UNIVERSIDAD
REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2011/2012

Proyecto de Fin de Carrera

Applet para la resolución de un Cubo de Rubik

Autor: Alberto Bueno Gonell

Tutores: Celeste Pizarro Romero

Clara Simón de Blas

Resumen del proyecto:

Nombre del proyecto: Applet de resolución del cubo de Rubik
Autor: Alberto Bueno Gonell
Fecha de entrega: 2012

En este proyecto se desarrollará un applet de Java que permita al usuario obtener la solución de un cubo de Rubik dada una configuración válida de éste.
El usuario podrá elegir la configuración bien mediante el intercambio de piezas o con el movimiento de las distintas caras del cubo.
Una vez dada la solución el usuario podrá ver paso a paso cuales son los movimientos para la correcta resolución del cubo.

En la elaboración del proyecto se pueden distinguir dos partes:

- Implementación del algoritmo de resolución del cubo
- Diseño de la interfaz gráfica en la que el usuario puede insertar una configuración del cubo y resolverlo o jugar con él.

ÍNDICE

1. INTRODUCCIÓN

1.1. Cubo de Rubik:

El cubo de Rubik es un rompecabezas mecánico inventado por el escultor y profesor de arquitectura húngaro Ernö Rubik en 1974.Se trata de un conocido rompecabezas cuyas caras están divididas en cuadrados de un mismo color que se pueden cambiar de posición. El objetivo se consigue al colocar todos los cuadrados de cada cara del cubo con el mismo color.

Se estima que se han vendido más de 350 millones de cubos de Rubik o imitaciones en todo el mundo. Su sencillo mecanismo sorprende tanto desde el punto de vista mecánico, al estudiar su interior, como por la complejidad de las combinaciones que se consiguen al girar sus caras, 43 252 003 274 489 856 000.

En el cubo típico, cada cara está cubierta por nueve cuadrados de un color sólido. Cuando está resuelto, cada cara es de un mismo color. Sin embargo, el rompecabezas se encuentra, principalmente, en cuatro versiones: el 2x2x2 "Cubo de bolsillo", el 3x3x3 el cubo de Rubik estándar, el 4x4x4 (La venganza de Rubik) y el 5x5x5 (Cubo del Profesor).

El cubo de Rubik consta de 26 piezas de tres tipos:

- 8 vértices
- 12 aristas
- 6 centros

Los centros van encajados a una pieza central de tres ejes, que es la encargada de que todas las caras puedan girar. Los vértices y aristas van encajados en la estructura que forma el eje central junto con los centros.

A su vez, el cubo se puede dividir en 6 tipos de caras, y que cada una podrá ser de cualquier tipo dependiendo de cómo coloquemos el cubo.
Estas caras serán: Frontal, Trasera, Izquierda, Derecha, Superior e Inferior. No tendrán una posición estricta ya que dependerán de nuestra perspectiva hacia ellas.

Para la resolución del cubo se emplea una notación para referirse a cada cara.
Cada cara tendrá dos tipos de movimiento, en el sentido de las agujas del reloj y el sentido contrario a las agujas. De este modo, para girar una cara del cubo, habrá que ponerla de frente al usuario y la girarla según sea convenga, devolviéndola posteriormente a su posición.

1.2. Applet:

Un Applet es un componente software, que suele ser pequeño, escrito en un lenguaje de programación, que se ejecuta bajo el control de una aplicación más grande que lo contiene, un navegador web.

Un applet Java es un applet escrito en el lenguaje de programación Java. Suelen funcionar en un navegador web utilizando la Java Virtual Machine (JVM), o en el AppletViewer de Sun.

Entre sus características podemos mencionar un esquema de seguridad que permite que los applets que se ejecutan en el equipo no tengan acceso a partes sensibles (por ej. no pueden escribir archivos), a menos que uno mismo le dé los permisos necesarios en el sistema; la desventaja de este enfoque es que la entrega de permisos es engorrosa para el usuario común, lo cual juega en contra de uno de los objetivos de los Java applets: proporcionar una forma fácil de ejecutar aplicaciones desde el navegador web.
En Java un applet (Subprograma), es un programa que puede incrustarse en un documento HTML; es decir en una página Web. Cuando un Navegador carga una página Web que contiene un Applet, éste se descarga en el navegador Web y comienza a ejecutarse. Esto nos permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página Web en su navegador.

El Navegador que carga y ejecuta el applet se conoce en términos genéricos como el contenedor de Applets. El kit de desarrollo de Software para java 2 (J2SDK) 1.4.1 incluye el contenedor de Applets, llamado appletviewer, para probar los applets antes de incrustarlos en una página Web.

Ventajas:

- Son multiplataforma (funcionan en Linux, Windows, Mac OS, y en cualquier sistema operativo para el cual exista una JVM).
- El mismo applet puede trabajar en "todas" las versiones de Java, y no sólo la última versión del plug-in. Sin embargo, si un applet requiere una versión posterior de la JRE, el cliente se verá obligado a esperar durante la descarga de la nueva JRE.
- Es soportado por la mayoría de los navegadores Web
- Puede ser almacenado en la memoria cache de la mayoría de los navegadores web, de modo que se cargará rápidamente cuando se vuelva a cargar la página web, aunque puede quedar atascado en la caché, causando problemas cuando se liberan nuevas versiones.
- Puede tener acceso completo a la máquina en la que se está ejecutando, si el usuario lo permite.
- Puede ejecutarse con velocidades comparables (pero en general más lento) a la de otros lenguajes compilados, como C + +, pero muchas veces más rápida que la de JavaScript.
- Puede trasladar el trabajo del servidor al cliente, haciendo una solución Web más escalable tomando en cuenta el número de usuarios / clientes.

Desventajas:

- Requiere el plug-in de Java, que no está disponible por defecto en todos los navegadores web.
- Sun no ha creado una implementación del plug-in para los procesadores de 64 bits.
- No puede iniciar la ejecución hasta que la JVM esté en funcionamiento, y esto puede tomar tiempo la primera vez que se ejecuta un applet.

- Si no está firmado como confiable, tiene un acceso limitado al sistema del usuario - en particular no tiene acceso directo al disco duro del cliente o al portapapeles.
- Algunas organizaciones sólo permiten la instalación de software a los administradores. Como resultado, muchos usuarios (sin privilegios para instalar el plug-in en su navegador) no pueden ver los applets.

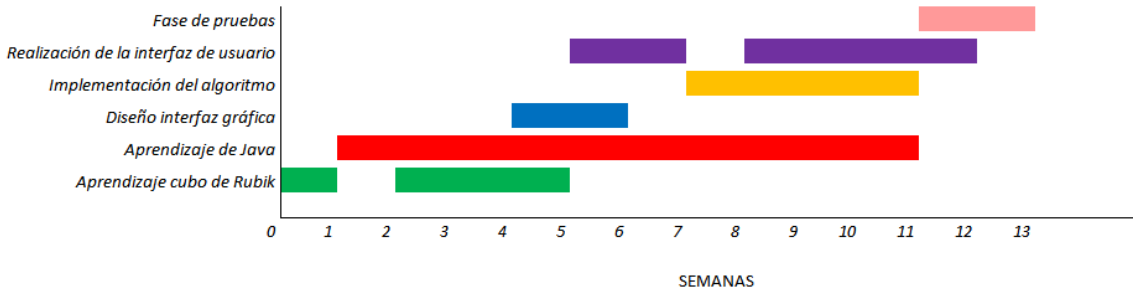
2. OBJETIVOS

El objetivo del proyecto es el desarrollo de un applet que mediante la representación gráfica de un cubo de Rubik, en el cual se pueda elegir su configuración mediante la interfaz gráfica, éste sea capaz de resolverlo indicando los pasos a seguir.

