Introducción

Muy bien, apartir de ahora comenzaremos un largo camino hacia convertirnos en programadores de juegos, primero debes saber que la programación de juegos es una ciencia muy compleja, donde se relacionan gran cantidad de factores, por ejemplo programación básica, programación grafica, matemática, física, y un largo etc., por lo que no esperes que de la noche a la mañana hagas el próximo súper éxito de ventas. Ten en cuenta que a lo largo de este manual nos remontaremos a los años 80' hasta llegar mas o menos a mediados de la decada del 90' (jejje pensarás que estoy loco) lo que quiero decir es que empezaremos ha hacer juegos super sencillos, ya que de esta forma iremos atravesando los procesos por los que pasaron los programadores de aquella epoca.

El presente manual tiene como objetivo enseñar la programación en 2d, quizas en otra oportunidad creare un manual en 3d, pero por ahora con las 2d tenemos bastante ;) .

haaa me olvidaba nunca programe en blitz basic asi que este será un aprendizaje también para mi, así que encargo a todos aquellos que lleguen a leer este manual que por favor, me hagan llegar todas las sugerencias, correcciones, o hasta cosas que le parezcan agregar a ajeansalle@gmail.com .

PARTE I: Fundamentos Basicos del Lenguaje Basic

Unidad I: Principios Básicos

Primero lo primero: Como Descargar Blitz3d

Para descargar el programa Blitz 3d debemos dirigirnos a la página principal del programa en:

www.blitzbasic.com

y luego dirigirnos al apartado **Products** y seleccionar **TRY NOW** del producto Blitz 3d

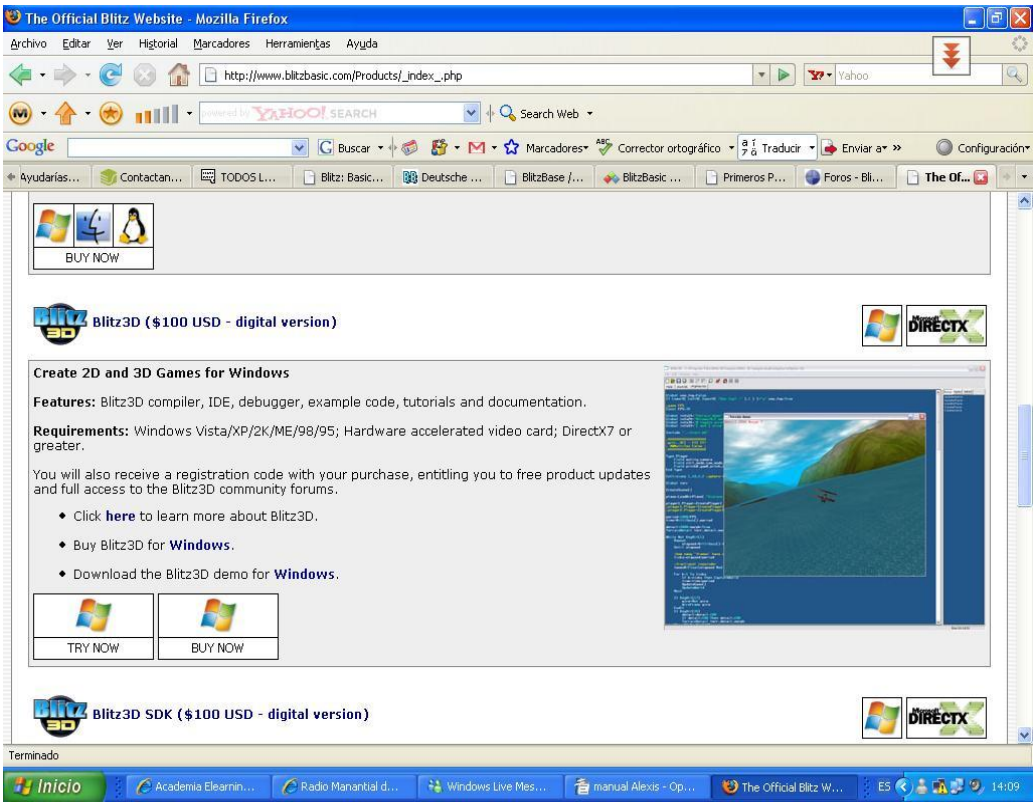


Fig. 1 Pantalla principal para descargar el programa Blitz 3d

Instalación del Programa

Una vez descargado el programa debemos ejecutar el archivo, por ejemplo **Blitz3DDemo183,** y luego seguir los pasos de instalación los cuales son los mismos que cualquier programa de Windows, en caso de que alguien tenga problemas con esto, les pido que me escriban asi lo especifico.

