Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Propuesta de Proyecto Terminal

Algoritmo de búsqueda de configuraciones de enlaces ortogonales en dos y tres dimensiones

Torres Jiménez Pedro Eduardo 206303123

Trimestre 10P

18 de junio de 2010

Me comprometo a asegurar que el alumno cuente con los recursos necesarios

para que esta propuesta se realice satisfactoriamente.

Asesor: Dr. Francisco Javier Zaragoza Martínez
Número económico: 20197

_____

CONTENIDO

OBJETIVO GENERAL

Desarrollar un algoritmo que busque las diferentes formas abiertas y cerradas que puede adoptar un alambre articulado con enlaces unitarios y dobleces ortogonales en dos y tres dimensiones.

OBJETIVOS PARTICULARES

- Diseñar un módulo que busque configuraciones cerradas
- Diseñar un módulo que busque configuraciones abiertas
- Diseñar un módulo que detecte configuraciones isomorfas
- Diseñar un módulo que dibuje y articule enlaces unitarios
- Implementar el algoritmo generado en OpenGL

ANTECEDENTES

El estudio de la teoría de grafos y estructuras combinatorias surge en el siglo XVIII cuando se encontró la necesidad de estudiar estructuras finitas que tuvieran aplicaciones trascendentes. El documento escrito por Leonhard Euler sobre los puentes de *Königsberg* y publicado en 1736 es considerado como el primer trabajo en la historia de la teoría de grafos.

Después de varios siglos, esta rama de las matemáticas discretas ha tomado gran relevancia en la resolución de problemas en diversas áreas. Hay diversas estructuras estudiadas en la actualidad, entre ellas, las de la geometría computacional, donde el doblado de enlaces[1] juega un papel importante en su estudio.

Existen ya diversos proyectos importantes respecto al estudio de doblado y configuración de enlaces.Éstos son:

Referencias externas:

Folding and Unfolding Linkages, Paper, and Polyhedra [1], donde se pretende por medio de enlaces formar figuras de origami y poliedros con base en consideraciones especiales de estos enlaces.

A semejanza del proyecto que se desarrollará, este trabajo forma poliedros y superficies (configuraciones cerradas en dos y tres dimensiones), con la singularidad de que los ángulos para articular enlaces son variables, además que trata de formar figuras de papel origami dadas inicialmente, mientras que este proyecto se limita sólo a configuraciones ortogonales.

Linkages as functions [2], enlaces como funciones para interpretar el comportamiento en estructuras.

Este trabajo tiene una alta complejidad matemática, aunque sólo estudia las formas posibles que un grafo puede adoptar, dadas ciertas condiciones iniciales, a diferencia del proyecto a realizar, el cual formará las posibles figuras abiertas y cerradas.

Geometry and topology of polygonal linkages[3]. Formación de figuras poligonales (configuraciones cerradas de enlaces) con variación en los ángulos de doblado

Muy similar al mencionado anteriormente, difiere en la topología y consideraciones de isomorfismo; el proyecto a desarrollar pretende sólo desarrollar configuraciones distintas y distinguir isomorfas[2].

Referencias internas:

Generación de polígonos con triangulaciones ortogonales [4]. Este proyecto es semejante al propuesto, sólo que la unidad de formación de polígonos son triángulos ortogonales

---

1 Un enlace es una representación gráfica de tamaño uno, puede ser considerado como un pedazo de un alambre o barra

2 Dos o más configuraciones son isomorfas si tienen la misma forma geométrica, salvo una simetría

JUSTIFICACIÓN

Las estructuras que se pueden representar en forma de grafos están en todas partes, y muchos problemas de interés práctico pueden ser representados por grafos.

El desarrollo de algoritmos para manejar grafos, es de gran interés en ciencias de la computación.

En general, el interés está centrado en cómo objetos tales como los enlaces pueden ser movidos o reconfigurados, con base en ciertas condiciones, dependiendo del tipo de objeto y del problema de interés

Tales enlaces tienen aplicaciones en la construcción de tubos hidráulicos y en la planeación de movimientos de brazos para robots. También, en el área de conexiones y formación de proteínas en la biología molecular.

El proyecto a desarrollar pretende, por medio de la presentación de las formas en las que se pueden configurar los distintos números de enlaces, obtener formas abiertas y cerradas en dos y tres dimensiones, para formar criterios en la visión de proyectos relacionados de la toma de decisiones de la vida real. Además, pretende también, obtener un enfoque didáctico debido a la utilidad algorítmica.

El problema que se resuelve es el siguiente: Dado un número entero, el algoritmo se encarga de obtener las formas posibles y distintas que se pueden construir con tal número, de manera que no choquen o se encimen entre sí los enlaces. Estos enlaces tienen la particularidad de tener 5 grados de libertad en el espacio, es decir, pueden ser movidos hacia arriba, hacia abajo, a la izquierda, a la derecha y hacia enfrente, puesto que son ortogonales. Las configuraciones se determinan al ir probando las formas en que puede ir creciendo una configuración. Esto se logra al tomar decisiones respecto a dónde pueden ser movidos los enlaces sin chocar. La técnica algorítmica principal por usarse es la de búsqueda con retroceso.

La forma en la que se presentará la solución es por medio de OPENGL, el cual es una API(Application Programming Interface), que ofrece al programador una interfaz con gráficos a nivel hardware, y la ventaja de su uso es que corre en diferentes plataformas comparadas con otras APIs.

La posible continuación de este proyecto podría ser:

- La optimización de formas para minimizar costos en estructuras tipo tubería.
- La implementación del algoritmo en videojuegos multiplataforma.
- La mejora del algoritmo para su funcionamiento bajo condiciones complejas.

DESCRIPCIÓN TÉCNICA

Descripción global

El proceso inicia cuando, al abrir la aplicación, el usuario debe ingresar un número n, de tal forma que se inicia el algoritmo y se van mostrando las figuras obtenidas.

