Scratch.

Si nos fijamos en la figura 9.1, podemos observar cómo se estructura cada apartado en la página web. Estos apartados vendrán dados por el cifrado y hemos optado por incluir una descripción de cada uno de ellos, donde se contará brevemente la historia de este para dar paso a la descripción de las reglas y de su funcionamiento. Posteriormente nos encontraremos con el proyecto de Scratch embebido (véase la figura 9.2) en nuestra página web donde cualquier usuario puede probar su funcionamiento. En la parte inferior de cada uno de estos cifrados se encuentra un enlace que nos llevará directamente a la página web del proyecto de Scratch, en la cual cualquier usuario puede comentar su opinión acerca del programa así como acceder al código interno del mismo para importar este proyecto para cambiar alguna funcionalidad o incluso mejorarlo.

Finalmente en la parte inferior de la página web, en el apartado de «Notas» se encuentra el nombre del creador de los cifradoa y del tutor.

Figura 9.1: Página web implementada para el Museo de Informática. En la figura podemos ver la estructura que tiene empezando por el título del trabajo, seguido del autor de la página y una pequeña introducción de cada uno de los cifrados implementados en Scratch con el programa embebido en la página. Esto último se puede ver en la figura 9.2.

9.2 Tecnologías empleadas

La página web ha sido creado en Webstorm, un entorno de desarrollo creado por la empresa *JetBrains* para en lenguaje Javascript. Esta empresa además de Webstorm posee un sinfín de programas para diferentes lenguajes de programación tales como Java, Android, Python, PHP, etc.

A continuación describiremos las tecnologías que hemos hecho uso para implementar nuestra página web: HTML, CSS y Javascript. Cada uno de estos lenguajes se ha empleado para una funcionalidad concreta dentro del proceso de elaboración. A continuación explicaremos para qué sirve cada una de estas tecnologías:

- **HTML.** De la abreviatura de *HyperText Markup Language*, en castellano lenguaje de marcas de hipertexto. HTML está compuesto por una serie de etiquetas que el navegador (Google Chrome, Mozilla Firefox, Safari...) interpreta en el momento que accedamos a la página web y le da forma al contenido para que el usuario pueda visualizarlo correctamente.

  Dentro de la creación de una página, podríamos catalogar HTML como la tecnología empleada para dar estructura y contenido.

Figura 9.2: Página web donde se muestra el programa implementado en Scratch embebido en la propia página.

- **CSS.** Se trata de una hoja de estilos que define el aspecto de la página web. Por ejemplo, CSS abarca cuestiones como la fuente, los colores, márgenes, líneas entre otras funcionalidades. A pesar de que HTML también ofrece esta funcionalidad, CSS lo realiza de manera más sofisticada y precisa. Actualmente está soportado por todos los navegadores.

  Por otra parte, este lenguaje no aporta ninguna funcionalidad extra a la página web a parte de darle una estética más vistosa y particular a cada página web.

- **JavaScript.** Es un lenguaje de programación que permite a los programadores crear acciones en sus páginas webs y de esta manera añadir cierto dinamismo a las páginas. Este lenguaje no requiere de compiladores como muchos otros lenguajes conocidos debido a que funciona en el lado del cliente, es decir, los propios navegadores son los que interpretan el código JavaScript.

  Cabe destacar que existen dos tipos de JavaScript, el primero es el que se ejecuta en el lado del cliente y el otro es el que se ejecuta en el lado del servidor.

De este modo, se ha podido realizar la página web que podemos observar en la figura 9.1 y que se incluirá en el Museo de Informática para que cualquier usuario pueda acceder a los programas realizados en el entorno de programación Scratch.

CAPÍTULO 10
Conclusiones

Último capítulo de la memoria, en él se plantean las consideraciones finales que hemos obtenido a lo largo del trabajo realizado. Se mostrará un resumen final y además se analizarán los objetivos que hemos alcanzado en el presente proyecto. Asimismo se incluirá al final del capítulo un apartado de trabajo futuro.