2.1. Método de trabajo

Al empezar a trabajar en el proyecto, debido a mi desconocimiento de cómo se solucionaba un cubo de Rubik tuve que profundizar en las características del cubo las distintas formas de solucionarlo que hay.

1. Adquisición de conocimientos necesarios para conocer diferentes algoritmos de resolución del cubo de Rubik.
2. Selección de herramientas y lenguaje de programación: Utilización del lenguaje Java en el entorno de programación EclipseGavab 2.0.
3. Instalación de las herramientas.
4. Adquisición de conocimientos de Java, tanto de estructuras simples como de creación de interfaces gráficas.
5. Diseño de la interfaz gráfica de la aplicación.
6. Implementación del algoritmo que es capaz de resolver el cubo de Rubik
7. Realización de la interfaz de usuario.
8. Fase de pruebas.



En este gráfico se representa de una forma aproximada el tiempo dedicado a cada fase del proyecto.

2.2. Algoritmos de resolución:

Para la resolución del cubo se han estudiado dos algoritmos distintos:

El primer algoritmo estudiado ha sido el llamado de Principiantes, puesto que lo pasos que se siguen son siempre los mismos, independientemente de la configuración del cubo, en el que se siguen los siguientes pasos:

1. Colocación de las aristas superiores: Este paso consiste en la creación de una cruz en la cara Superior, de modo que estas aristas queden alineadas con sus respectivos centros.
2. Colocación de los vértices en la capa superior: Habrá que colocar los 4 vértices en su posición, de este modo habremos resuelto una cara entera y tendremos las caras frontal, izquierda y derecha por igual, formando una especie de T.
3. Resolución de la capa central: Hay que colocar todas las aristas de las caras adyacentes en su sitio.
4. Hacer una cruz en la cara inferior: En este paso no es necesario que las aristas de la cara inferior estén alineadas con sus centros (paso siguiente).
5. Colocación de las aristas en la cara inferior: La cruz formada en el apartado anterior debe coincidir con el color de los centros de las caras laterales.
6. Colocación de las esquinas en la capa inferior: Ya sólo nos quedan 4 vértices por colocar. Habrá que poner cada esquina en su sitio, aunque esté girada.
7. Orientación de las esquinas en la capa inferior: Una vez estén colocadas en su sitio, habrá que voltearlas para que queden en su sitio.

Después de estos 7 pasos ya habremos conseguido resolver el cubo de Rubik.

El segundo algoritmo estudiado se resuelve por el método de Kociemba. El método de Kociemba es una mejora del algoritmo de Thistlethwaite, un algoritmo que resolvía el problema en pocos movimientos, basándose en la teoría de grupos para los diferentes movimientos, pudo agruparlos en cuatro.

Kociemba logró disminuir el número de grupos únicamente a dos. También empleó el método de búsqueda equivalente a IDA*, con lo que se mejoró el proceso de búsqueda y se obtuvieron mejores resultados.

Es un algoritmo de búsqueda por medio de nodos, utilizando una gran tabla con los posibles movimientos partiendo desde un estado aleatorio, empleando la búsqueda IDA* eligiendo de esta manera los mejores caminos para llegar a la solución.

Búsqueda IDA*: Este método consiste en un algoritmo de profundidad iterativa en el que se hace uso de la información heurística de que se dispone sobre el problema para decidir que nodo expandir a continuación, y hasta dónde llegar en cada una de las iteraciones del proceso.

Motivación de elección del algoritmo: El proyecto ha sido desarrollado con el método de Kociemba, ya que una vez comprendidas las tablas de los posibles movimientos resultaba más sencilla su implementación, a la vez que el resultado del algoritmo es un número de pasos mucho menor.

3. Descripción informática

3.1. Medios Hardware y Software:

Medios Hardware:

El desarrollo del proyecto se ha realizado en un ordenador portátil marca Acer, con procesador Intel Core i3-370M a 2,40 GHz y 4GB de RAM.

Medios Software:

Para la creación del Applet se ha usado tanto el sistema operativo Windows 7 Professional como Linux (Ubuntu).

El editor de Java utilizado ha sido el Eclipse Gavab 2.0.

El navegador utilizado ha sido el Google Chrome 19.0.1084.56.

Se han realizado pruebas en distintos navegadores:

- Google Chrome 19.0.1084.56.
- Internet Explorer 8

3.2. Lenguaje de programación:

El lenguaje elegido para el desarrollo del proyecto ha sido Java por varios motivos como la compatibilidad con todos los navegadores y sistemas operativos o el desconocimiento de los demás lenguajes con los que crear un Applet.

Java es un lenguaje de programación con el que podemos realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía Sun Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

Una de las principales características por las que Java se ha hecho muy famoso es que es un lenguaje independiente de la plataforma. Eso quiere decir que si hacemos un programa en Java podrá funcionar en cualquier ordenador del mercado. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc.

Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente.
La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

Java fue pensado originalmente para utilizarse en cualquier tipo de electrodoméstico pero la idea fracasó. Uno de los fundadores de Sun rescató la idea para utilizarla en el ámbito de Internet y convirtieron a Java en un lenguaje potente, seguro y universal gracias a que lo puede utilizar todo el mundo y es gratuito. Una de los primeros triunfos de Java fue que se integró en el navegador Netscape y permitía ejecutar programas dentro de una página web, hasta entonces impensable con el HTML.

Actualmente Java se utiliza en un amplio abanico de posibilidades y casi cualquier cosa que se puede hacer en cualquier lenguaje se puede hacer también en Java y muchas veces con grandes ventajas. Para lo que nos interesa a nosotros, con Java podemos programar páginas web dinámicas, con accesos a bases de datos, utilizando XML, con cualquier tipo de conexión de red entre cualquier sistema. En general, cualquier aplicación que deseemos hacer con acceso a través web se puede hacer utilizando Java.

3.3. Especificación de requisitos

- *Editar:* Permite entrar al modo Editar del programa. En él se podrá cambiar la configuración del cubo arrastrando la pieza que se quiere mover al lugar deseado.
- *Jugar:* Permite entrar al modo Jugar del programa. En él se podrán hacer los movimientos normales de las piezas del cubo.
- *Resetear:* El cubo vuelve a su configuración inicial.
- *Solución*: Aplica el algoritmo de resolución a la configuración actual del cubo, mostrando por pantalla los pasos a seguir.
- *SiguienteMovimiento:* Permite, una vez exista una solución, ver el movimiento del cubo respecto al paso que tenía que hacer.
- *Ayuda:* Permite visualizar al usuario la ayuda de las distintas funciones del applet.

3.4. Descripción del algoritmo de resolución:

El algoritmo de Kociemba se basa en la generación de distintas etapas que factibles para resolver un cubo que está desordenado. Con una serie de movimientos se va llegando a submetas, las cuales se van guardando en dos vectores de manera consecutiva y dependiendo de la fase a la que pertenezcan, fase1 o fase2.

Cuando empiezan los movimientos se calculan las tablas transitorias, que servirán para seguir el camino más óptimo.

Una vez calculadas las tablas se usa el método de búsqueda IDA*, que buscará todas las posibles soluciones de cada etapa. Si no lo encuentra en la primera etapa pasará a buscarlo en la segunda. Estas soluciones son guardadas en los vectores anteriormente nombrados, bien de la fase1 o de la fase2. Una vez terminado el algoritmo deberá elegir el mejor camino para resolver el cubo.