El proceso de ejecución del proyecto pasa por los siguientes módulos:

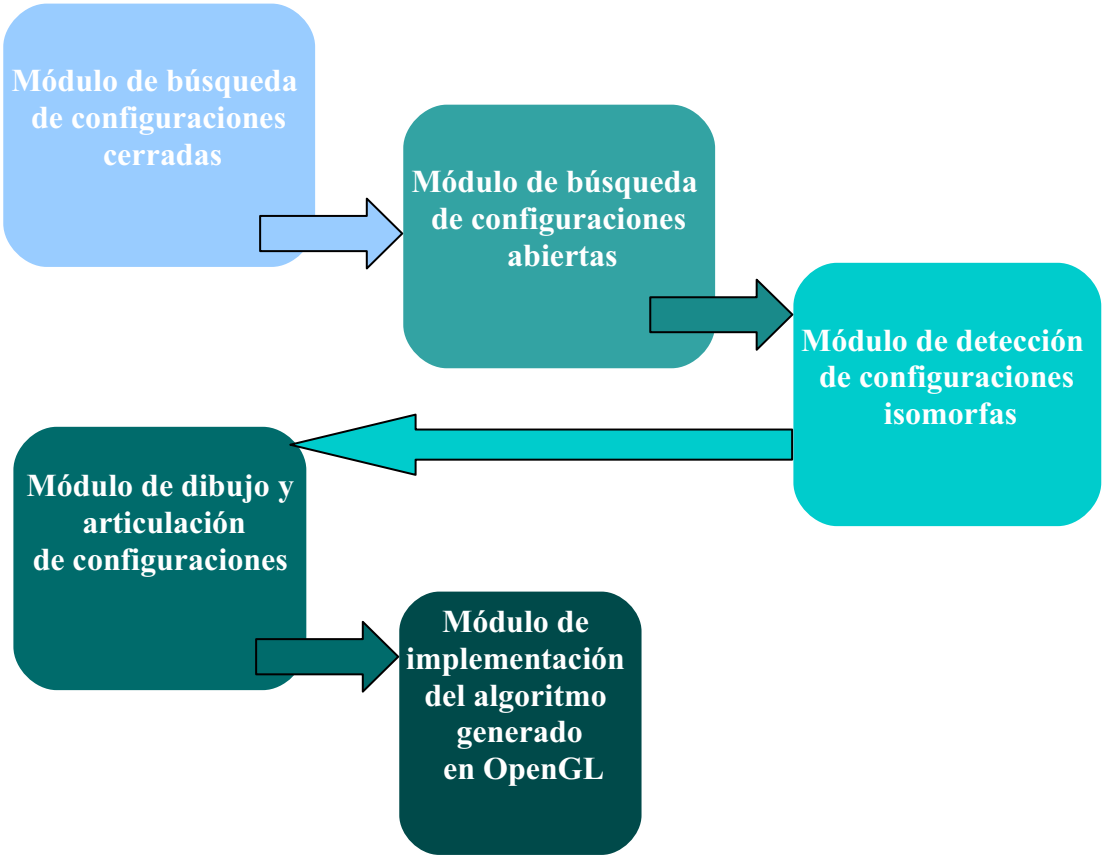


Figura 1. Diagrama de interacción de los módulos.

CONFIGURACIONES CERRADAS: En este módulo se determinan las formas cerradas en dos y tres dimensiones a partir de n enlaces. Habrá ocasiones en las que no existirán dichas configuraciones, en caso contrario, se mostrarán.

CONFIGURACIONES ABIERTAS: En este módulo se determinan las formas abiertas en dos y tres dimensiones a partir de n enlaces y se mostrarán.

DETECCIÓN DE CONFIGURACIONES ISOMORFAS: En este módulo se determinan las formas equivalentes o isomorfas de n enlaces. Al discriminar varias configuraciones se elegirá sólo una y se mostrará.

DIBUJO Y ARTICULACIÓN DE CONFIGURACIONES: En este módulo se detallan las formas en las que un enlace se puede mover o configurar en el espacio. Se hace la suposición de que cada enlace tiene tamaño 1. Al iniciar la ejecución se mostrará un alambre de enlaces como punto de partida.

IMPLEMENTACIÓN DEL ALGORITMO GENERADO: En este módulo se implementará la solución en OpenGL, para su visualización en 3D.

ESPECIFICACIONES TÉCNICAS

El desarrollo del proyecto concluye con un programa ejecutable, capaz de *renderizar*[3] gráficos 3D en una plataforma Linux, y a su vez ejecutar los módulos anteriormente descritos. El proceso inicia una vez abierto el programa ejecutable. El usuario ingresa un número entero n positivo y selecciona alguna de entre dos opciones (configuraciones abiertas o configuraciones cerradas), debido a la naturaleza del problema el algoritmo es un poco lento.

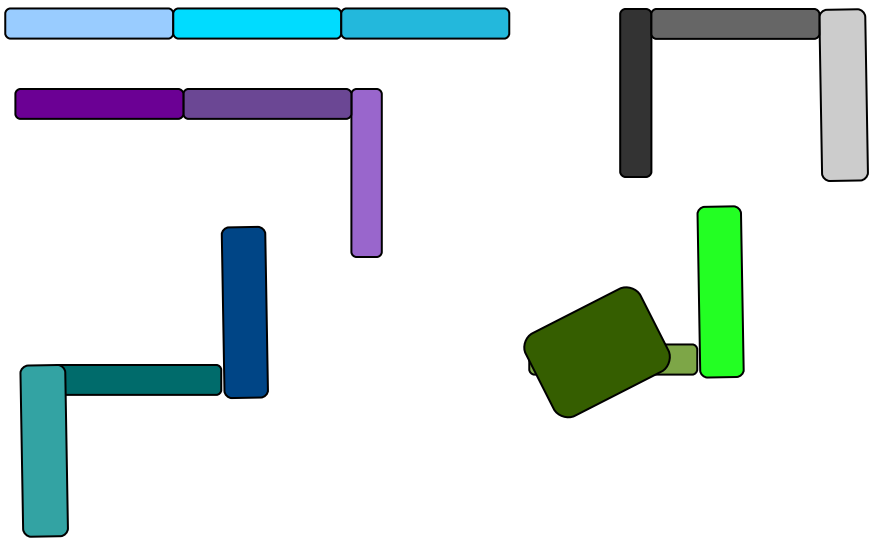


Figura 2. Posibles configuraciones abiertas en dos y tres dimensiones con tres enlaces.

Ésta ilustración muestra las posibles configuraciones abiertas en dos y tres dimensiones con n=3 (tres enlaces).

Las solución del problema está delimitada por el rango 1<=n<=20.

Los algoritmos que se implementarán serán diseñados en C++, OpenGL permite aplicarle multitud de funciones a los gráficos por medio del lenguaje C/C++

La plataforma de desarrollo será Ubuntu 9.10 en conjunto con OpenGL 3.2 y C++. Todos los módulos especificados arriba serán desarrollados en tales plataformas.

La plataforma de ejecución será en el Sistema Operativo Ubuntu 9.10

3 Renderizar es un término utilizado para referirse a la graficación de algún objeto en una API.

ENTREGABLES

Los productos que entregarán anexos al reporte final son:

- Modelado del sistema
- Código fuente
- Documentación
- Manual de usuario con ejemplos de uso
- Manual de instalación

BIBLIOGRAFÍA

[1]Demaine E.D,"Folding and Unfolding Linkages,Paper and Polyhedra",In Proceedings of the Japan Conference on Discrete and Computational Geometry, Lecture Notes in Computer Science,pp. 1-8,2000

[2]Kapovich M., and Millson J., ▪Linkages as functions ▪,Annals of Math Universality theorems for configuration spaces of planar linkages, 1973.

