UNIVERSIDAD HISPANOAMERICANA

Sede xxxxx

Arquitectura de computadoras II

Profesor: xxxxx

Proyecto Libre xxxxx en lenguaje ensamblador

Alumnos:

xxxxx

Abril, 2011

Tabla de contenido

Descripción del programa y su funcionamiento 3

Trama del juego: 3

Funcionamiento del Juego: 3

Funcionamiento del programa 4

Fuente de código: 4

Variables: 4

Procedimientos: 5

Etiquetas principales: 8

Anexos: 10

Descripción del programa y su funcionamiento

## Trama del juego:

En nuestra galaxia vecina, Andrómeda, han existido unos seres llamados los asteríscobots desde antes que el planeta Tierra diera a luz a sus habitantes.

Estos seres son muy inteligentes pero corren con la desventaja de que solo actúan bajo sus instintos de supervivencia.

Desde hace algún tiempo han realizado visitas pacíficas a la tierra sin que los Terrestres sospechen nada pero en una de esa visitas chocaron contra uno de los satélites que orbitan la tierra y debido a sus instintos ellos contra atacaron siendo descubiertos así por los habitantes de la tierra.

Y ahora los terrestres tienen que defender su territorio …………….

## Funcionamiento del Juego:

El programa xxxxx es un juego interactivo que trata de una nave espacial que como objetivo tiene que eliminar las barricadas de asteriscos de los enemigos antes que ellos lleguen a la posición de la nave y la destrocen.

Para ello la nave cuenta con un disparador que se activa con la tecla espacio, así como también puede desplazarse por la pantalla utilizando para ello las flechas de dirección del teclado o en su defecto puede salir con la tecla ESC.

………

Y como es de suponer, cada vez que no queden asteriscos en la pantalla se pasará de nivel o en su defecto se ganará el juego (que volverá a comenzar).

Funcionamiento del programa

## Fuente de código:

El juego “xxxxxxx” está desarrollado en lenguaje ensamblador del procesador 8086 y para la implementación del código no utiliza macros, o cualquier otro tipo de archivo externo, si no que todo está escrito en el programa principal, además de esto, para su funcionamiento es necesario copiar los archivos **Nivel1, nivel2 y nivel3 en la raíz C:\** esto es porque el juego carga los niveles desde esos archivos y por lo tanto no deben ser modificados.

## Variables:

En el juego se utiliza una variedad de variables, algunas utilizadas para ciertas funciones específicas y otras que se utilizan en la mayor parte del programa.

Se describe cada una con su función principal

|  |  |  |
| --- | --- | --- |
| **Nombre de la variable** | **Tipo de dato** | **Descripción** |
| Aux | Db | Variable multipropósito |
| Aux2 | Db | Variable multipropósito |
| Buffer | Db | Almacena el caracter actual leído en el archivo de texto. |
| Cant\_lineas | Db | Almacena la cantidad de líneas de asteriscos (enemigos) por nivel. |
| Carac | Db | Utilizada principalmente para almacenar un carácter capturado en pantalla |
| Cola\_navex | Db | Posición horizontal donde se encuentra la parte inferior de la nave. |
| Cola\_navey | Db | Posición vertical donde se encuentra la parte inferior de la nave |
| Cuerpo\_navex | Db | Posición horizontal donde se encuentra la parte media de la nave. |
| Cuerpo\_navey | Db | Posición vertical donde se encuentra la parte media de la nave |
| Der | Equ | Mantiene el valor de la flecha de dirección Derecha del teclado |
| Err\_msg | Db | Almacena un mensaje de error en caso de que no se pueda abrir el archivo de texto |
| Esquina\_aux | Db | Utilizada para almacenar la posición vertical inferior de los asteriscos para que puedan ser “bajados” una línea (por lo mismo es auxiliar, ya que cada línea que se baje el ataque se descuenta de esta para no perder la posición original) |
| Esquina\_inf\_ataque | Db | Contiene la posición inferior vertical de la última línea de asteriscos |
| Felicidades | Db | Contiene el mensaje para cuando se ha superado por completo el juego |
| Filename | Db | Contiene el nombre y ruta del archivo de texto utilizado para inicializar los niveles |
| Game\_over | Db | Contiene el mensaje para cuando se ha perdido un nivel |
| Instrucciones1, Instrucciones2, Instrucciones3, Instrucciones4 | Db | Guardan cadenas de caracteres con la información que el usuario ocupa saber para jugar |
| Izq | Equ | Mantiene el valor de la flecha de dirección izquierda del teclado |
| Juego\_ganado | Db | Contiene el mensaje para cuando se ha superado un nivel |
| Nave1, nave2 y nave3 | Db | Contienen el dibujo de la nave (cada una de ellas con un sector diferente de la nave) |
| Niv\_act | Db | Nivel en el que se está jugando |
| Nivel1, nivel2 y nivel3 | Db | Guardan el nombre del archivo que se utilizará para los diferentes niveles |
| Punta\_navex | Db | Posición horizontal donde se encuentra la parte superior de la nave. |
| Punta\_navey | Db | Posición vertical donde se encuentra la parte superior de la nave |
| Teclazos | Db | Almacena la cantidad de teclas presionadas por el usuario |
| X | Db | Representa la posición horizontal del cursor |
| Y | Db | Representa la posición vertical del cursor |