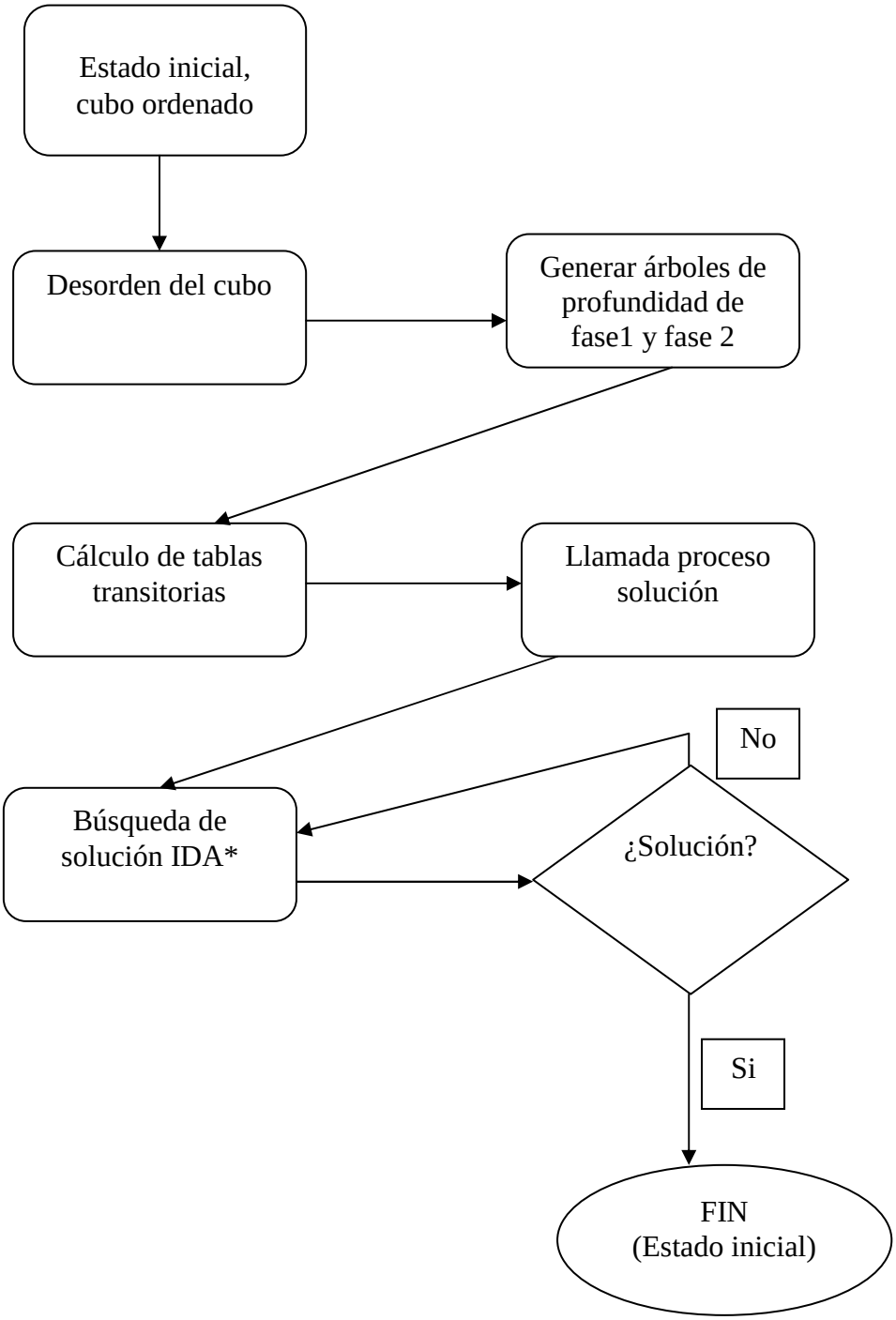
La búsqueda se realiza tratando de encontrar el camino más adecuado empleando la heurística de la base de datos de patrones. Mientras se va realizando la búsqueda va comparando si ya llegó al estado final, en caso contrario, continúa la búsqueda. Ésta acaba cuando el estado final es igual al estado definido al principio, es decir, cuando el cubo ya ha sido resuelto.

3.5. Algoritmo de resolución:

```
MaxLongitud=9999

    function Kociemba ( posición p)

        for profundidad d desde 0 to maxLongitud

            BusquedaFase1( p; d )

        endfor

    endfunction

    function BusquedaFase1 ( posicion p; profundidad d )

        if d=0 then

            if submeta alcanzada y ultimo movimiento fue A, B, A' o B' then

                IniciaFase2( p )

            endif

            elseif d > 0 then

                if prune1[p]< = d then

                    for each movimiento de m disponible

                      BusquedaFase1( resultado de aplicar m a p; d-1 )

                    endfor

                endif

            endif

        endif

    endfunction
```

```
function IniciaFase2 ( posicion p)
        for profundidad d from 0 to maxLongitud - profundidadActual
                BusquedaFase2( p; d )
        endfor
endfunction

function BusquedaFase2( posicion p; profundidad d )
        if d=0 then
                if solucionado then
                        Encontró una solución!
                        maxLongitud = profundidadActual-1
                endif
               elseif d>0 then
                        if prune2[p]< =d then
                                for each movimiento de m disponible
                                  BusquedaFase2( resultado de aplicar m a p; d-1)
                                endfor
                        endif
                endif
        endif
endfunction
```

3.6. Diagrama de flujo del algoritmo:



4. IMPLEMENTACIÓN DEL PROYECTO

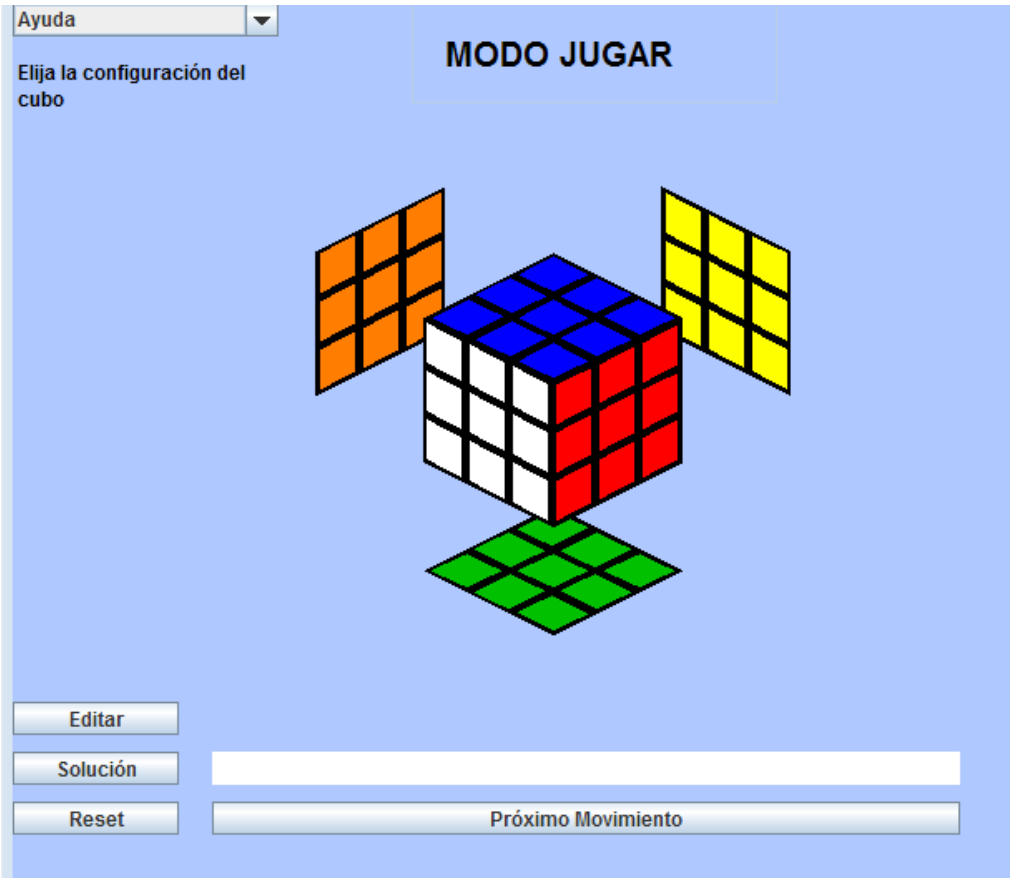
4.1. Construcción del applet:

Para la construcción de la interfaz gráfica se han utilizado las distintas clases que ofrece el lenguaje Java, clases como JButton, JTextField o JPanel entre otras muchas.