[3]Connelly R, and D. Demaine E.,CRC Handbook of Discrete and Computational Geometry,Segunda edición, Ontario, Canadá,2004.

[4] Carlos Alberto Olaguibert Segura, Universidad Autónoma Metropolitana unidad Azcapotzalco, Proyecto terminal, Ingeniería en Computación.“Generación de polígonos con triangulaciones ortogonales”, 2009.

CALENDARIO DE TRABAJO

Las actividades se realizarán durante el trimestre 10-O y 11-I en las ueas Proyecto Terminal I y Proyecto Terminal II, según el siguiente calendario:

RECURSOS

Los recursos a utilizar estarán disponibles por parte del alumno y son:

1. Open GL3.2
2. Compilador C++ builder
3. Sistema Operativo Ubunto 9.10

Transformación de configuraciones abiertas ortogonales en dos y tres dimensiones

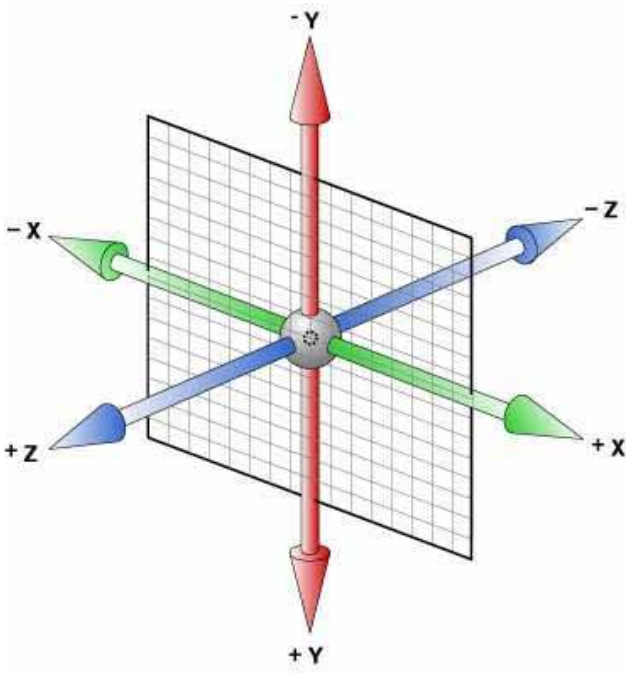
Resultados

Consideraciones.

La salida del programa generacion.cpp dado un número $1 \leqslant n \leqslant 20$ es una lista con la construcción de todas las posibles formas que puede adoptar un alambre de tamaño n.

Para su construcción se usan las siguientes consideraciones solo por comodidad.

$$\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 1$$
$$-\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 6$$
$$\begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 2$$
$$-\begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 5$$
$$\begin{bmatrix} z \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 3$$
$$-\begin{bmatrix} z \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 4$$



ya que toda configuración se puede representar como suma de vectores ortogonales, una configuración

$$\hat{C}_i = \sum_{i=1}^{n} v_i$$

así por ejemplo el vector

$$\hat{C}_{4i}\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\-1\\0\end{bmatrix}=\{1,2,1,6\}$$

Pruebas a las que se somete una configuración C o suma de vectores para el caso de las formas abiertas.

✔Prueba de números contiguos
✔Prueba de choque
✔Prueba de isomorfismo
✔Prueba de recorrido al reverso

Prueba de números contiguos: Se refiere a que dado un vector i, la sucesión de una configuración, en su posición i+1 tiene 5 grados de libertad, puesto que no puede regresar al mismo punto pues se translaparía.

Ej.

$\hat{C}_{4i}=\{1,2,3,1\}$ *válida*

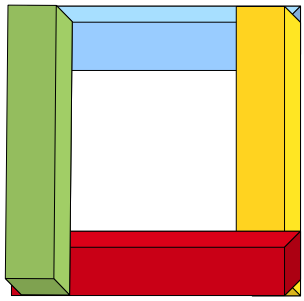
$\hat{C}_{4i}=\{1,2,3,4\}$ *inválida*

puesto que el opuesto de 1 es 4.

Prueba de choque : Se refiere a la prueba hecha a una configuración C, donde si se regresa a un vértice visitado anteriormente, se descarta tal vector y no pasa dicha prueba.

Ej

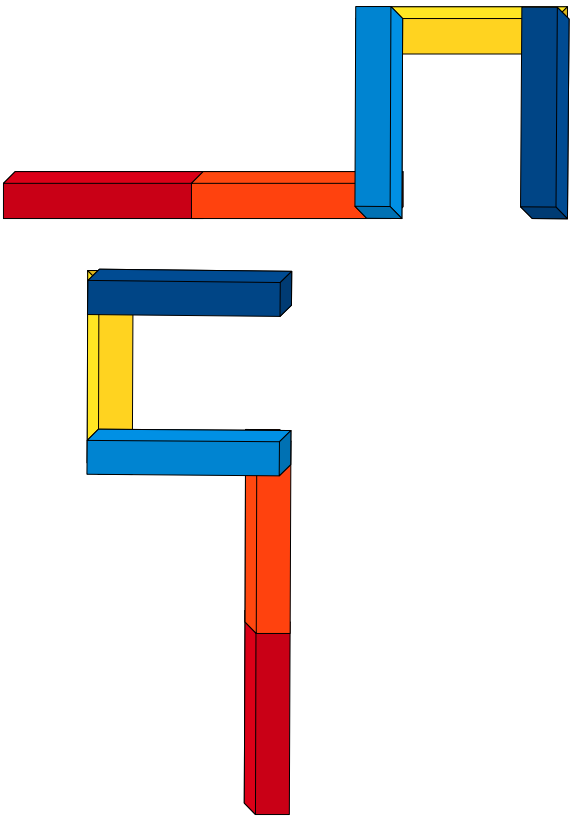
$\hat{C}_{4i}=\{1,2,6,5\}$ *inválida*



Prueba de isomorfismo: Prueba hecha a todas las posibles configuración (permutaciones) y comparándose entre ellas de acuerdo a su forma y simetria

Ej

$$isomorfo(\hat{C}_{4i})=\{1,1,2,1,6\} \quad isomorfo(\hat{C}_{4j})=\{1,1,5,1,2\}=verdadero$$



Prueba de isomorfismo: Prueba basada en la idea de que una misma configuración C se puede puede construir de cualquiera de sus dos extremos considerando como punto inicial bajo:

$$\hat{C}_n=\{v_1, v_{2,}\ldots, v_n\}=\lambda_n\{v_n, v_n-1\ldots v_1\}=\{v_n^q, v_n-1^q\ldots v_1^q\}$$

donde la propiedad q se refiere a el inverso visto al inicio del documento.