10.1 Consideraciones finales

En el proyecto actual se ha demostrado que con un lenguaje de programación orientado al público en general como Scratch, es posible la creación de cifrados de menor o mayor complejidad con una cierta facilidad a pesar de que en su época histórica supusieron una revolución a la hora de encriptar mensajes para evitar que esa información caiga en manos equivocadas. Para la creación de estos cifrados se ha tenido que realizar un trabajo de investigación y se han utilizado numerosos artículos que se pueden encontrar en la bibliografía al final del documento.

Tal como se explicó en el capítulo 3 de introducción a Scratch, este introduce a los usuarios noveles de cualquier edad en el mundo de la programación. Para llegar a conseguir este objetivo, Scratch ofrece un lenguaje intuitivo basado en bloques similares a *LEGO* que no requiere escribir líneas de código, sino que el usuario simplemente tiene que arrastrar estos bloques dando lugar a un conjunto de bloques organizados entre sí y de esta manera generar un fragmento de código totalmente funcional. Cada uno de estos fragmentos que el usuario crea pertenece a un objeto dentro del entorno de Scratch. Estos objetos creados interactúan entre ellos y de esta forma una vez finalizamos el proyecto tendríamos la funcionalidad final.

Si bien facilita la programación respecto a otros lenguajes de programación actuales de más alto nivel, Scratch resulta un poco limitado y para la creación de ciertos tipos de programas no es la mejor elección. Por ejemplo, en nuestro caso en particular, a la hora de realizar los cifrados hemos tenido que limitar la longitud de nuestros mensajes ya que en el caso de que quisiéramos encriptar un documento extenso haría falta una *scrollview*[1] que Scratch no posee.

Debido a esto, lo primero que hemos realizado en el trabajo ha sido introducir la historia de los cifrados a nuestros usuarios para que entiendan el impacto que han tenido a lo largo de la historia de la criptografía y de esta manera aprender las bases del funcionamiento de cada uno de ellos. Posteriormente se ha explicado la razón por la que hemos escogido Scratch como lenguaje de programación para nuestros programas y qué

[1] Funcionalidad que posee la mayor parte de lenguajes de alto nivel que consiste en poder desplazar el documento de manera vertical u horizontal de tal forma que el usuario pueda visualizar el contenido.

funcionalidad nos ofrece para la elaboración de estos programas, al mismo tiempo que el usuario aprende y se divierte creándolos en Scratch.

Una vez hemos introducido el entorno de Scratch, el siguiente paso ha sido explicar cómo hemos implementado cada uno de los objetos más importantes de cada cifrado para darle la funcionalidad que requiere en los programas. Para ello, se ha explicado uno a uno estos objetos con su respectivo fragmento de código. Además, se ha incluido en cada cifrado un apartado de posibles ampliaciones que podrían llevarse a cabo en caso de que alguien decidiera escoger el presente trabajo para ampliarlo.

Finalmente, se describe la página web que hemos creado para el Museo de Informática de l'Escola Tècnica Superior d'Enginyeria Informàtica en la que se podrá consultar cada uno de los cifrados que hemos elaborado para el presente trabajo, con una pequeña introducción a la historia y las instrucciones necesarias para utilizar el programa creado. Por otra parte, se podrá acceder al código interno de los proyectos para que cualquier usuario pueda continuar añadiendo nuevas funcionalidades. Se espera que el trabajo anime a los usuarios noveles a aprender a programar de manera divertida y didáctica.

10.2 Objetivos cumplidos

Una vez llegado a los últimos puntos del documento, podemos afirmar que hemos cumplido los objetivos propuestos que hemos mencionado en el capítulo 1. Estos objetivos son la realización mediante Scratch de los siguientes programas:

- Escítala espartana
- Tablero de Polibio
- Cifrado de César
- Disco de Alberti
- Ataque por fuerza bruta

Para cada uno de estos programas se ha descrito un capítulo entero donde se habla del proceso de implementación junto a la explicación, con el único fin de que el lector pueda seguir en cada momento el capítulo sin ningún tipo de problemas.

