UNIVERSIDAD NACIONAL

JOSÉ MARÍA ARGUEDAS



ESCUELA PROFESIONAL DE INGENIERIA DE SISTEMAS

**DOCENTE: ing. Iván Soria Solís**

**INTEGRANTES:**

- Rómulo Quispe Altamirano

- Yaison Huaman Mallqui

- Alex Huayllas Chircca

ANDAHUAYLAS 2014

**Proyecto de CUBO**

**Introducción**

Un poco de historia del cubo mágico o cubo rubik como quieran llamarlo, fue inventado por el escultor y profesor de arquitectura Húngaro Erno Rubik en 1974 es un rompecabezas mecánico tridimensional.

Gano el premio Alemán a mejor juego de rompecabezas, hasta el momento se han vendido 350 millones de cubos en todo el mundo (2009), y es el más vendido del mundo.

Bueno, el cubo Rubik tiene seis caras cubiertas por nueve pegatinas de seis colores uniformes (tradicionalmente blanco, rojo, azul, verde y amarillo).

El método consiste en resolver el cubo de Rubik por capas, primero la superior, luego la central, y por último la inferior. Con un poco de practica el cubo Rubik se puede armar en menos de 2 minutos, y con mucha practica incluso se puede bajar treinta segundos. (Fuente: Wikipedia).

Crear una aplicación para cubo Rubik es un desafío debido a la amplitud del problema. Las aplicaciones actuales buscan resolver el cubo de manera óptima, eso implica una exploración de un espacio de estados enorme, lo que consumiría mucho tiempo. Hoy en día se usan distintos métodos para encontrar la solución óptima en el menor tiempo posible. El resultado será una cadena de movimientos que resuelve un cubo dado, y dicha solución la obtendremos casi al instante.

**OBJETIVO:**

El objetivo del presente proyecto desarrollar una aplicación CUBO, valiéndose o utilizando lenguajes de programación como java con el entorno de trabajo IDEnetBeans8.0.

## REQUISITOS FUNCIONALES Y NO FUNCIONALES:

### REQUISITOS FUNCIONALES

Describen el comportamiento que debe adoptar el sistema. Qué acciones ha de desarrollar, cómo las debe de llevar a cabo y qué resultado se debe dar para cada una de ellas.

* Interfaz presentable
* Inicializar el juego
* Controles
* desordenar
* Mover
* Resolver
* Salir

### LISTA DE REQUISITOS NO FUNCIONALES

* **Sistema operativo.**

El sistema operativo será win7 para la implementación y desarrollo de la aplicación del proyecto de CUBO.

* **Requerimientos de producto**

Se desarrollara en el lenguaje de programación JAVA V java con el entorno de trabajo IDEnetBeans8.0, utilizando un Applet, también utizamos las propias librerías de Java

* **Requerimientos organizacionales**
* Implementación de la aplicacion tendrá una duración de cuatro meses.
* **Requerimiento externo.**
* El cliente puede modificar agregar al código de CUBO, eliminar, etc.
* **Requerimiento calidad:**
* Para el buen funcionamiento del proyecto de CUBO, es necesario que esta se mueva en unos márgenes de calidad adecuados.

**Otros requisitos.**

* **Requisitos que deben cumplir los usuarios**

Los usuarios o jugadores que quieran jugar el CUBO, tendrán que primero leerse los manuales del CUBO en donde nos explica paso a paso de cómo se utiliza el CUBO.

**Requisitos funcionales y no funcionales**

**Descripción del cubo:**

Es un cubo de 3x3x3 que está formado por 6 caras móviles con 9 casillas en cada una de ellas. Cada casilla esta etiquetado con el color correspondiente a su cara.

**UP**

Cara superior, abreviada con la letra U (color BLANCO)

**LEFT**

Cara izquierda, abreviada con la letra L (color AMARILLO)

**FRONT**

Cara frontal, abreviada con la letra F (color NARANJA)

**RIGHT**

Cara derecha, abreviada con la letra R (color ROJO)

**BACK**

Cara trasera, abreviada con la letra B (color VERDE)

**DOWN**

Cara inferior, abreviada con la letra D (color AZUL)

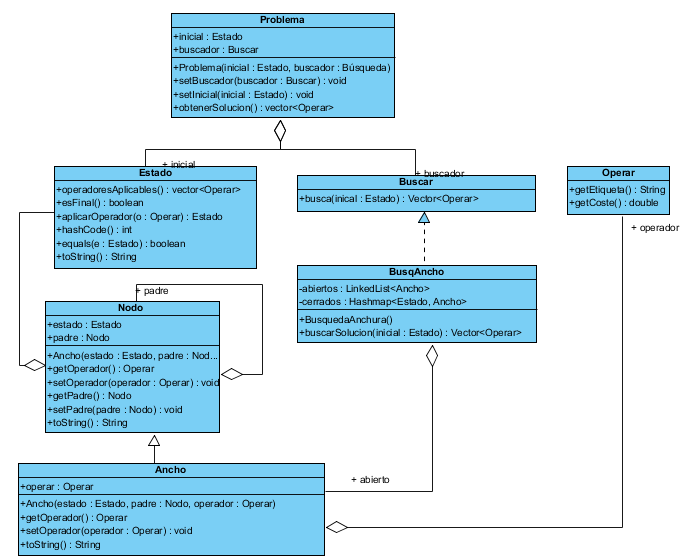
Los doce movimientos posibles en un cubo, por cada una de las caras anteriores.