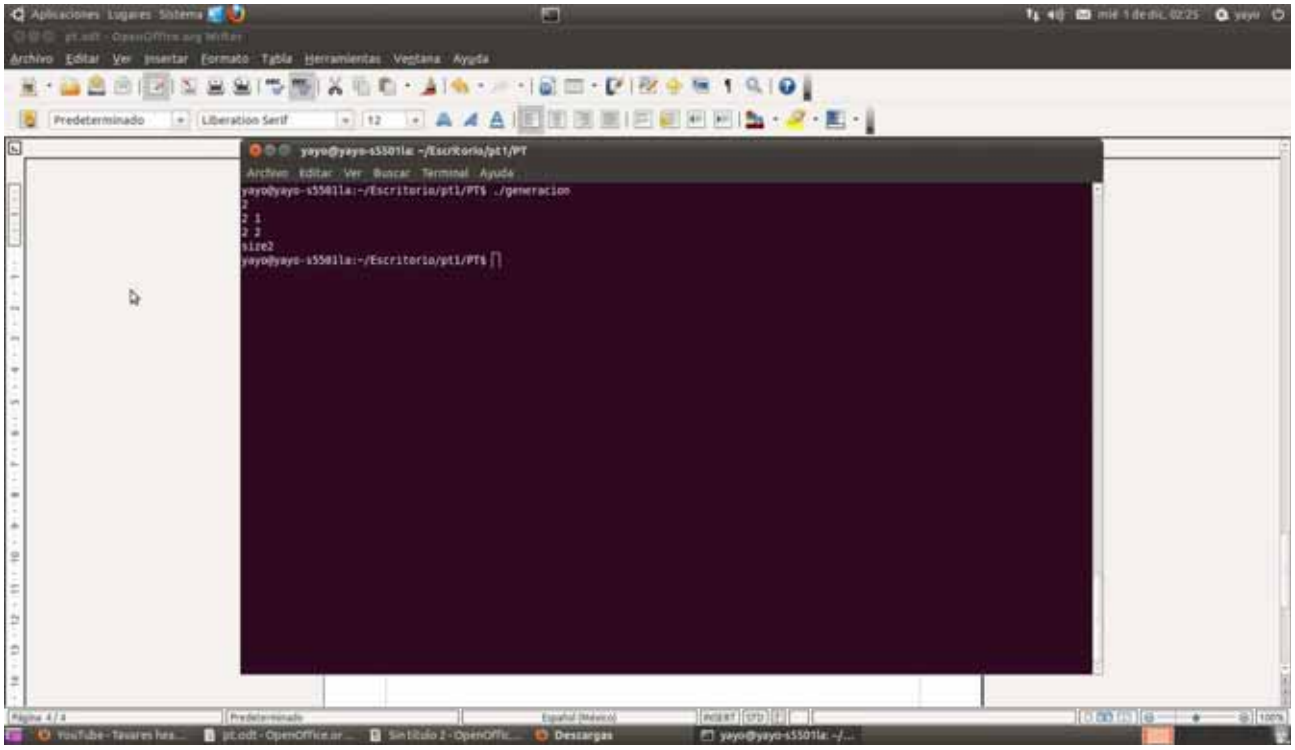
N	Inverso (N)
1	6
2	5
3	4
4	3
5	2
6	1

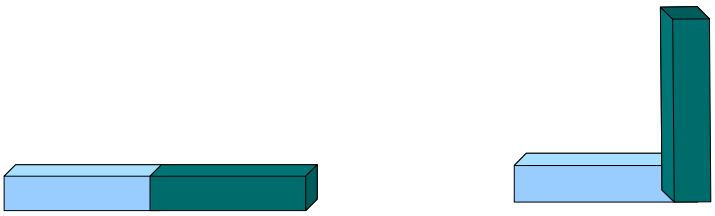
Se tomaron de esta forma, puesto que si se suma siempre es 7 y es más fácil su implementación.

Todas estas consideraciones se tomaron en cuenta al implementar el algoritmo descrito y se han marcado como funciones dentro del código anexo.