A la hora de diseñarlo se han tenido en cuenta los siguientes criterios:

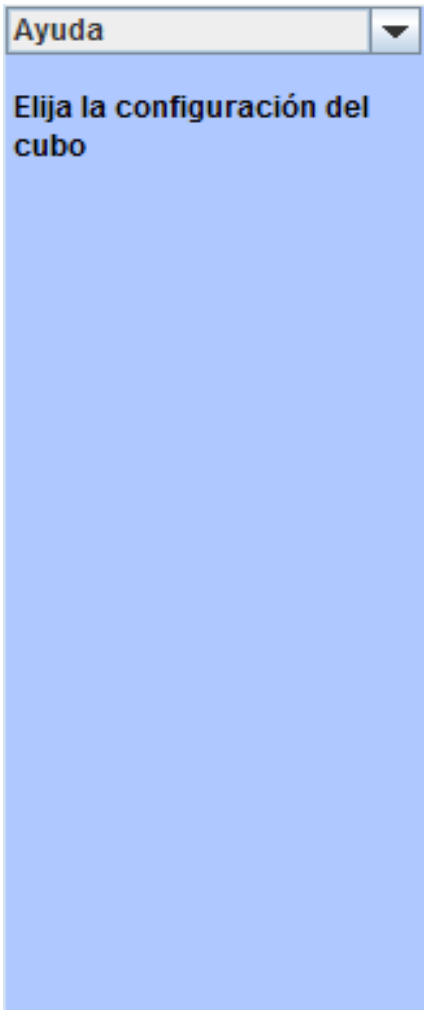
- Facilidad e intuitividad de ser usada.
- Mostrar la solución claramente.
- Ser lo más explicito posible en los casos de ayuda para las notaciones.
- Ofrecer todas las funciones especificadas anteriormente.

Aquí una visión general del Applet:

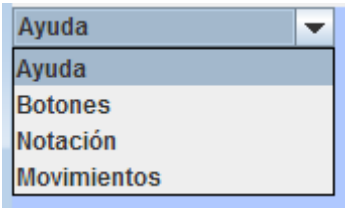


En la interfaz gráfica se distinguen varias partes:

Panel de ayuda:

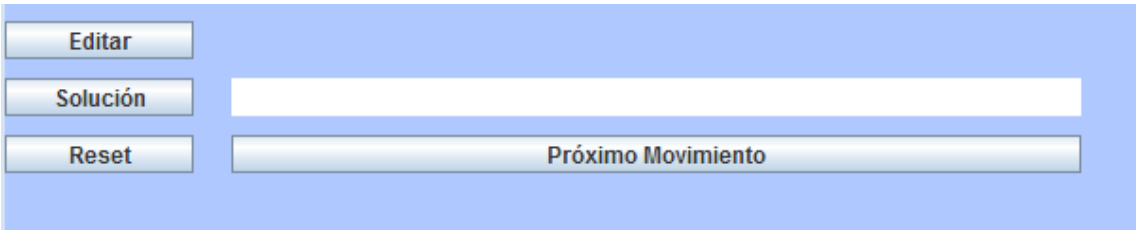


Este panel contiene un desplegable en el que se muestran las distintas opciones que se pueden elegir para mostrar la ayuda de cada una de ellas. También hay un cuadro de texto en el cuál aparecerá la explicación de cada una de las opciones.



Las opciones incluidas en la Ayuda son: los botones, la notación que se da en la solución y los movimientos, en los que se explica cómo se deben de mover las piezas según el movimiento que toque hacer.

Panel de Herramientas:

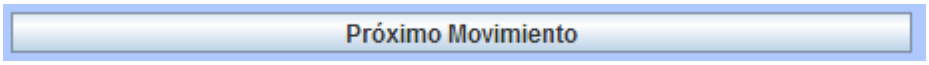


En este panel se pueden ver todos los métodos, que son los botones, que se pueden utilizar en el applet. Los botones permiten al usuario acceder a distintas formas para cambiar la configuración del cubo o generar la solución una vez que esté desordenado.

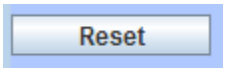
Explicaremos para que sirve cada uno de los botones del panel de herramientas.

El botón *Editar* sirve para cambiar de modo al *modo Editar*, en un principio el Applet se inicia en el *modo Jugar*. Este mismo botón se convierte en el botón *Jugar* cuando es presionado, y su función será la inversa de éste, volver al *modo Jugar*.

El botón *Solución* generará los pasos que se deben de seguir para resolver el cubo una vez esté desordenado. La solución saldrá impresa en el cuadro de texto que tiene al lado.

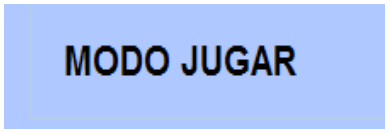


Este botón sirve para que una vez dados los pasos que hay que seguir resolver el cubo, se generar el movimiento en el dibujo del cubo, y en el cuadro de texto de la solución indique cual será el siguiente movimiento.



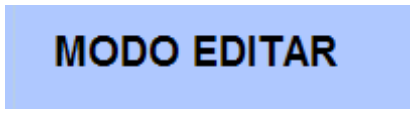
El botón *Reset* sirve para devolver al cubo a su estado inicial.

Panel de Modo:



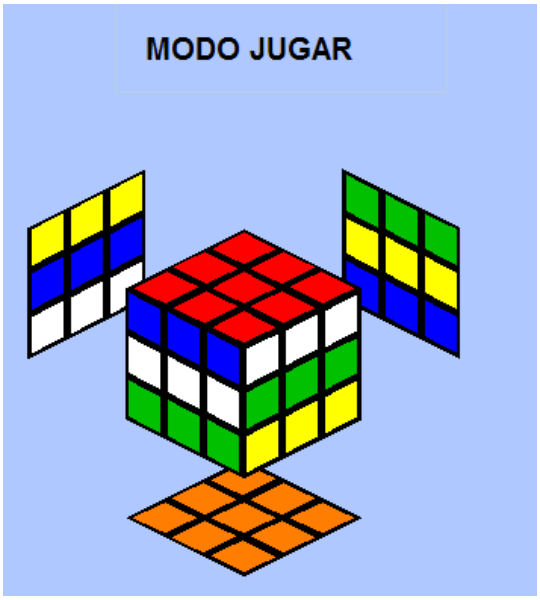
Este panel únicamente informa el modo en el que se encuentra el Applet, bien en *modo Jugar* o en *modo Editar*.

El modo cambiará siempre que se pulse el botón *Editar* o *Jugar*, según sea su modo actual.



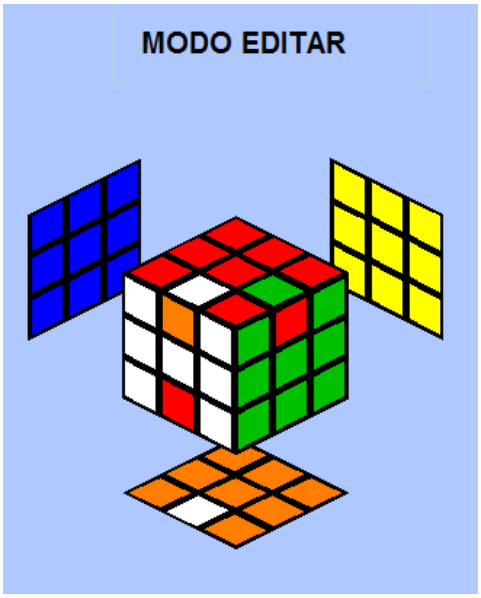
La diferencia de los Modos Editar y Jugar es la siguiente:

En el *modo Jugar*, las piezas se moverán como sucede en la realidad, se moverá la cara entera, como podemos ver en esta imagen.



