Implementación del juego Othello 3D

Trabajo final para la asignatura Introducción a la Programación I
Primer Cuatrimestre del Primer Año de la Carrera Ingeniería de Sistemas de la Facultad de Ciencias Exactas, UNCPBA.

Autores: Gastón Alejandro Zemma
Gabriel Alberto Álvarez

e-mail: zemma@lettera.net
gaa.hd@lettera.net

Profesores: M.Sc. Jane Pryor
Ing. Hernán Cobo

Departamento de Sistemas, Fac. de Ciencias Exactas,
Universidad Nacional del Centro de la Pcia. De Buenos Aires

RESUMEN:

El presente trabajo es la realización del juego othello como trabajo practico para la cátedra de Introducción a la Programación I del primer cuatrimestre del primer año de la carrera Ingeniería de Sistemas, pero con iniciativas propias como ser graficas en tres dimensiones y opción de jugar contra la PC por ejemplo. Nuestra solución fue implementada (como era requerido por la cátedra) en Turbo Pascal, pero adicionalmente incluimos rutinas en el lenguaje ensamblador que éste trae incorporado ya fue necesaria alta velocidad debido a las graficas tridimensionales.

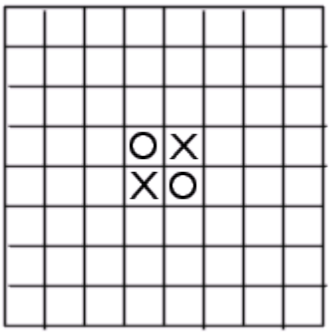
Como principal conclusión destacamos que utilizando las técnicas aprendidas en la asignatura se consiguió implementar una solución satisfactoria al problema planteado.

INTRODUCCION:

Este trabajo ha sido realizado para la asignatura Introducción a la Programación I, la cual se cursa en el primer cuatrimestre del primer año de la carrera Ingeniería de Sistemas de la Universidad Nacional del Centro de la Provincia de Buenos Aires.
El trabajo practico final de dicha asignatura consistió en la implementación del juego Othello, también conocido como Reversi.

Enunciado del Problema

La siguiente es la descripción provista por la cátedra para la implementación del juego:



"Consiste en un juego de tablero para dos participantes, los cuales deben ir agregando de a una ficha de forma alternada. El tablero es de 8x8 con la siguiente disposición inicial:

Figura 1. Posición inicial del tablero en el juego Othello

Cuando un jugador agrega una ficha todas las del jugador contrario que queden encerradas entre dos fichas del primero se invertirán, es decir pasan a ser del jugador actual y cambian de tipo de ficha. El encierro puede ser horizontal, vertical o en diagonal."

En consecuencia, nuestro objetivo fue desarrollar un juego similar a Othello, con algunas reglas específicas. para la materia Introducción a la Programación I. La principal regla que difiere de las originales es que se puede ubicar una ficha de cualquier jugador en cualquier posición vacía del tablero.
Además, nos propusimos dotar a la aplicación de características no solicitadas por la cátedra. Las principales características incorporadas fueron una interfaz gráfica tridimensional y la posibilidad de jugar contra la máquina

Implementación del Juego

El trabajo descrito fue programado casi en su totalidad en Turbo Pascal 7 de Borland. Sin embargo, algunas rutinas gráficas fueron implementadas en lenguaje ensamblador debido a la velocidad requerida para su ejecución.
En la figura 2 se puede ver una captura de la interfaz tridimensional implementada