Restricciones de la version de prueba:

La principal restricción del programa version demo, es que no se pueden crear ejecutables (es decir que si o si se debe tener el entorno para ejecutar los juegos) y tiene un limite de tamaño del archivo, lo cual para proyectos complejos no nos serviría, igualmente la verson Full, cuesta 100 USD (lo cual no es mucho) y se puede comprar por internet.

Para realizar este curso utilizaremos la versión de prueba del programa.

Pantalla Principal

Ni bien ejecutemos el programa veremos la pantalla principal del mismo, el cual esta compuesta por un menú, la barar de herramientas, con botones los cuales utilizaremos para hacer determinadas acciones, como por ejemplo ejecutar el juego ypor último encontramos el área central, que es donde escribiremos el código.

En el área central, cada vez que abramos el programa aparecerá la ayuda del mismo, podremos ver desde aca algunos ejemplos, acceder a los comandos y referencia del lenguaje .

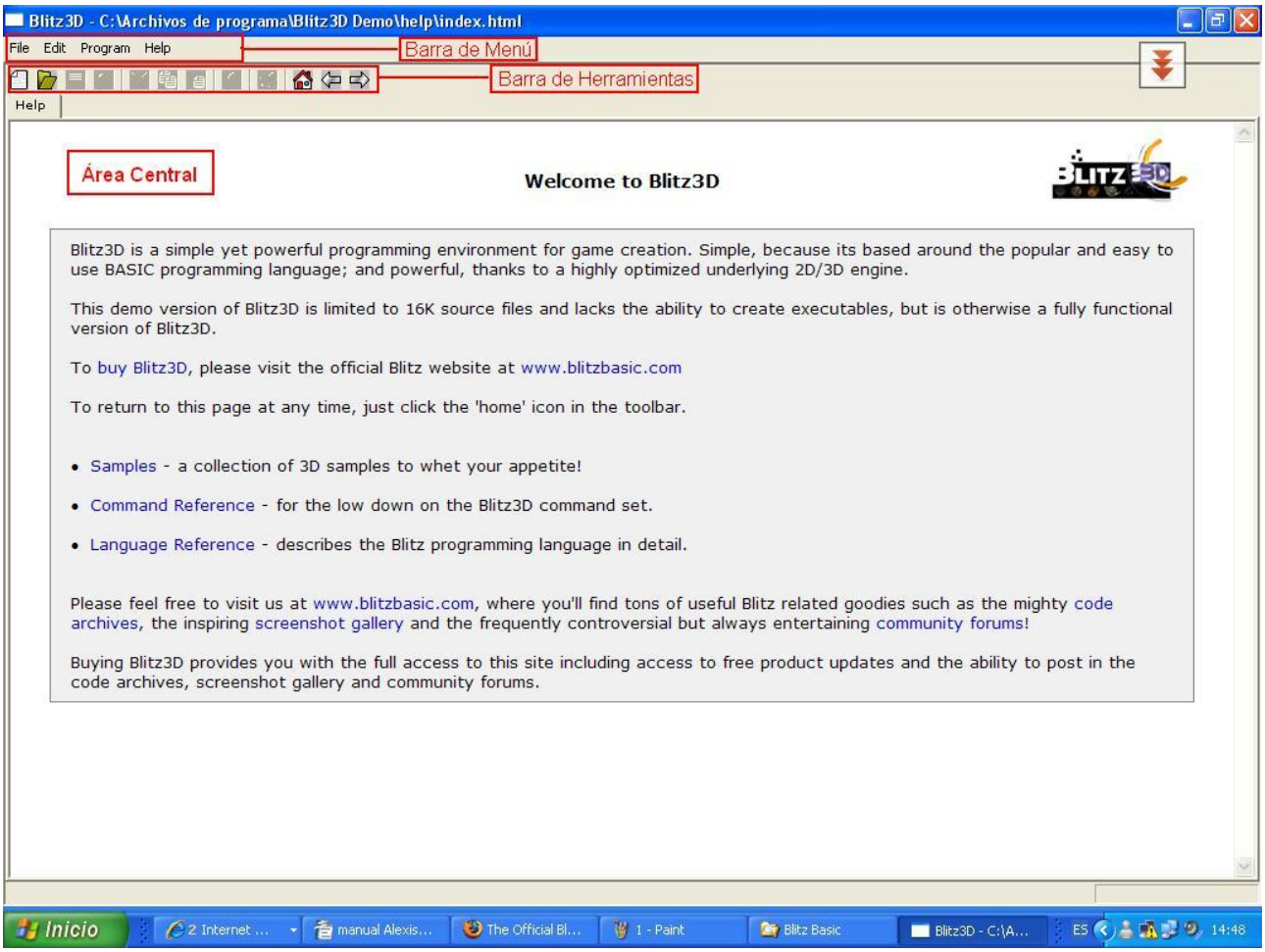


Fig 2 Pantalla Principal del programa Blitz 3d

EL entorno de Desarrollo “IDE”

Barra de Herramientas

Permite realizar operaciones basicas con el codigo Fuente, a continución detallamos la función de cada botón

Fig. 3 Barra de Herramientas.