Se puede intuir que las dos caras que se han movido han sido la cara superior y la cara de abajo.

En el *modo Editar* la modificación será por piezas individuales, es decir, se pondrá cada pieza en el lugar que se desee, intercambiándose así por la que esté en el lugar que desea ocupar.

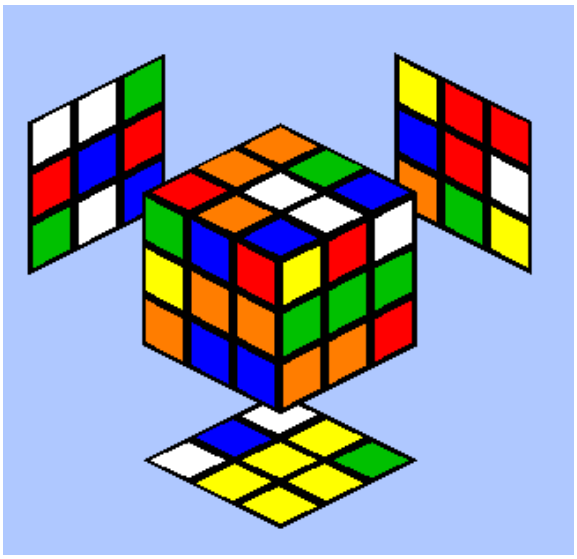


Se puede ver claramente que las piezas que no están en su lugar, han sido intercambiadas por otras, sin necesidad de mover la cara.

Estos movimientos no tienen porque generar una configuración válida del cubo, mientras que en el *modo Jugar*, la configuración siempre será correcta.

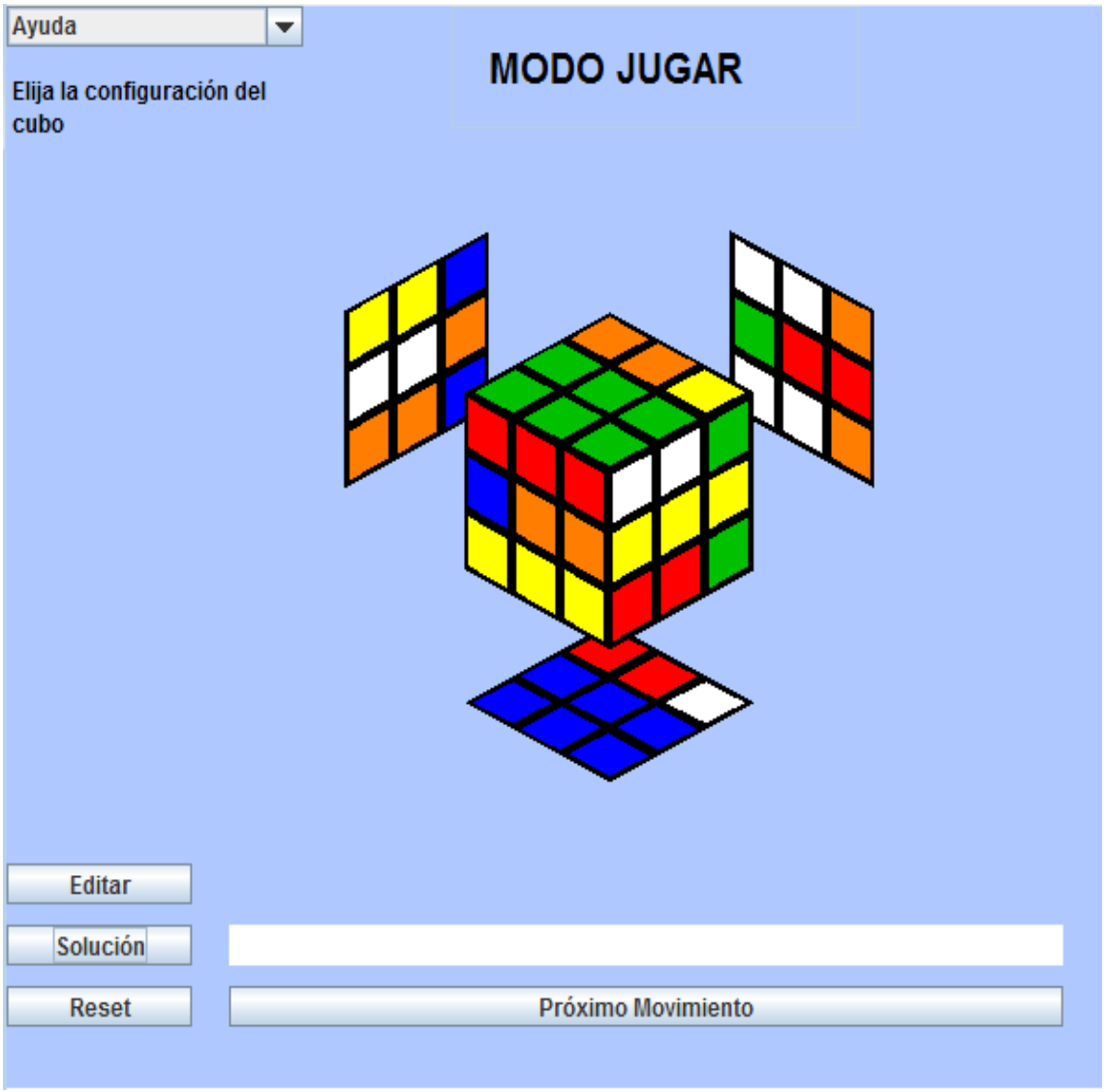
Panel de dibujo:

Este panel consta de la implementación del dibujo del cubo, e irá cambiando según ocurran distintos sucesos.