10.3 Trabajo futuro

Para finalizar el documento, cabe mencionar que en cada uno de los capítulos destinados a explicar el desarrollo de los cifrados (4, 5, 6 y 7) se ha incluido una sección de posibles ampliaciones, en el que se describen algunas de las mejoras que se podría añadir a los programas y así darle un mayor número de funciones. Además de las posibles ampliaciones, en la actualidad existen diversos cifrados que cualquier usuario con un mínimo conocimiento de criptografía y de Scratch puede implementarlo en dicho entorno. Estos podrían ser el cifrado afín y la sustitución simple.

10.3.1. El cifrado afín

El cifrado afín data del año 1967 y se trata de un cifrado monoalfabético de sustitución. Este cifrado sería interesante de implementar en Scratch ya que es un cifrado asimétrico, es decir, la clave para cifrar y descifrar es distinta. Los parámetros que se requieren

son tres: a, k y la n. Al igual que en el cifrado de César cada carácter del alfabeto español viene asociado con un índice que va desde el '0' al '26' ('A' a la 'Z') y la función de cifrado viene dada por la siguiente fórmula:

$$C_i = (a * M_i + k) \ mod \ n$$

Donde:

- C_i = criptograma generado
- a = constante
- M_i = índice del carácter
- K = clave
- n = longitud del alfabeto

Por otra parte, la función de descifrado es la siguiente donde la única peculiaridad es el inverso multiplicativo de a:

$$M_i = (C_i - k)a^{-1} \ mod \ n$$

10.3.2. La sustitución simple

El cifrado por sustitución simple es un cifrado monoalfabético donde cada carácter del alfabeto se sustituye siempre por el mismo carácter a lo largo de todo el proceso de cifrado. La clave para cifrar el mensaje es una cadena de caracteres que será la encargada de establecer la relación sobre qué carácter se sustituye. Por ejemplo, si tenemos la clave '*CLAVE*' la relación quedaría tal como se muestra en la tabla 10.1:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P
C	L	A	V	E	B	D	F	G	H	I	J	K	M	N	Ñ	O

Q	R	S	T	U	V	W	X	Y	Z
P	Q	R	S	T	U	W	X	Y	Z

Tabla 10.1: Tabla resultado al introducir la clave *'CLAVE'*

Si nos fijamos en la tabla 10.1, lo primero que se escribe es la clave y posteriormente rellenaremos los huecos con los caracteres del alfabeto sin repetir los ya usados en la clave del cifrado. Una vez tenemos la tabla bastará con sustituir los caracteres para cifrar; así pues, el mensaje *'HOLA MUNDO'* se cifrará por *'FÑJC KTMVÑ'*.

Presentados estos cifrados, sería bastante interesante implementar estos cifrados en Scratch con el fin de tener una mayor cantidad de programas relacionados con la criptografía clásica y de esta manera aumentar la biblioteca de proyectos para que los visitantes del Museo de Informática de l'Escola Tècnica Superior d'Enginyeria Informàtica tenga un abanico más amplio de métodos criptográficos donde poder elegir.

Bibliografía

[1] Alberti wheel. Consultar en http://www.dcode.fr/alberti-cipher.

[2] Anónimo La escítala, Febrero, 2017. Consultar en http://ojoscuriosos.com/la-escitala/

[3] Bibliografía de León Battista Alberti. Consultar en https://www.saylor.org/site/wp-content/uploads/2011/06/Leon-Battista-Alberti.pdf.