Nuevo: Abre una nueva pestaña para escribir código, la pestaña actual no se cierra.

Abir: Abre un archivo con codigo fuente que tengamos guardado.

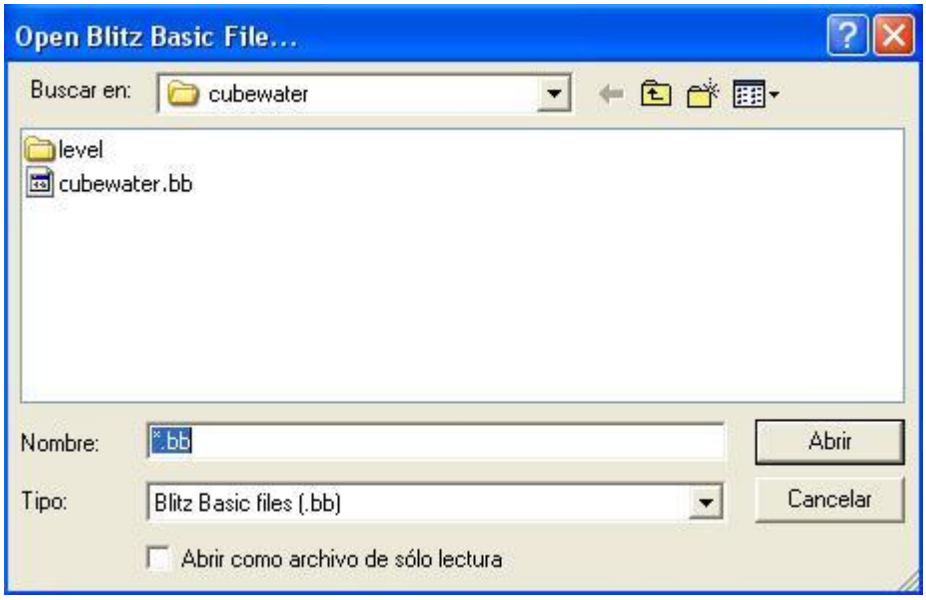


Fig 4. Ventana Abrir

Guardar: Nos permite guardar el archivo en nuestra PC.

Cerrar: Cierra la Pestaña Actual

Cortar: cumple la misma función que en todos los programas windows, corta el texto seleccionado para que luego pueda ser colocado en otra posición o pestaña.

Copiar: Cumple una función parecida a la anterior, pero no borra el texto que se copia.

Pegar: Coloca en la posición donde se encuentra el cursor, el texto que fue copiado o cortado.

Buscar : Permite buscar una palabra en el codigo fuente.

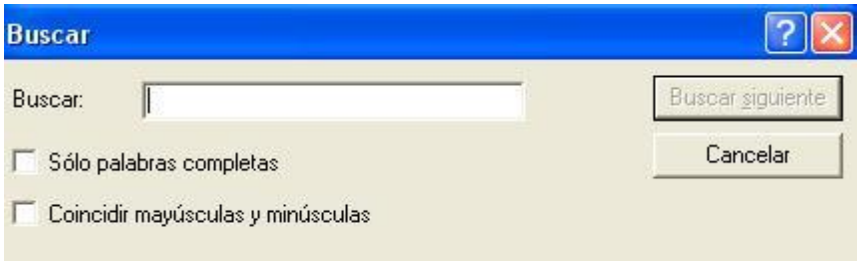


Fig. 5. Ventana Buscar

Ejecutar: Ejecuta el codigo del programa, si todo sale bien tendriamos que ver nuestro juego =)

Home: Nos envia a la pantalla de ayuda.

Barra de Menú

Archivo/File: Nos encontramos con los items comunes al manejo de archivos, por ejemplo nos permite abrir, guardar , cerrar, etc.

Fig. 6: Menú Archivo/File

Edición/Edit: Permite cortar,copiar, pegar seleccionar o remplazar el código fuente. Tambien podemos desde este menú mostrar o esconder la barra de herramientas.