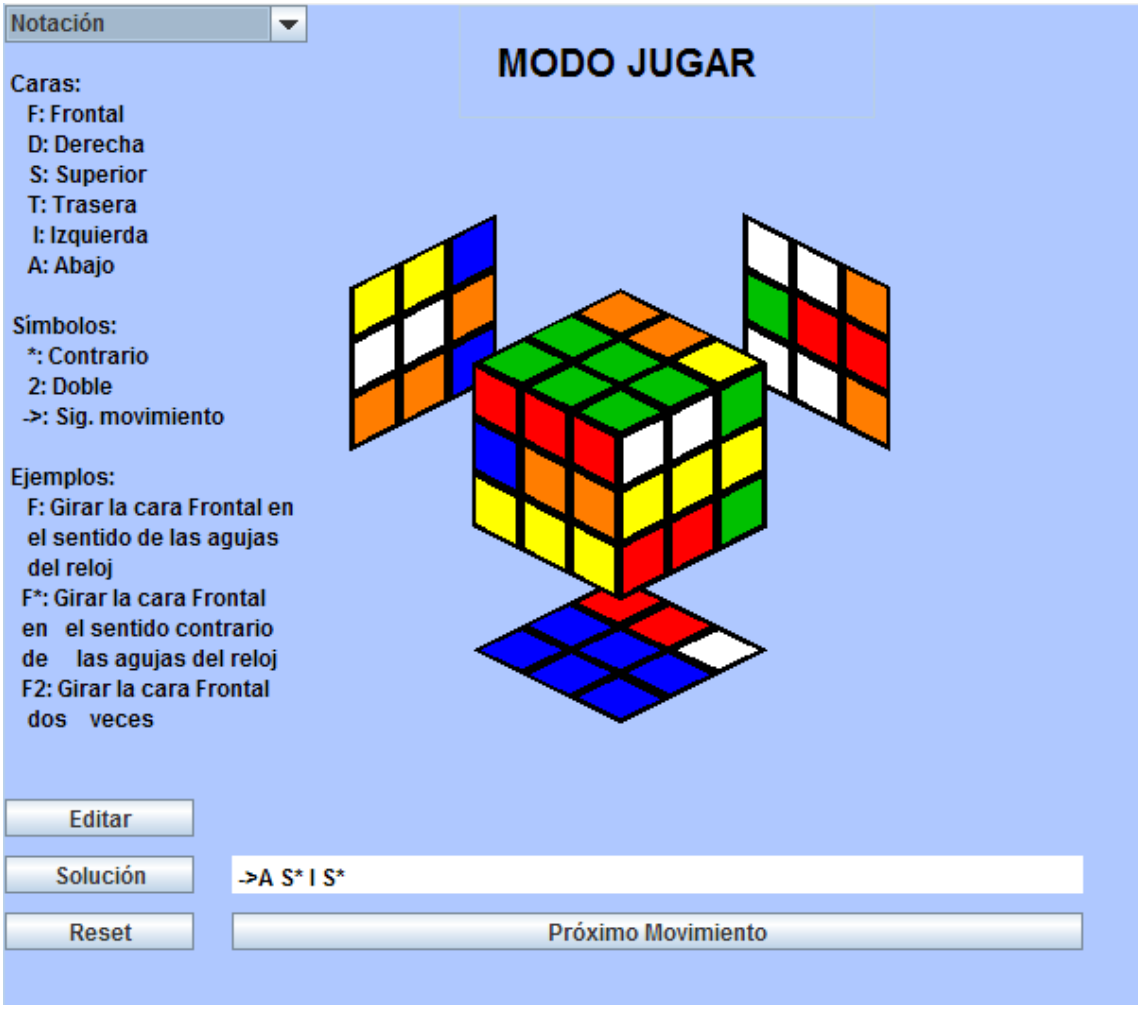


4.2. Ejemplo de funcionamiento del Applet:

En primer lugar generaremos una configuración del cubo sencilla, para que no sea demasiado pesada la explicación. Para ello haremos tres movimientos de cuatro caras en el *modo Jugar*, para que la configuración sea correcta.



Una vez tenemos ya la configuración, procederemos a utilizar el botón *Solución*, que la imprimirá en el cuadro de texto. Como somos principiantes, también activaremos el *panel de ayuda de notación*, para entender cuáles deben de ser los movimientos.



Como vemos, la solución que nos ha generado en principio, sin saberse la notación, no tiene mucho sentido. De modo que el panel de notación nos ayudará a comprender los pasos.

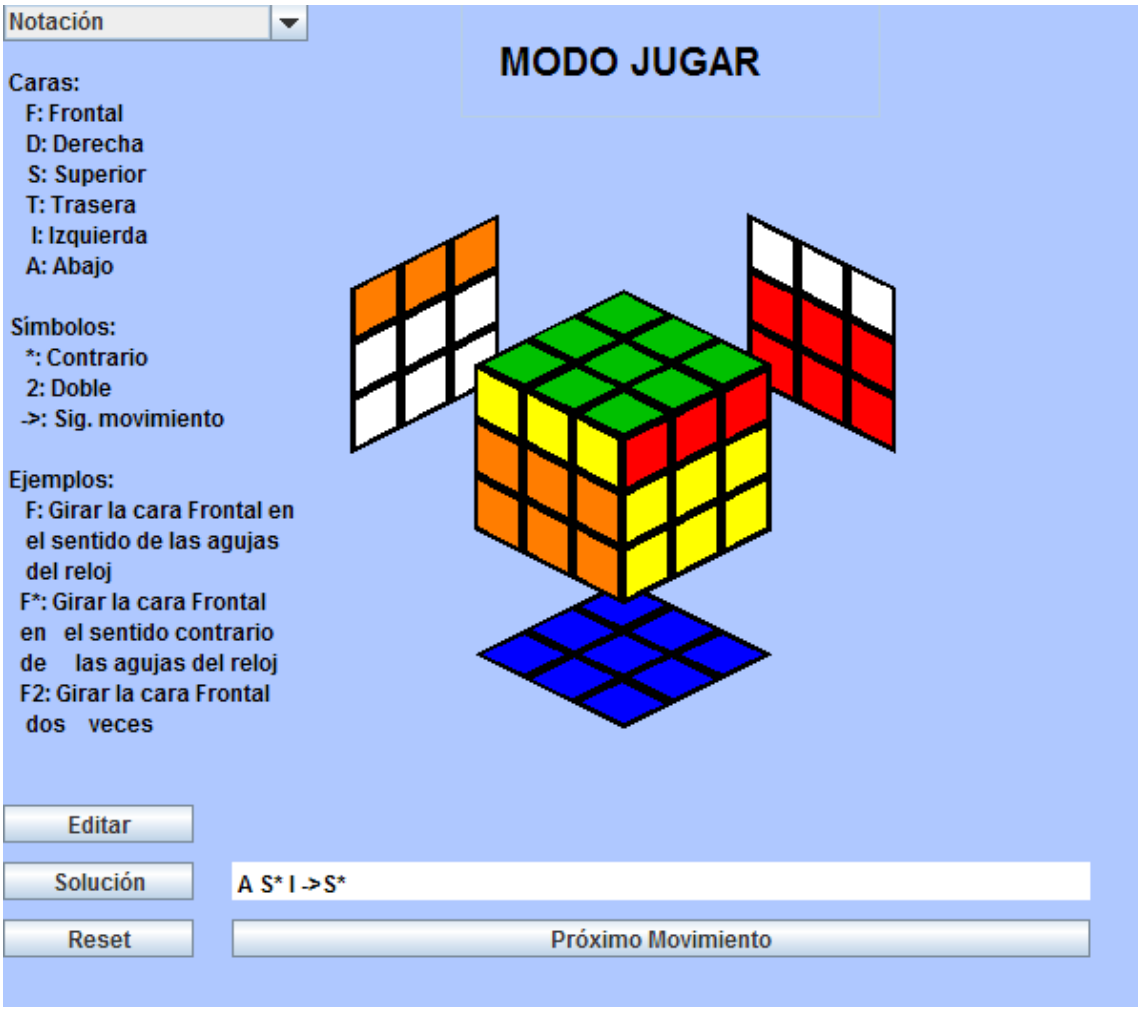
Vemos que en primer lugar viene el símbolo -> seguido de una *A*, lo que significa que el siguiente movimiento que habrá que hacer es *Girar la cara de abajo [A] en el orden de las agujas del reloj*.

Para ello, hay dos posibles alternativas:

- Apretar el botón de *Próximo Movimiento* pasando así automáticamente al siguiente paso.
- Mover nosotros la cara correspondiente tal y como nos dice la solución.