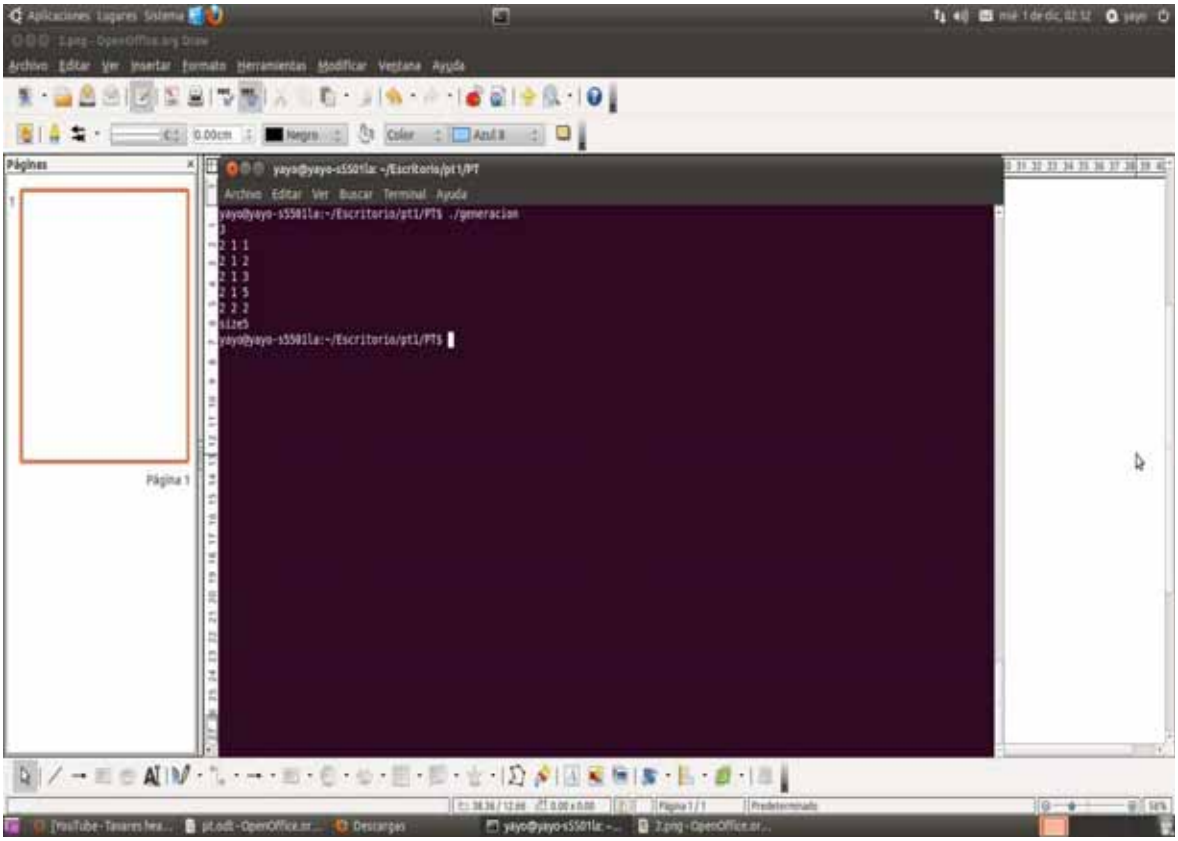
Salidas

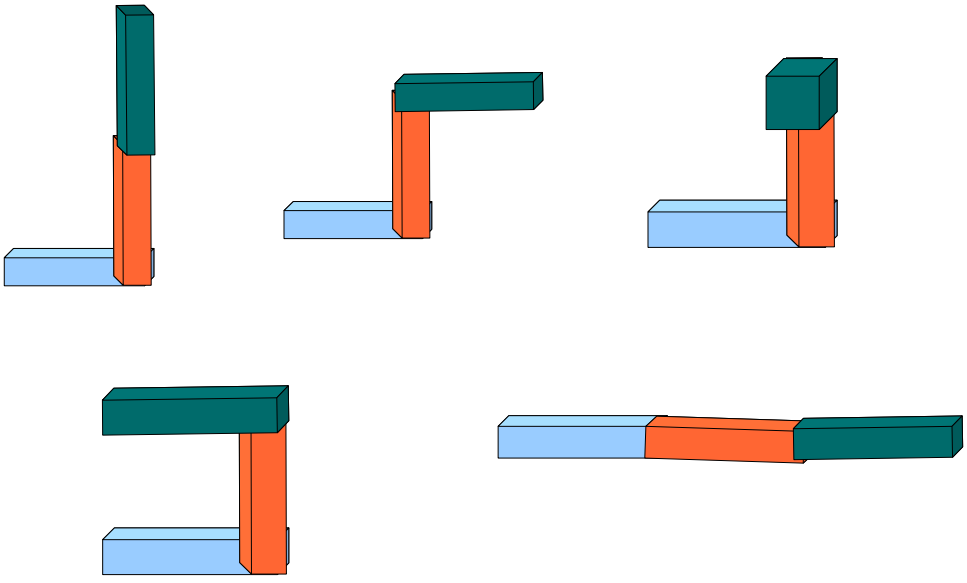
n=2


yayo@yayo-s5501la: ~/Escritorio/pt1/PT
Archivo Editar Ver Buscar Terminal Ayuda
yayo@yayo-s5501la:~/Escritorio/pt1/PT$ ./generacion
2
2 1
2 2
size2
yayo@yayo-s5501la:~/Escritorio/pt1/PT$