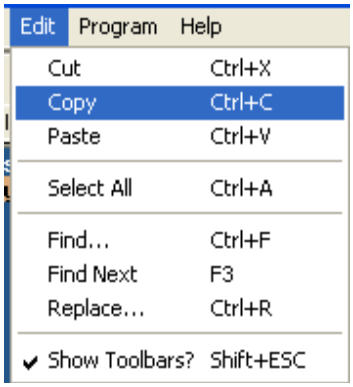


Fig. 7: Menú Edición/Edit

Programa/Program: Permite correr el programa, ejecutar el debugger para encontrar errores, crear el ejecutable,

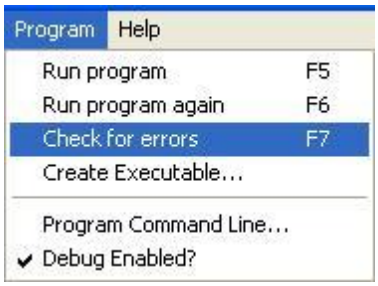


Fig. 8: Menú Programa/PRogram

Ayuda/Help: Muestra la ayuda del programa, muy util para poder aprender, y cuando nos olvidamos algún comando.

Unidad 2: Conceptos Basicos de Blitz3d

¡Vamos a Programar!, pero... ¿Qué es un programa?

Un programa es un conjunto de instrucciones las cuales son ejecutadas por el compilador (este se encarga de traducir, las sentencias de un lenguaje al codigo maquina, es decir a 0 y 1) de forma secuencial.

Hay gran diversidad de lenguajes de Programación (estos los veremos en detalle en el modulo introducción a la programación de computadores) como por ejemplo Assembler, el cual es un lenguaje de bajo nivel o el C, C++, C#, Basic, Etc.

```
DATOS SEGMENT M1 DB "HOLA MUNDO",10,13,"$" DATOS ENDS
CODIGO SEGMENT ASSUME CS:CODIGO, DS:DATOS MOV AX,DATOS MOV DS,AX LEA DX,M1
MOV AH,9 INT 21H MOV AX,4C00H INT 21H CODIGO ENDS END
```

Codigo1: Tipico programa que imprime en pantalla “Hola Mundo” realizado en assembler.

```
Print “Hola Mundo”
```

Codigo 2: Hola mundo en el lenguaje Basic

En los cuadros con el codigo 1 y 2 vemos la gran diferencia entre un lenguaje de Alto nivel, como el basic, y uno de bajo nivel como el Assembler, podemos notar como para hacer una misma tarea, se simplifica utilizando los lenguajes de alto Nivel.

En el modulo correspondiente a introducción a la programación, veremos como se debe codificar

correctamente, las herramientas con que cuentan los programadores, como son los diagramas de flujo y Pseudocodigo y muchas más cosas.

Codificación en Blitz 3d. ¡Comenzemos a Programar!

Muy bien antes de comenzar a crear nuestros programas debemos saber algunas cosas respecto al compiladore de Blitz3d.

Comentarios

Los comentarios permiten ingresar líneas de texto, las cuales no serán tomadas por el compilador como código a ejecutarse, sino que servirán para el programador, para saber que hace una determinada línea o bloque de código. Los comentarios cumplen la función de documentar el código, para los primeros programas no será necesario, ya que serán muy sencillos, pero es una buena practica y se vuelve impresindible en proyectos de mediana o gran envergadura. El simbolo que se usa para avisar al compilador que se trata de un comentario es el punto y coma(;).

```
;Imprime en la pantalla el mensaje Hola Mundo
Print "Hola Mundo"
```

Código 3: En este caso se explica que hace la sentencia "Print".

Palabras Reservadas Estas son palabras que utiliza el lenguaje internamente y no pueden ser utilizadas por identificadores (como por ejemplo: Variables, Nombres de funciones, Etiquetas etc)

After, And, Before, Case, Const, Data, Default, Delete, Dim, Each, Else, ElseIf, End, EndIf, Exit, False, Field, First, Float, For, Forever, Function, Global, Gosub, Goto, If, Insert, Int, Last, Local, Mod, New, Next, Not, Null, Or, Pi, Read, Repeat, Restore, Return, Sar, Select, Shl, Shr, Step, Str, Then, To, True, Type, Until, Wend, While, Xor, Include

Cuadro 1: Palabras reservadas utilizadas por el lenguaje Blitz3d.

Identificadores

Los identificadores se utilizan para poder acceder a una porción de memoria con un determinado valor, es decir en lugar de acceder a la dirección de memoria 0001, accedemos a la variable **"Vida".** Resumiendo un poco, los identificadores serán los nombres de las variables, constantes, estructuras, Funciones, etc.