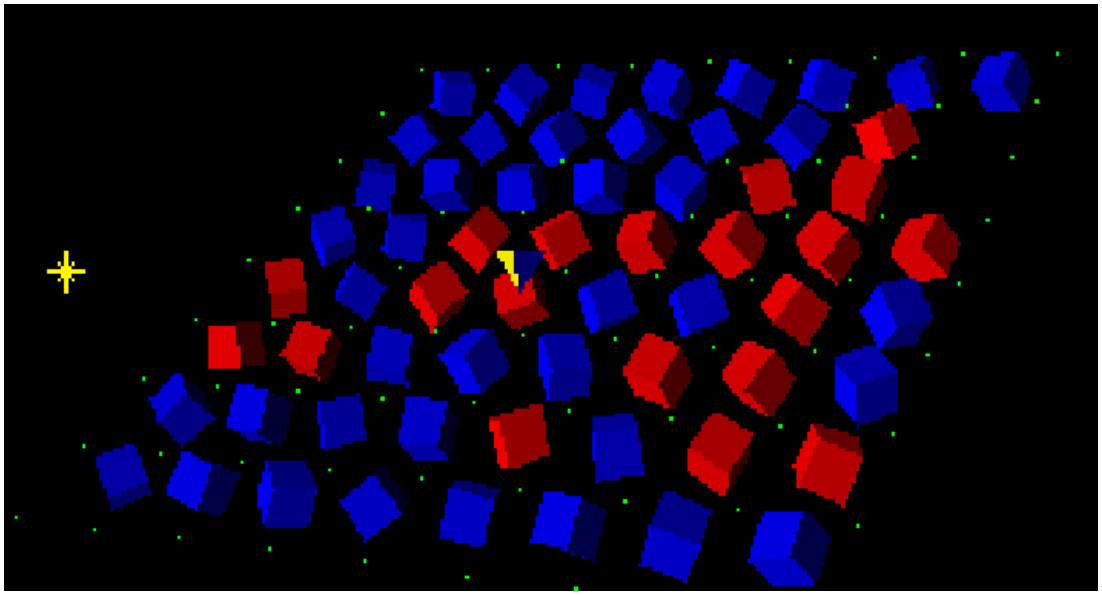


Figura 2 – Visualización de la interfaz a usuario del juego implementando

OTHELLO:

En esta sección se describe cómo se implementó el problema propuesto. La sección 2.1 explica la estructura de la solución mientras que la sección 2.3 contiene detalles específicos y código fuente sobre la implementación de la solución. El código fuente completo se encuentra anexo.

Estructura de la Solución:

-

Primeramente se decidió separar el problema en dos, para atacarlo de una manera mas lógica y distribuir el trabajo (hecho por dos personas) de una forma razonable.
Uno de los integrantes programaría el juego propiamente dicho, y el otro trabajaría en el motor de gráficos tridimensionales como se describe en las subsecciones I y II.

Sobre el Motor de tres dimensiones:

Lo primero fue idear como crear la ilusión de tres dimensiones. Para esto se imagino que cada punto en el espacio estaba dentro del monitor y se calculo por trigonometría donde seria visto por el observador (ubicado a una distancia aproximada) el punto en que se proyectara sobre el “vidrio” del monitor.

Se decidió usar el modo de video conocido como 13h (320x200 pixeles con 256) por ser este el mas fácil de manejar ya que la pantalla entera entra en 64kb.

El motor de gráficos se pensó de una manera que se pueda adaptar fácilmente a otras aplicaciones que requieran graficas similares.

El tablero de juego se implemento principalmente como un objeto de tres dimensiones para reutilizar de una forma mas conveniente el código, pero luego se debió hacer que sea un objeto diferente (con mas cantidad de vértices que los objetos comunes), ya que la memoria no alcanzaría para hacer todos los objetos con un máximo de 64 vértices. A pesar de esto los procedimientos del tablero presentan una gran similitud con los de los objetos comunes.

Se introdujo sombreado plano con una fuente de luz (que se puede mover para apreciar el efecto que genera).

Las fichas tridimensionales (que son cubos) están declaradas como un puntero a un arreglo de punteros a objetos tridimensionales (1..64) y tienen coherencia con las fichas del tablero matriz (en donde se calculan las jugadas, etc.) mediante el numero de índice del arreglo que esta copiado en la posición del tablero matriz que debe ir (en el tablero matriz también esta indicado a que jugador pertenece esa posición).

El cursor que indica la posición y el jugador que debe mover es también un objeto tridimensional y es manejado mediante los mismos procedimientos que las fichas (código bastante reutilizable).

Los colores diversos para los sombreados de las fichas, etc. se logran redefiniendo los 256 colores de la paleta en modo 13h.

La implementación del mouse se logro mediante la inclusión de una librería externa.

Sobre el programa del juego othello en si:

-

Lo primero que se planteo fue un diagrama en bloque con los problemas que teníamos para realizar el juego (llamado estrategia).

Luego se fueron construyendo los procedimientos en relación con los problemas citados previamente .

Un detalle a destacar fue que se busco una estructuración que permitiera minimizar el código redundante y facilitar la lectura, por ejemplo el procedimiento que cambia las fichas del jugador contrario en las de uno.

El programa presenta una modalidad de juego de una persona con otra o una persona contra la PC.

Detalles específicos considerados de interés

Un punto interesante a tener en cuenta es la unión del juego con el motor de tres dimensiones, esto principalmente se logro añadiendo al tablero matriz (citado arriba, que originalmente solo contenía el carácter que representaba al jugador dueño de esa posición) el numero de índice del arreglo de fichas tridimensionales. Luego un procedimiento llamado tras cada cambio se encarga de actualizar el tablero de 3D (crear las fichas nuevas necesarias, cambiar el color de las que variaros, etc.)

Otro detalle es el dato devuelto por la función que calcula la mejor posición en el modo de jugar contra la PC. La posición es devuelta como un solo valor entero; la coordenada X esta dada multiplicada por 10 y se le suma la coordenada de Y, esto fue necesario ya que una función solo puede devolver un valor.