* Seis giros de ¼ de vuelta en el de las agujas del reloj sobre cada una de las 6 caras
* Notación: U, D, L, R, F, B
* Seis giros de ¼ de vuelta en el sentido contrarios a las agujas del reloj sobre cada una de las 6 caras
* Notación: Ui, Di, Li, Ri, Fi, Bi

**El código está estructurado en 3 paquetes:**

1. **Paquete buscar:** este interfaz es genérico utilizado para la implementación de algoritmos de búsqueda.

* **Diagrama de clases:**
* **Clases:**
* Problema: Es la clase genérica que resuelve mediante búsquedas en espacio de estado.
* Nodo: Es la superclase con la definición de nodos (contiene estados) a usar en los algoritmos de búsqueda.
* **Interfaz:**
* **Buscar:** Usa los métodos de búsqueda.
* **Estado:** Objetos que almacenan estados del problema.
* **Operar:** Objetos que tomen el papel del operador.
* **Ancho:** Implementación de la búsqueda en anchura haciendo uso de listas ABIERTAS Y CERRADOS.
* **Nodo:** Nodos de búsqueda en anchura.

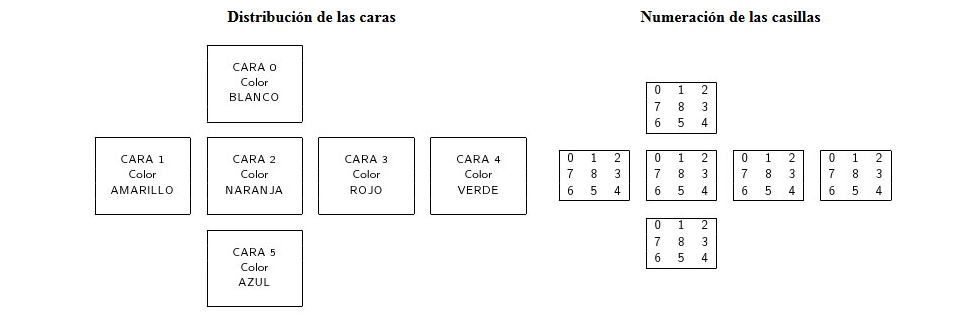


1. **Paquete Figura:** Este paquete se encarga de gestionar al cubo de rubik.

* **Diagrama de clases:**
* **Clases:**
* **Fcubo:** Encapsula la informacion sobre un cubo para gestionarlo.
* Cada cubo esta formado por seis caras.
* Incorpora metodos para duplicar y comparar cubos, determinar si es solucion y aplicar movimientos sobre las caras.
* Incluye arrays estatico con informacion sobre la vecindad de cada cara y los indices de las casillas “fronterizas” de cada cara vecina.
* **Cara:** Encapsula la informacion sobre una cara de un cubo e informacion adicional para gestionar.
* Cada cara almacena en un arrays de nueve posiciones la situacion de sus casillas.
* La distribucion de las casillas en este array sigue la distribucion en esperal.
* **Casilla:** Encapsula la informacion sobre una casilla de una cara de cubo.

Para cada casilla se almacenan dos atributos.

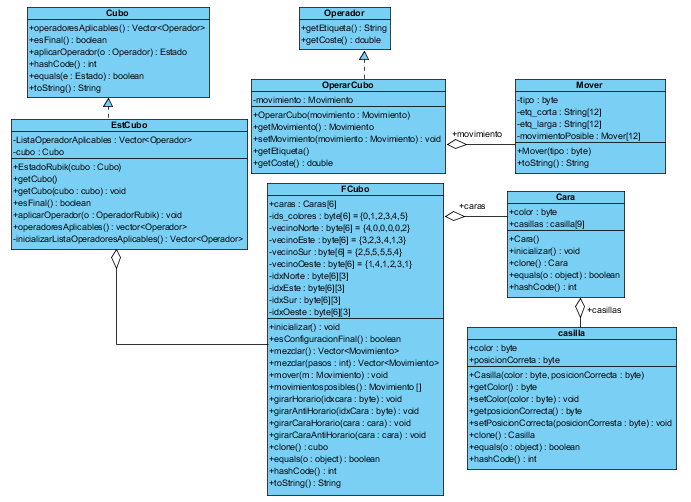
* Color: un valor de tipo byte que identifica el color de esa casilla. Es decir, nos indica implictamente en que cara deberia de estar situada.
* PosicionCorrecta: Un valor de tipo byte que indica la posicion que esa casilla deberia ocupar en el array de casillas asociada a la cara a la que pertenece, conforme a la distribucion “en espiral”.
* **Mover:** Encapsula los movimientos permitidos sobre los cubos.
* Por razones de eficiencia los objetivos correspondientes a los doce movimientos posibles ya estaran creados y se almacenan en atributos estaticos.