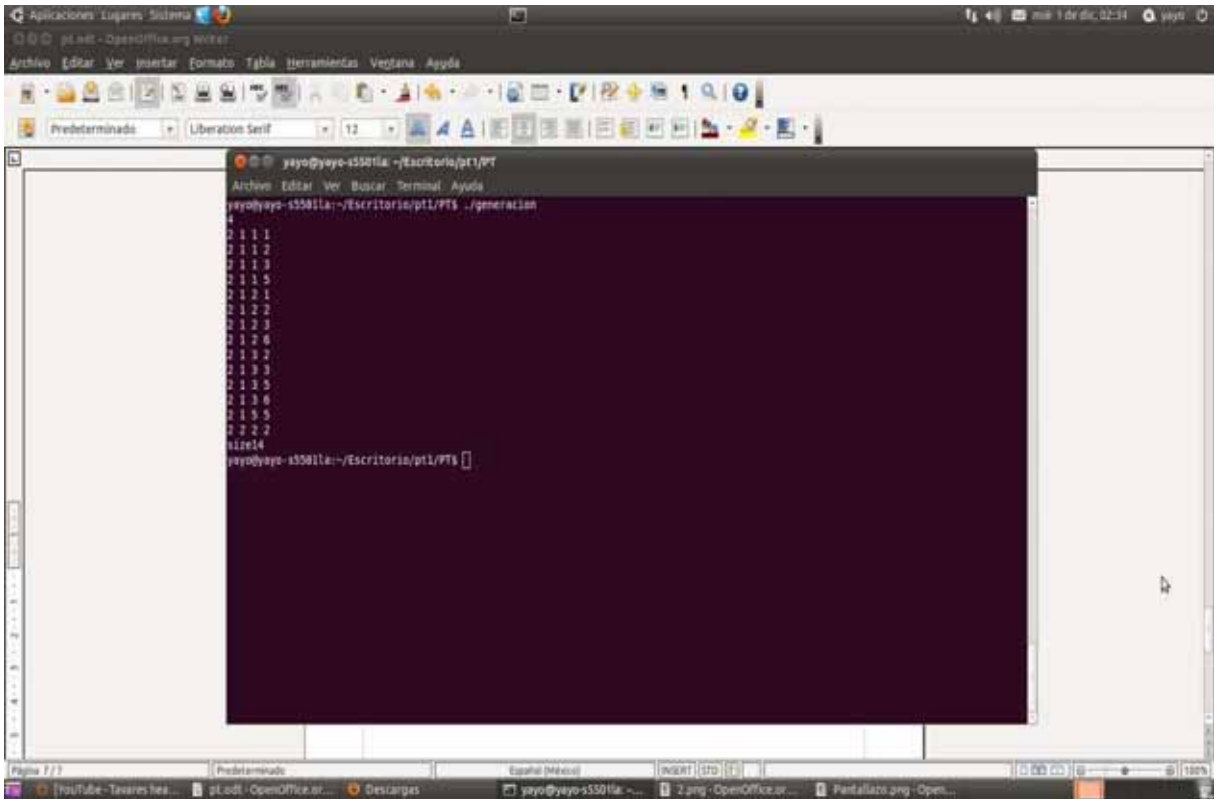


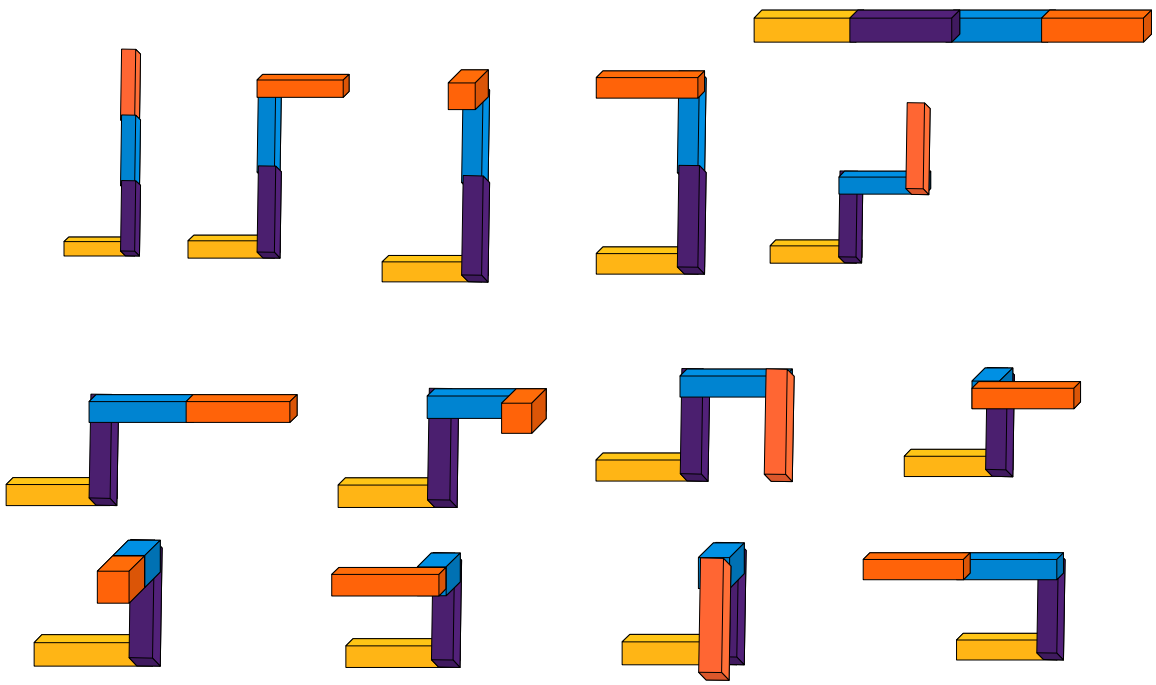
salidas
n=3


yayo@yayo-s5501la: ~/Escritoria/pt1/PT
Archivo Editar Ver Buscar Terminal Ayuda
yayo@yayo-s5501la:~/Escritoria/pt1/PT$ ./generacion
3
2 1 1
2 1 2
2 1 3
2 1 5
2 2 2
size5
yayo@yayo-s5501la:~/Escritoria/pt1/PT$



salidas
n=4


yayo@yayo-s550tla: ~/Escritorio/pt1/PT
Archivo Editar Ver Buscar Terminal Ayuda
yayo@yayo-s550tla:~/Escritorio/pt1/PT$ ./generacion
4
2 1 1 1
2 1 1 2
2 1 1 3
2 1 1 5
2 1 2 1
2 1 2 2
2 1 2 3
2 1 2 6
2 1 3 2
2 1 3 3
2 1 3 5
2 1 3 6
2 1 5 5
2 2 2 2
size14
yayo@yayo-s550tla:~/Escritorio/pt1/PT$



salidas

N	Generadas	Tiempo de ejecución
1	1	0m0.252s
2	2	0m0.291s
3	5	0m0.330s
4	14	0m0.373s
5	43	0m0.431s
6	135	0m0.455s
7	470	0m1.074s
8	1716	0m11.501s
9	6640	2m49.905s
10	27356	
11	120406	
12	577948	
13	3005333	
14	9440857	
15	17430935	
16		
17		
18		
19		
20		

Segunda parte
Configuraciones cerradas en dos y tres dimensiones

Consideraciones.

Una configuración cerrada $\hat{C}n_i$ es un vector con n enteros que especifica la sucesión de vectores unitarios ortogonales (suma de vectores) dando como resultado la igualdad de la posición de donde se partió a donde se llegó osea entre el punto inicial y el punto final .

Para su construcción se usan las siguientes consideraciones solo por comodidad.

$$\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 1$$

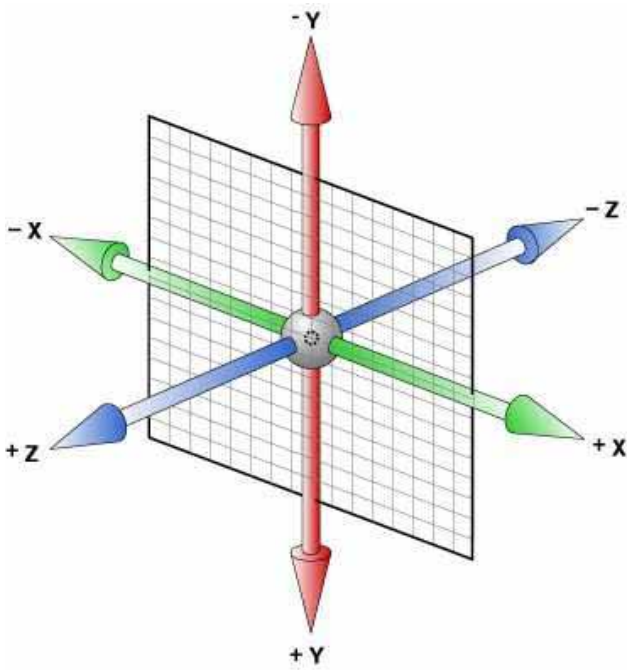
$$-\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 6$$

$$\begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 2$$

$$-\begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = 5$$

$$\begin{bmatrix} z \\ 0 \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 3$$

$$-\begin{bmatrix} z \\ 0 \\ 0 \end{bmatrix} \rightarrow -\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 4$$



ya que toda configuración se puede representar como suma de vectores ortogonales, una configuración

$$\hat{C}_i = \sum_{i=1}^{n} v_i$$

así por ejemplo el vector

$$\hat{C}_{4i}\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}1\\0\\0\end{bmatrix}+\begin{bmatrix}0\\-1\\0\end{bmatrix}=\{1,2,1,6\}$$

Para generar todas las posibles formas cerradas diferentes es necesario partir de una o m·s formas iniciales, las cuales son el origen de las restantes.

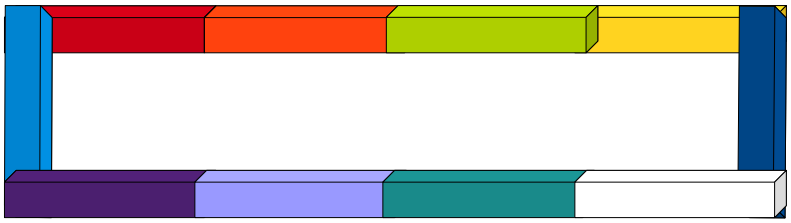
El método utilizado para generar las primitivas en dos dimensiones (cuadrados o rect·ngulos) fue el siguiente.

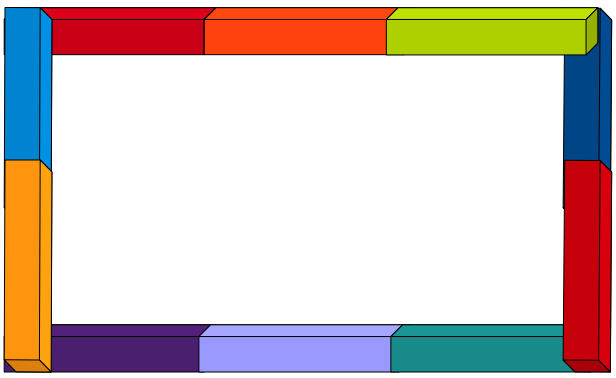
◆Dado n dividir entre dos para saber el número de enlaces a ocupar para dos de los lados diferentes del rect·ngulos.
◆Encontrar los valores para los cuales la suma da n/2
◆Llamemos a cada una de estas figuras primitiva, y para cada primitiva y para cada una de sus configuraciones hijas, solo se realiza un cambio de dos enlaces (doblez).