[4] Champion, Craige. "Polybius and the Aetolians: A Historigraphical Approach."forthcoming in Blackwell's Companion to Greek and Roman Historiography, ed. J. Marincola. Oxford: Basil Blackwell

[5] Cifrados *Vigenère*. Consultar en http://ciphermachines.com/vigenere.

[6] Cking. (2011). "Unbreakable" Codes Throughout History: The Polybius Square and the Caesar Shift. July.

[7] De Cifris. *A Treatise on Ciphers* (1467), trans. A. Zaccagnini. Foreword by David Kahn, Galimberti, Torino 1997

[8] De Pictura, 1435. On Painting, in English, De Pictura, in Latin, On Painting. Penguin Classics. 1972. ISBN 9780140433319

[9] *De re aedificatoria*. (1452, Ten Books on Architecture)

[10] Discuss Scratch. Consultar en https://scratch.mit.edu/discuss/.

[11] DSIC-UPV (2014) Criptografía Clásica. Seminario de la asignatura de Criptografía del Grado en Ingeniería Informática, Valencia.

[12] El cifrado de César y otros cifrados monoalfabéticos. Consultar en http://www.dma.fi.upm.es/recursos/aplicaciones/matematica_discreta/web/aritmetica_modular/cesar.html.

[13] García, Juan. Qué es un ataque por fuerza bruta. *Conceptos y seguridad*, 7, Mayo, 2013. http://faqoff.es/que-es-un-ataque-por-fuerza-bruta/

[14] Gutiérrez, Pedro. ¿Qué es y como surge la criptografía?: un repaso por su historia. https://www.genbetadev.com/seguridad-informatica/que-es-y-como-surge-la-criptografia-un-repaso-por-su-historia

[15] Historia del tablero de Polibio. Consultar en http://www.cryptool-online.org/index.php?option=com_content&view=article&id=70&Itemid=80&lang=en.

[16] Historia y vida de Cayo Julio César. Consultar en https://www.biografiasyvidas.com/biografia/c/cesar.htm.

[17] Historia y vida de *León Battista Alberti*. Consultar en https://www.biografiasyvidas.com/biografia/a/alberti_leon.htm.

[18] J. M. Wing (2006). Computational Thinking. Viewpoint. Communications of the ACM, 49(3):33-35.

[19] Kahn, David. (1967). The codebreakers: the story of secret writing. New York: MacMillan.

[20] MaManus, S. (2013). Scratch programming in Easy Steps. United Kingdom: East Steps Limited.

[21] Marqués Moros, Miguel. (2016). *Diseño e implementación de videojuegos clásicos con Scratch*. Trabajo Fin de Grado. ETSINF. UPV.

[22] Programa interactivo de la *Escítala espartana*. Consultar en http://nosolomates.es/?page_id=762.

[23] Programa interactivo del tablero de Polibio. Consultar en http://www.dcode.fr/polybius-cipher.

[24] Prudencio, M. (2013). Una herramienta lúdica de iniciación a la programación, Scratch. Linux Magazine, 28:78-82.

[25] Resnick, Mitchel. et al. (2009). Scratch: Programming for all. Communications of the ACM, 52(11):60-67.

[26] Resnick, Mitchel (2013). Lifelong Kindergarten. Cultures of Creativity, LEGO Foundation. http://web.media.mit.edu/ mres/papers/ CulturesCreativityEssay.pdf

[27] Scratch Paint Editor. Consultar en https://wiki.scratch.mit.edu/wiki/Paint_Editor.

[28] Scratch - Imagine, Program, Share. Consultar en https://scratch.mit.edu/.

[29] Serrano, G. (2015). Programación Scratch para niños. FranciaL Amazon Me- dia EI S.à.r.l.

[30] The history of the *Cipher wheel*. Consultar en http://www.cypherwheel.com/history.html

[31] Villaescusa Vinader, Samuel. (2015). *Diseño de videojuegos clásicos con Scratch*. Trabajo Fin de Grado. ETSINF. UPV.