* **EstCubo:** Implementacion del interfaz cubo.Busqueda.Estado en el que delega las operaciones del interfaz estado.
* Por razones de eficiencia, un array con los 12 operaciones aplicables se almacenan en un vector estatico asociado a la clase para no tener que crearlo cada vez que se le pida al estado la lista de operaciones que son aplicables sobre él.
* **OperarCubo:** Implementacion del interfaz cubo.busqueda.operador, contiene un objeto cubo.figura.movimiento.

1. **Paquete Grafico:** se incluye unicamentepara experimentacion, no es necesario utilizarlo en la practica a entregar.

* **Clases:**
* **JpCubo:** Encapsula al applet AnimCube en un JPanel, añadiendo métodos para recuperar el cubo mostrado en pantalla y para establecer su configuración.
* **JfCubo:** Ejemplo de ventana(JFrame) donde se usa JpCubo para mostrar y manipular graficamente un cubo.



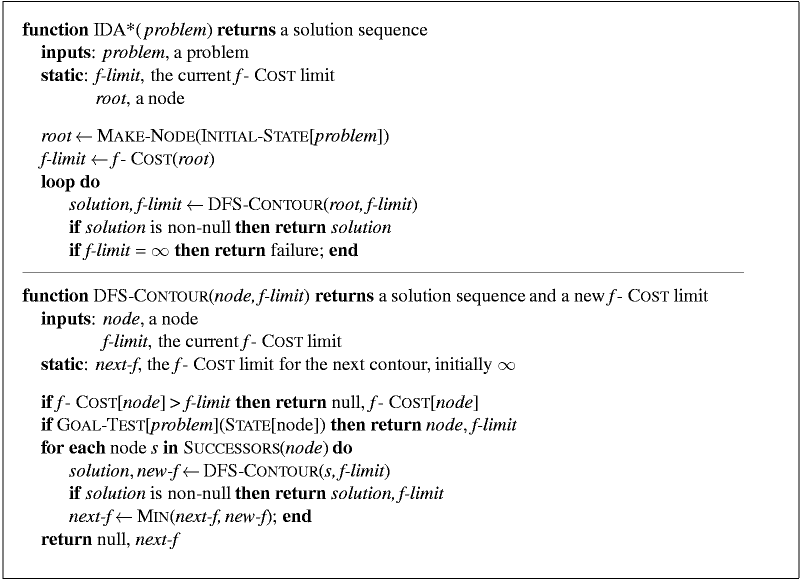
**Resultados**

Es muy importante mencionar que el cubo usa el algoritmo de IDA.

IDA es una variante de la búsqueda en profundidad iterativa que usa el esquema de evaluación de la búsqueda y se representa de esta manera A\*(f (e)=g (e)+h (e)) esta función es para controlar el alcance de cada una de las iteraciones.

El funcionamiento de IDA es:

* Realiza una serie de búsquedas en profundidad.
* En cada iteración se utiliza una cota que señala el valor máximo permitido para la función f.
* La cota inicial corresponde con el valor f del ESTADO\_INICIAL.
* Al generar sucesores no se insertan en ABIERTOS estados cuyo valor f sea mayor que la COTA actual.
* Si termina una iteración sin encontrar solución, se reinicia la búsqueda con una COTA actual mayor.
* La nueva COTA será el menor de los valores de f descartados en la iteración anterior



Con este algoritmo llegamos a una solución muy óptima ya que ayuda a resolver el cubo de una fácil.

**Bibliografía:**

[**http://es.wikipedia.org/wiki/Cubo\_de\_Rubik**](http://es.wikipedia.org/wiki/Cubo_de_Rubik)

[**http://programacion.net/articulo/java\_3d\_169/2**](http://programacion.net/articulo/java_3d_169/2)

[**http://ruwix.com/widget/AnimCube.java**](http://ruwix.com/widget/AnimCube.java)

[**http://panamahitek.com/java-3d-programando-en-la-tercera-dimension/**](http://panamahitek.com/java-3d-programando-en-la-tercera-dimension/)

[**http://panamahitek.com/java-3d-programando-en-la-tercera-dimension/**](http://panamahitek.com/java-3d-programando-en-la-tercera-dimension/)

[**http://www.java3d.org/**](http://www.java3d.org/)

Juan Antonio Palos. Modelo grafico de Java 3D recuperado de

**minipaint-mconde.googlecode.com/.../Modelado%20Gráfico%20(**Java**%20...**

Manual para el cubo Rubik

**www.jaimegarciaserrano.com/archivos/**RUBIK**\_PDF.pdf**