Así para n=10 tenemos

10/2=5

4	1
3	2





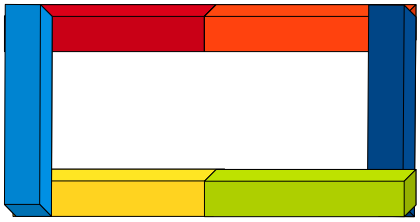
Pruebas a las que se somete una configuración cerrada .

✔Llenar una lista de movimientos v·lidos.
✔Llenas una lista de movimientos inv·lidos o tabú.
✔Prueba de choque
✔Realizar doblez para un ·ngulo formado por dos enlaces
✔Realizar doblez a una configuración cerrada completa o mapa para construir las versiones 3D a partir de las primitivas 2D.

Llenar una lista de movimientos válidos:

Dada una configuración $\hat{Cn}_i$ se prueban los cambios contiguos de los elementos del vector y se enlistan solo los movimientos v·lidos. Dado que es una forma cerrada hay que considerar al vector como una cola cerrada

Ejemplo.



$C_n = 2,2,6,5,5,1$

Posibles cambios	Posición
$2 \rightarrow 6$	2 y 3
$6 \rightarrow 5$	3 y 4
$5 \rightarrow 1$	5 y 6
$1 \rightarrow 2$	6 y 1

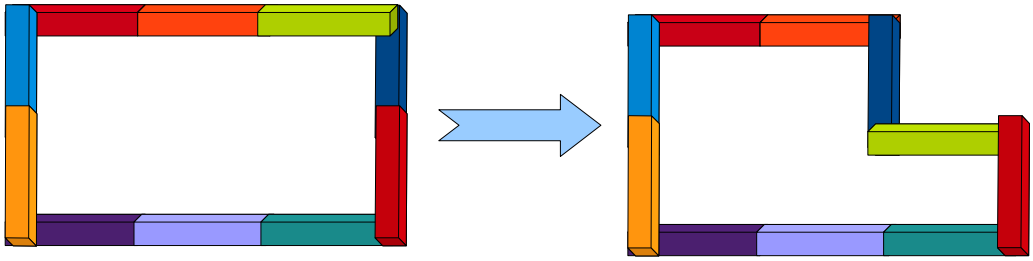
...sta tabla representa todos los posibles dobleces que se pueden hacer a la forma para ir generando las dem·s configuraciones usando el mismo número de enlaces.

La naturaleza del algoritmo como se observa es recursiva por lo que en cada iteración se construyen nuevas formas solo al realizar un doblez.

Llenar una lista de movimientos inválidos o tabú:

Nos permite acotar nuestro espacio de búsqueda reduciendo el tiempo de ejecución del algoritmo, se basa en la idea de descartar visitas inútiles antes visitadas y nos ayuda para descartar los elementos repetidos de la lista v·lida de movimientos de las configuraciones.

Ejemplo. $C_n = 2,2,2,6,6,5,5,5,1,1$ $C_n = 2,2,6,2,6,5,5,5,1,1$

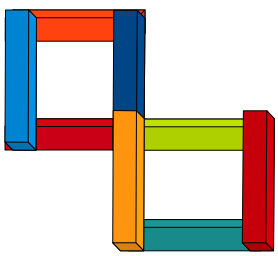


La nueva figura debe descartar el movimiento anterior, así una lista de movimientos inv·lidos de la segunda configuración serían las posiciones 3 y 4, por lo que la lista v·lida sería $5 \to 1$ $1 \to 2$ $2 \to 6$ $6 \to 2\,(descartada)$ $6 \to 5\,(inválida)$

Prueba de choque

Se refiere a la prueba hecha a una configuración C, donde si se regresa a un vértice visitado anteriormente, se descarta tal vector y no pasa dicha prueba.

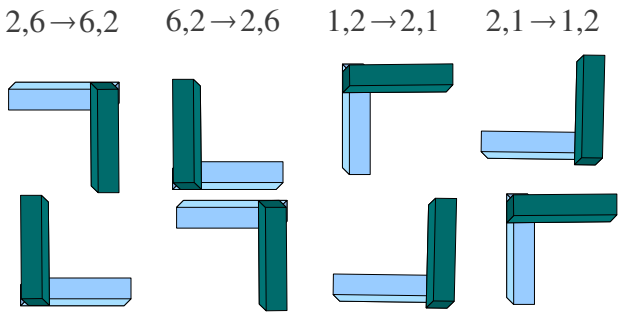
Ejemplo de choque



Realizar doblez para un ángulo formado por dos enlaces.

Estrategia para modificar configuraciones antiguas y generar nuevas

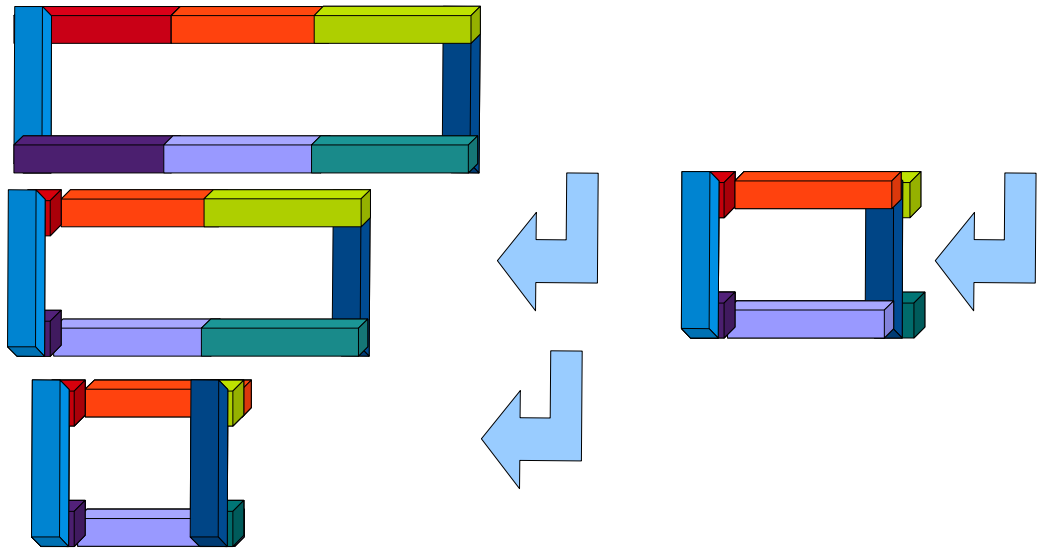
Se basa en la idea de ir doblando la figura actual para formar otra, un solo doblez por iteración, así los dobleces permitidos son:



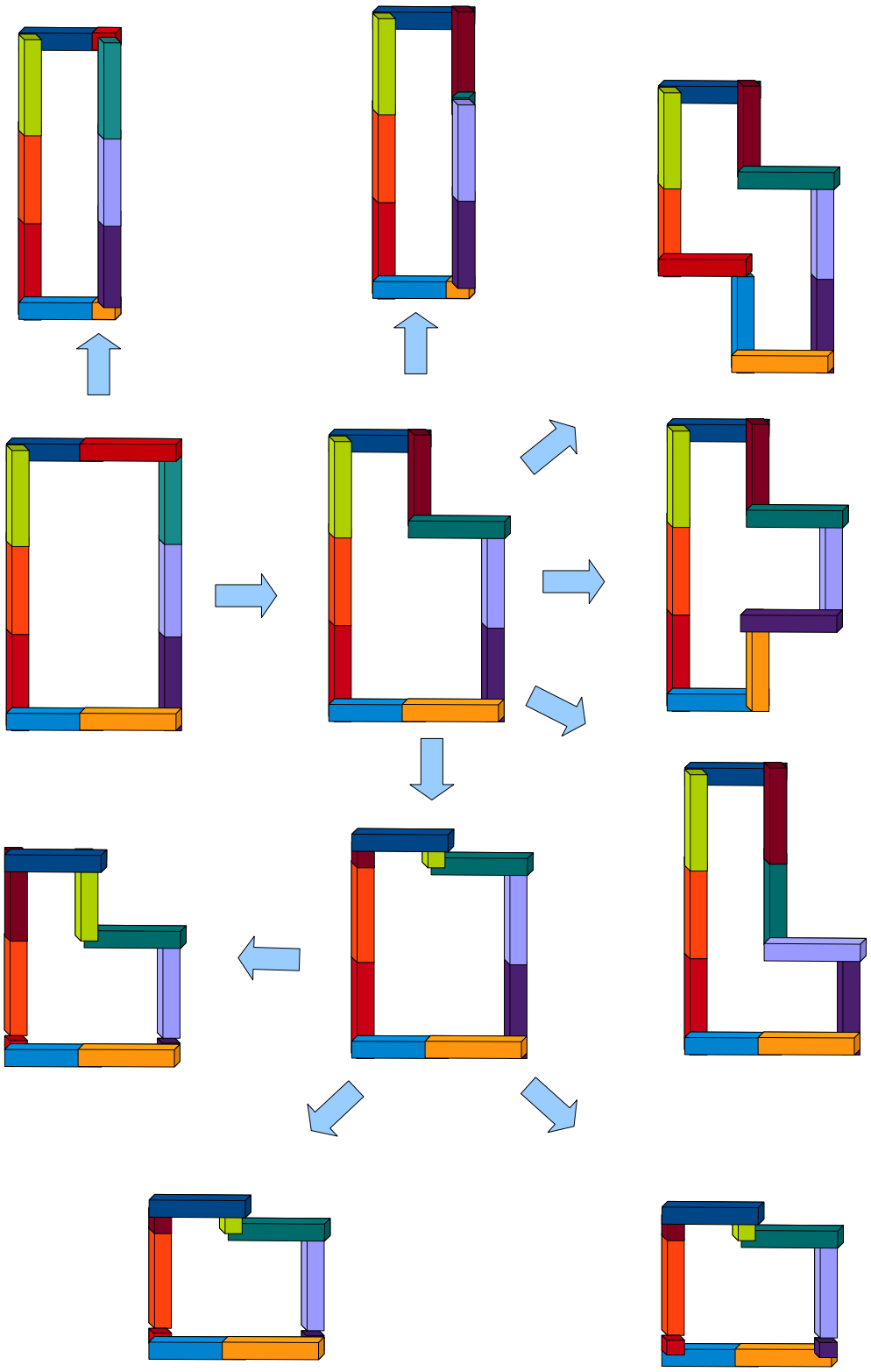
Realizar doblez a una configuración cerrada completa o mapa para construir las versiones 3D a partir de las primitivas 2D.

La idea detrás de esta estrategia es formar las configuraciones en 3D que complementan a las anteriores, basada en la idea de doblar pestañas de las formas generadas en el eje z.

Ejemplo.



Simulación Gráfica del Algoritmo completo (primera fase 2D) para n=10



Resultados

N	Formas
2	0
4	1
6	2
8	9
10	38
12	211
14	1421
16	5288
18	9308
20	16355

Conclusiones

El número de configuraciones distintas obtenidas tanto para las formas abiertas y las cerradas crece muy r·pido. Existe una notoria diferencia en la cardinalidad de los distintos conjuntos para un mismo número dado, mayor para las formas abiertas debido a la naturaleza del problema puesto que es m·s facil encontrar un mayor número de combinaciones de formas libres que cerradas.

Respecto al tiempo de ejecución necesario para encontrar las formas cerradas fue menor en todos los casos por el tipo de búsqueda, el espacio de búsqueda, las técnicas ocupadas, la naturaleza del problema hicieron la búsqueda m·s acotada y por lo tanto m·s r·pida.

Para el caso de las formas abiertas habían m·s aspectos a considerar (choques, isomorfismo, repeticiones) también habían búsqueda adicionales por que para números aun pequeños muchos de los casos eran configuraciones isomorfas.

El implementar memoria para evitar ciertas búsquedas fue una buena forma de mejorar el algoritmo, mejoró en las últimas pruebas por casi la mitad del tiempo.

Para mejorar el algoritmo se requieren buenas ideas y heurísticas, ya que el enfoque glotón ocupado con mayor peso hace el algoritmo lento, aunque era de esperarse puesto las pruebas a las que se someten las configuraciones.

En la exploración de las formas abiertas se partía del alambre original de tamaño n y se iba doblando hasta encontrar camino sn choque, esta solución hizo que la exploración fuera exahustiva mientras que en las formas cerradas con recursividad se iban formando las nuevas formas y no existian tantas discriminaciones de configuraciones porque se iban generando naturalmente, de ahi la gran diferencia en tiempo de ejcución del problema para este último caso.

Bibliografía

[1]Demaine E.D,"Folding and Unfolding Linkages,Paper and Polyhedra",In Proceedings of the Japan Conference on Discrete and Computational Geometry, Lecture Notes in Computer Science,pp. 1-8,2000

[2] Sedgewick Robert "Algoritmos en c++" Addison-Wesley, 1990

[3] Rosen K. "Discrete mathematics and its applications, 6th " McGrawHill,1999