## Procedimientos:

A continuación se detallan los procedimientos y rutinas más importantes utilizadas para la creación del mismo. En esta sección sólo se describe la función principal del procedimiento, para ver la implementación del mismo se deberá consultar el programa fuente.

|  |  |
| --- | --- |
| **Nombre** | **Descripcion** |
| Ajusta\_nivel | Se encarga de seleccionar el archivo dependiendo del nivel que se muestra en la variable niv\_act, manda a leer el archivo y ajusta las variables de cant\_lineas y esquina\_inf\_ataque para su uso futuro |
| ajusta\_nivel proc  cmp niv\_act,01h  je ajusta\_nivel1  cmp niv\_act,02h  je ajusta\_nivel2  cmp Niv\_act,03h  je ajusta\_nivel3    mov niv\_act,1  jmp inicio  ajusta\_nivel1:  mov esquina\_inf\_ataque , 3  mov cant\_lineas, 03h  mov al, 0 ;read only  mov dx, offset nivel1  jmp lee  ajusta\_nivel2:    mov cant\_lineas ,04h  MOV esquina\_inf\_ataque , 04H  ;open existing file  mov al, 0 ;read only  mov dx, offset nivel2  jmp lee  ajusta\_nivel3:  mov cant\_lineas,05H  MOV esquina\_inf\_ataque , 05H  ;open existing file  mov al, 0 ;read only  mov dx, offset nivel3  jmp lee  ret  endp | |
| Bajar\_ataque | Una de los más importantes, es el encargado de bajar un renglón cada línea de asteriscos utilizados como enemigos |
| BAJAR\_ATAQUE PROC  mov dl,cant\_lineas  .  .  .  .  .  RET  ENDP | |
| Bajar\_linea | Utilizada para bajar un renglón de asteriscos de la línea de ataque |
| bajar\_linea proc  mov x,-1  mov aux2, 30d  . | |

## Etiquetas principales:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Procedimiento donde es usada** | **Descripcion** |
| ajusta\_nivel1:, ajusta\_nivel2:, ajusta\_nivel3: | Ajusta\_nivel | Encargadas de ajustar el nivel correspondiente (variables) y mandar a leer el archivo |
| ataque: | Velocidad | Manda a llamar al procedimiento bajar ataque y devuelve los teclazos a 0 |
| bajar: | bajar\_linea | Utilizada para bajar un caracter a la línea siguiente y borrarlo de la anterior |
| bjlin: | bajar ataque | Esta se utiliza para bajar una línea de asteriscos a la línea siguiente |
| dispara: | N/A | Llama a los procedimientos subir bala, incrementa los teclazos y verifica la velocidad |

Descripción las áreas del juego



1- Representa el área de la ofensiva, la cual estará compuesta por asteriscos.

2- En esta zona se muestra el nivel actual.

3- Nave.

4- Zona de desplazamiento de la nave u “ofensiva”.

5- Línea limítrofe para la nave.

6- Las instrucciones del juego.

Anexos:

**Juego completo “xxxxx”**

.Model medium

.stack 100h

.data

; sizes dw 0

instrucciones1 db 'Dispara -->Espacio$'

instrucciones2 db 'Izquierda -->Tecla de direccion <-- $'

.code

inicio:

mov ax,@data

mov ds,ax

………………………..

finalizar proc

mov Ah,4ch

int 21h

ret

endp