Un punto a tener en cuenta son las rutinas en ensamblador utilizadas para "dibujar" de un modo mas rápido, estas son bastante elementales y pueden ser optimizadas. Sin embargo, nuestro conocimiento no es suficiente para realizar estas optimizaciones; y en su estado actual las rutinas cumplen su cometido. En parte fueron tomadas de ejemplos de tutoriales de programación y fueron modificadas con el fin de trabajar en nuestro proyecto.

Definiciones de los tipos de datos usados por los objetos tridimensionales:

```
{ Tipo cara (polígono) }
type TCara =    record
                        Vertices : integer;  { Número de vértices en el polígono }
                        Vertice : array[1..MaxVerticesCara] of integer;
                                { Indices correspondientes a cada vértice }
                        normal : TVertice;  { vector normal al plano de la cara }
                        color : byte;       { color de la cara }
                        coloro: byte;
                        Z : double;
                end;

{ Tipo Objeto Tridimensional }
TObjeto3D =     record
                        NVertices : integer;  { Número de vértices }
                        NCaras : integer;     { Número de caras }
                        ox, oy, oz : double;  { Coordenadas del origen del objeto }
                        Vertice : array[1..MaxVertices] of TVertice; { Vértices }
                        Cara : array[1..MaxCaras] of TCara;         { Caras }
                end;

{ Tipo Vértice }
type TVertice = record
                        x, y, z : double;  { Coordenadas del vértice }
                        x2d, y2d : integer; { Coordenadas de la proyección en 2D }
                end;
```

CONCLUSIONES:

La principal conclusión es que utilizando las técnicas aprendidas en la asignatura se consiguió implementar una solución satisfactoria al problema planteado.

Entre las dificultades que se nos presentaron queremos destacar:

- - Limitada memoria que permite manejar el Turbo Pascal 7 de Borland se debieron usar algunas variables globales por ello.
  Debido a la limitada capacidad para manejar la memoria del compilador de Pascal de Borland se debió recurrir al uso de algunas técnicas que utilizaran menos memoria, aunque esto perjudico en parte la correcta estructuración de la solución
- - Velocidad de la BGI.
  La BGI (Borland Graphics Interfase) que trae incorporada el ambiente Pascal, no alcanzaba para los requerimientos de velocidad del juego por lo que se debió recurrir al lenguaje ensamblador; lo que fue un problema en si ya que no conocíamos dicho lenguaje.
- - Unión de el juego con el motor en tres dimensiones.
  La idea original de dividir las tareas de programación entre los dos autores trajo consigo algunos problemas sobre la "fusión" de las dos partes. Esto se podría haber evitado desde un principio fijando pautas que permitieran un trabajo organizado.
- - Limitada resolución del modo grafico.
  El modo de video 13h en el que trabaja el juego tiene como desventaja que presenta una resolución de 320x200 píxeles y solo 256 colores diferentes simultáneos, y teniendo en cuenta que el sombreado requiere muchos matices diferentes 256 colores pueden llegar a ser pocos. Sin embargo se utilizo igualmente este modo de video debido a la facilidad que presenta su manejo, esto es porque toda la pantalla (64 Kbytes) entra entera en un segmento de memoria.

Entre las experiencia que nos aportó la realización de este trabajo, queremos mencionar:

- - Notamos la eficacia de una buena estructuración para no desorientarse en un programa de varias líneas, ya que cuando el programa se fue haciendo más complejo se volvió fundamental mantenerlo ordenado.
- - Aprendimos bastante sobre la creación de gráficos por PC, especialmente en el modo de video 13h.
- - Vimos la necesidad de administrar bien la memoria, puesto que al trabajar con tipos de datos medianamente grandes la cantidad de memoria se volvió un problema.
- - Mediante la necesidad de implementar algunas rutinas en lenguaje ensamblador vimos que este trabajaba rápido, pero esto trajo consigo mayor complejidad al programa.

AGRADECIMIENTOS:

Queremos agradecer a las siguientes personas por el apoyo prestado:
Jane Pryor, Hernán Cobo, Álvaro Ortigosa, y Fernando Rossi.

REFERENCIAS:

- Serie de tutoriales de programación de FAC, aka Alfonso Alba; http://galia.fc.uaslp.mx/~ganzo/prog/tutorial.html, Internet.
- The PC Games Programming Encyclopedia (PCGPE), http://teeri.oulu.fi/pub/msdos/programming/gpe
- O'Brien, Stephen, Turbo Pascal 7 Manual de Referencia, McGraw-Hill Inc., Madrid España (1993).
- Dewdney A. K., "Juegos de Ordenador", Investigación y Ciencia (edición en español de Scientific American) Nro. 96, Prensa Científica, Barcelona España (1984).