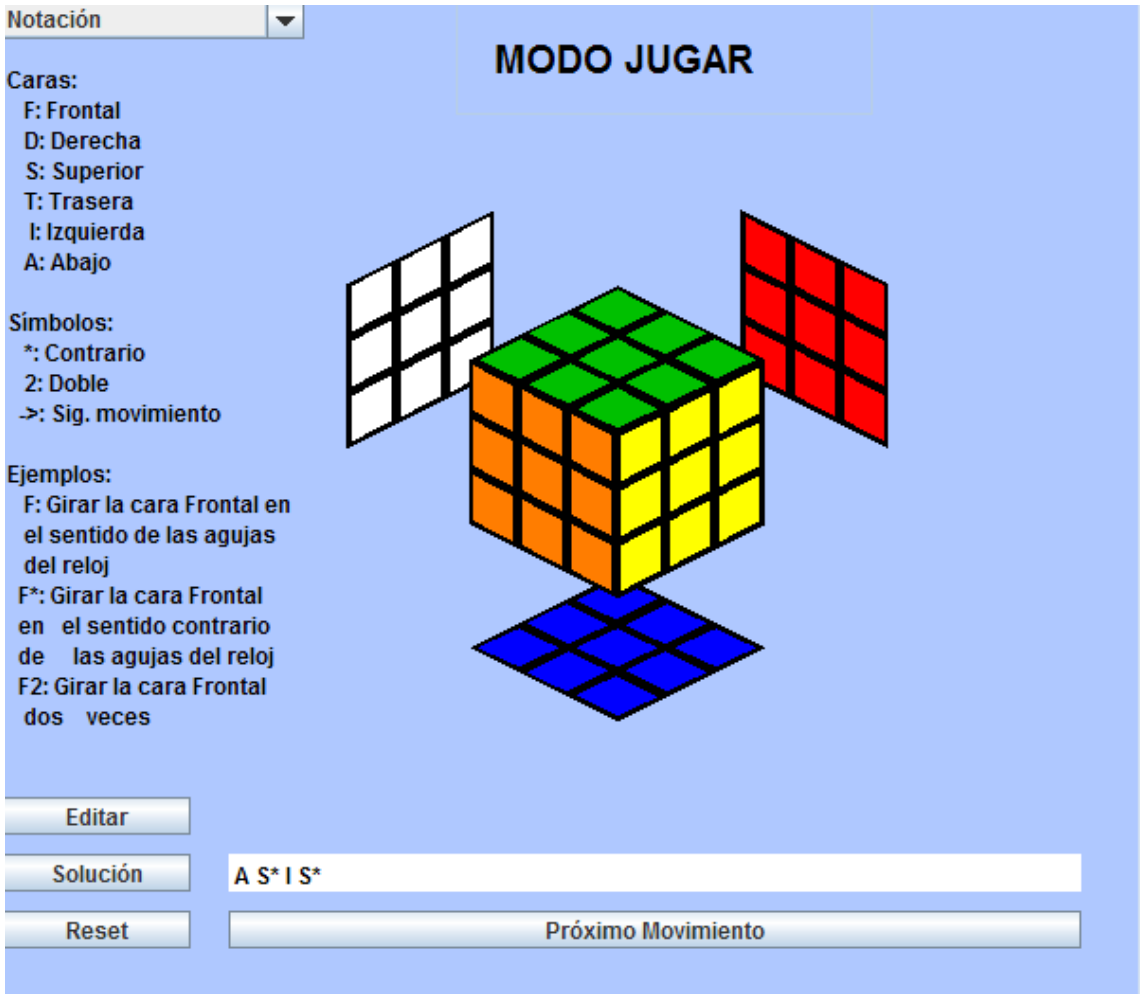
Ambas alternativas son válidas, pero si no dominamos bien los movimientos podremos equivocarnos en alguno y echar a perder lo anterior.

De modo que iremos apretando el botón de Próximo Movimiento hasta que nos quede, por ejemplo, un paso.



Llegados a este punto, únicamente nos queda un movimiento para completar nuestro cubo de Rubik.

La solución nos dice que el siguiente paso es *S**, mirando el panel de ayuda vemos que S hace referencia a la cara superior, mientras que * significa contrario. De modo que habrá que *girar la cara superior en el sentido contrario a las agujas del reloj*.



De este modo, habremos conseguido resolver el cubo de Rubik.

4.3. IMPLEMENTACIÓN:

Las clases que han sido necesarias para la implementación en java del proyecto han sido las siguientes:

- *PosicionesCubo*
- *Movimientos*
- *Configuración*
- *Solución*
- *SolucionKociemba*
- *Vista*
- *VistaPerspectiva*
- *Interfaz*

De estas clases, *Vista, VistaPerspectiva* sirven para dibujar el cubo. *PosicionesCubo y Movimiento* tienen que ver con *Solución* y *SolucionKociemba*, ya que a partir de ellas van a ser posibles los movimientos del cubo y la recogida de datos para la realización del algoritmo de resolución. *Configuracion* es una clase que está directamente involucrada con todas las demás, ya que en ella hay variables globales que intervienen en todas las clases. La clase *Interfaz* sirve para crear el Applet y también se apoya en la *Vista* y *VistaPerspectiva*, que las inicializa y las pinta, así como en *Solucion*.

Puesto que hay una gran cantidad de código solo se explicaran las partes más importantes de cada clase.

4.3.1. Implementación de la vista del cubo:

Vista.java

Esta clase será utilizada para crear el cubo en el panel. Contiene las características básicas que tendrá el cubo, es decir, el color del cubo de fondo y de las demás caras, así como la anchura y la altura. También contendrá características generales y los objetos gráficos. Sus atributos son los siguientes:

```
Image offImage;
Graphics offGraphics;
int width, height;

Color baseColor = new Color(0,0,0);                    //cubo negro
Color colors[] = {
                    new Color(0,0,255), //azul
                    new Color(255,0,0), //rojo
                    new Color(255,255,255), //blanco
                    new Color(0,192,0), //verde
                    new Color(255,125,0), //naranja
                    new Color(255,255,0) //amarillo
};

Configuracion configuracion;
ActionListener main;
```

- ***public void inicializar()***

  Crea la estructura en la que irá pintado el cubo.

- ***public void reset()***

  Pinta el cubo en su forma inicial.

- ***private void editMove(int f1, int f2)***

  Intercambia las piezas en el modo editar.

- ***public boolean showMove(int face, int qu)***

  Permite el moviente de las caras en el modo jugar o al generar la solución.

- ***public void mousePressed(MouseEvent e)***

  Procedimiento que se ejecuta cuando el usuario hace click con el ratón sobre el panel del cubo.

- ***public void mouseReleased(MouseEvent e)***

  Procedimiento que se ejecuta cuando el usuario deja de presionar algún botón del ratón sobre el panel cubo.

- ***public void mouseDragged(MouseEvent e)***

  Procedimiento que se ejecuta cuando el usuario arrastra el ratón sobre el panel del cubo.

- ***public void tryMove(int m, int q)***

  Procedimiento que prueba un movimiento y si es correcto lo ejecuta.

- ***void doMove(int m, int q)***

  Procedimiento que ejecuta un movimiento.

- ***void doEvent(boolean user)***

  Procedimiento que ejecuta un evento en el cubo, como puede ser un movimiento de piezas.

- ***public void update(Graphics g)***

  Procedimiento que actualiza el dibujo del cubo de Rubik.

VistaPerspectiva.java

Esta clase contiene todos los atributos del cubo para pintarlo y para realizar los movimientos. Sus atributos son los siguientes:

```
//Representación gráfica de los datos
Polygon faceletsI[] = new Polygon[54];
Polygon faceletsO[] = new Polygon[54];
Polygon faceOriMarks[][] = new Polygon[54][4];
Polygon cubeBack[] = new Polygon[4];
```

- ***public ViewerDiag(int x, int y, Settings s, ActionListener m)***

  Constructor de la clase, a partir de los datos que recibe pintará el cubo en su panel.

- ***private void dopoly(int x0, int y0, int dx1, int dy1, int dx2, int dy2, int c, double b)***

  Procedimiento que marca los límites del cubo.

- ***public void paint(Graphics g)***

  Procedimiento que dibuja el cubo.

- ***protected void checkMouseMove(int x, int y, int d)***

  Procedimiento que comprueba si las coordenadas están demasiado lejos para poder mover una pieza.

- ***protected int getFacelet(int x, int y)***

  Función que devuelve el número de pieza del cubo dada sus coordenadas.

- ***private void executeMouseMove(int f, int x, int y)***

  Procedimiento que mueve la pieza f a la coordenada x,y.

4.3.2 Implementación del algoritmo Kociemba:

Solucion.java

Clase principal para generar la solución que utiliza a la clase SolucionKociemba para generar la solución al problema. Contiene los siguientes atributos:

```
boolean preparado=false;       //variable para ver si se puede resolver.
boolean pararsol=false;//variable para parar el algoritmo de
                                        resolución.
boolean running=false;//variable que indica si está ejecutando.
int sollong = 0;           //tamaño de la solución.
int solmoves[] = new int[40];//array para los movimientos de la
                                        solución.
int solcantmoves [] = new int[40];//array para la cantidad de
                                        movimientos.
int posicionlista[][];//array que contiene la posición actual.
Configuracion configuracion;
ActionListener main;
```

- ***public void run()***

  Procedimiento que comienza la ejecución si está preparado para resolverlo, sino calcula las tablas.

- ***public void stopSolving()***

  Procedimiento que sirve para parar la ejecución del algoritmo.

- ***void doEvent(int t)***

  Ejecuta una acción dependiendo del resultado del algoritmo.

- ***protected boolean parityOdd(int pieces[], int start, int len)***

  Procedimiento que devuelve true si la permutación es impar.

- ***protected void num2perm(int pieces[], int start, int len, int pos)***

  Procedimiento que convierte el número pos en una permutación y lo pone en el array pieces[].

- ***protected void num2partperm(int pieces[], int start, int len, int np, int p0, int pos)***

  Procedimiento que convierte el número pos en una permutación de np elementos y lo pone en el array de pieces[].

- ***protected void num2ori(int pieces[], int start, int len, int val, int pos)***

  Procedimiento que convierte el número pos en la orientación y lo pone en el array de pieces[].

- ***protected int ori2num(int pieces[], int start, int len, int val)***

  Procedimiento que convierte la orientación de los vértices en un número.

SolucionKociemba.java

Esta clase contiene todos los elementos necesarios para poder aplicar el algoritmo de resolución sobre el cubo de Rubik desordenado.

Esta clase tiene muchos atributos, arrays, para la formación de las tablas y todas las posibles soluciones. Otros atributos importantes son:

```
int phase1len=0;        // profundidad de la búsqueda en la fase1
int phase2len=0;        // profundidad de la búsqueda en la fase2
int maxdepth = 25     // tamaño de la mejor solución encontrada
```

- ***public boolean setPosition(CubePosition cubePos, boolean test)***

  Función que devuelve un booleano dependiendo de las tabla que haya calculado si es buena o no.

- ***protected boolean solve()***

  Procedimiento que aplica el algoritmo IDA* en la fase 1, si no hay una solución mejor abandona el procedimiento.

- ***private boolean search1()***

  Hace una búsqueda en profundidad a través de todas las posiciones del vector de fase1 en profundidad. Si no hay solución devuelve false, si la hay se queda con la más prometedora.

- ***private boolean solve2()***

  Procedimiento que aplica el algoritmo IDA* en la fase2, siempre y cuando no se esté aplicando aun en la fase1, posteriormente coge la mejor solución.

- ***private boolean search2()***

  Hace una búsqueda en profundidad a través de todas las posiciones del vector de fase2 en profundidad, Si no hay solución devuelve false, si la hay se queda con la mejor.

- ***protected void init()***

  Procedimiento que inicializa todos los datos. Calcula los datos de las tablas y la posición de todas las piezas.

PosicionesCubo.java

En esta clase se encuentran todas las posiciones posibles del cubo, sus rotaciones y sus posibles movimientos. Esta clase no contiene prácticamente subprogramas de valor, pero sus datos son imprescindibles para la ejecución del algoritmo.

Movimientos.java

Esta clase genera el resultado, es decir, convierte los resultados obtenidos en la secuencia de movimientos que hay que seguir para desarrollar la solución del cubo de Rubik. Consta únicamente de un fragmento de código importante:

- ***public String toString(boolean inverse, int pos)***

  Esta es la rutina que imprime la solución final en la interfaz gráfica.

4.3.3. Implementación del Java Applet:

Interfaz.java

La clase Interfaz contiene toda la parte visual del Applet. Para ello se han creado los siguientes objetos gráficos.

Botones JButton

BotonEditarJugar

BotonResetear

BotonSolucion

BotonPasoSiguiente

Cuadros de texto JTextPane

Ayuda

Pasos

Desplegable JComboBox

CboBox

Cuadros de texto JTextField

ModoJugarEditar

Paneles JPanel

helpPanel – Contiene *Ayuda* y *CboBox*

botPanel – Contiene *BotonEditarJugar BotonResetear BotonSolucion* y *BotonPasoSiguiente Pasos*

topPanel – Contiene *ModoJugarEditar*

Vista *vistaCubo*, contiene *VistaPerspectiva*

Las operaciones que implementa son:

- ***public void init()***

  Procedimiento que se inicia al comenzar el Applet. Su función es inicializar todos los valores.

- ***public void stop()***

  Procedimiento que para la ejecución del algoritmo cuando se aprieta de nuevo el botón Solución.

- ***public void solve()***

  Procedimiento que inicia el algoritmo de Kociemba cuando se aprieta el botón de Solución.

- ***public void actionPerformed(ActionEvent e)***

  Procedimiento que se ejecuta de manera distinta según el lugar en el botón en el que se clickee.

- ***private void stepForward()***

  Procedimiento que avanza una posición en el algoritmo de resolución cuando se pincha en el botón de Próximo Movimiento.

5. Conclusiones y trabajo futuro

5.1. Conclusión:

Desarrollar en el proyecto de fin de carrera un Applet Java me ha sido de gran utilidad para conocer mejor el lenguaje Java. Antes de empezar el proyecto, si que había oído hablar sobre la programación en Applets pero nunca me había interesado por ella, y puesto que tampoco tenía una base de Java muy buena he tenido que adquirir conocimientos del lenguaje continuamente durante la realización del proyecto.

He disfrutado especialmente con la creación de clases de objetos gráficos, tanto la interfaz que ha sido más sencilla como el conseguir dibujar el cubo con un aspecto bastante bueno pese a no haber tenido una base sobre la que apoyarme.

Debido al desarrollo del proyecto, he aprendido la resolución del cubo de Rubik mediante dos algoritmos distintos, uno bastante básico y otro más complejo en el que hay que conocer mucho más a fondo todas las características de cubo. Este algoritmo, pese a tener una dificultad mayor y en términos poco tiene que ver con lo que estamos habituados al cubo de Rubik ha servido para solucionarlo.

Con la conclusión del proyecto he podido comprobar que parte de los conocimientos de asignaturas, a priori distintas, como son las relacionadas con las matemáticas y las de programación, son aplicables conjuntamente al desarrollo de un proyecto único, y por tanto, interdisciplinar.

5.2. Trabajo futuro

Una ampliación de este proyecto podría ser el desarrollo de otro algoritmo de resolución para cubos de otro tamaño, ya sean 2x2x2, 4x4x4 o cualquier otro. Ya que el algoritmo de resolución sería muy parecido.

Otra mejora sería el llegar a implementar el dibujo del cubo en 3D, pudiendo de este modo moverlo y ver en una interfaz más real cómo serían los movimientos, incluyendo la posibilidad de generar distintas soluciones según el número de movimientos para gente que en un principio no controle el desarrollo del cubo.

6. Bibliografía

[1] Título: *Programación Java.*

Autor: Yakov Fain. Editorial(Anaya). AÑO 2011

[2] Título: *Finding Optimal Solutions to Rubik's Cube Using Pattern Databases.*

Autor: Richard Korf.

Link: http://www-compsci.swan.ac.uk/~csphil/CS335/korfrubik.pdf

[3] Autor: Herbert Kociemba.

Link: http://kociemba.org/cube.htm

[4] Enciclopedia de contenido libre de la web.

Link: www.wikipedia.org

[5] Web en la que vienen varios ejemplos de cómo usar los objetos de las interfaces gráficas o la creación de applets.

Link: http://www.tutorial-enlace.net/tutoriales-Java.html

[6] Applet similar con el que jugar.

Link: http://www.schubart.net/rc/