Fig 9: Dibujo de bancos de memoria de una pc, notese que en este caso la variable "Vida", se encuentra en la posición 0001

Los identificadores tienen una serie de reglas para definirlos:

- Debe empezar siempre por un carácter alfabético, aunque después puede estar precedido por Simbolos o Números.

Los siguientes son ejemplos de identificadores validos:

_Puntaje
Puntaje_1
Vida
vida

Los siguientes son ejemplos de identificadores no validos:

123Vida
@Vida

Debemos tener en cuenta que para B3d, es indistinto si escribimos en mayúsculas o minusculas, por ejemplo VIDA y vida, se tratan de la misma variable.

También Blitz 3d reconoce si utilizas un mismo nombre para diferentes tipos de datos, por ejemplo si defines una variable Puntaje, y luego creas una función Puntaje, Blitz sabe en que momento utilizar cada una.

Tipos de Datos

Blitz trabaja con 3 tipos de datos:

Enteros o integer: Estos son los números que no tienen parte decimal, como por ejemplo -30, 158 y abarcan del rango -2147483648 al +2147483647

Punto flotante o Float: Estos son los números que tienen la parte decimal, por ejemplo 3.5 , 1.0. se debe tener en cuenta que a la hora de realizar calculos, estos se calculan mas lentos que los números enteros.

Cadena o string: son los valores alfanumericos, un dato de tipo string puede contener cualquier tipo de carácter, letras, simbolos o números, pero se debe tener en cuenta que con estos últimos no se podrán realizar calculos.

VARIABLES

Las variables son los tipos de datos mas simples. Son nombres o identificadores que el programador asigna a una porción de memoria donde almacenará valores numericos o caracteres.

Automaticamente, B3D, identifica el tipo de dato número entero. En caso de los valores de tipo números flotantes y alfanúmericos debe ser avisado. Para ello se necesita añadir un simbolo a continuación del identificador "$" para texto y "#" para coma flotante.

Por Ejemplo en la declaración **Nivel=1**, B3D identifica que el valor de la variable **"Nivel"** es un entero, aunque se podria colocar Nivel%=1 (el simbolo % indica que se trata de un valor Entero).

Si quisieramos crear una variable de tipo string con el texto, "Jugador 1 ha muerto" deberiamos hacer:

Mensaje$="Jugador 1 a Muerto"

Un ejemplo de declaración número con coma seria:
Velocidad# = 180.3

Para asignar una valor a una variable se debe utilizar el simbolo Igual (=).

Ejemplo de Asignación

Vida = 3

En las variables se pueden incluir operadores matematicos para poder realizar operaciones con estas, por ejemplo si nuestro personaje perdio una vida podriamos hacer.

Vida = Vida - 1

En este caso la variable vida pasa a tener 2

Alcance de las Variables

Las variables pueden ser:

Globales: Son aquellas variables que se declaran una sola vez, y en las diferentes partes del programa se van modificando, sin importar si se modifica en el programa principal o en las funciones.

Local: Las variables locales, tienen su campo de acción en las funciones que la declaran.

Para declarar una variable Global se debe hacer:

```
; Declaración de una variable global
Global Vida = 3
;Declaración de una variable local
Local Vida=3
```

Código 4: Declaración del alcance de las variables

En caso de que la variable no se declare, la tomara por defecto como **Local.**

CONSTANTES: Las constantes son en realidad variables pero con valores fijos que no serán cambiados durante la ejecución del programa. Adquieren utilidad en aspectos del programa que nunca cambian como el valor del seno de 90º o una resolución de pantalla.

Para declarar una constante se utiliza la palabra Reservada "Const"

```
; Declaración de una constante
Const Max_Velocidad = 200
```

Código 5: Declaración de una constante

Instrucciones del Lenguaje

Sentencias

Las sentencias son las ordenes que le damos al programa para que realize una determinada acción. Por ejemplo si quiero que el programa imprima en pantalla el Hola Mundo, debo indicarle al ordenador colocando la sentencia Print "Hola Mundo"

Las sentencias se pueden agrupar en diferentes categorias:

1. Sentencias de flujo de programa o condicionales
2. Sentencias para bucles
3. Sentencias para calculos o de asignación
4. Sentencia de Control de programa

Flujo de programa:

Estas sentencias permitirán cambiar el rumbo del programa después de comprobar y evaluar si se cumple algún tipo de condicion .

Estas condiciones estan formadas por expresiones que contiene los operadores de comparación, por ejemplo una expresión de este tipo podria ser Vida < 1.

Operadores de Comparación
Mayor que: >
Mayor o Igual que: >=
Menor que: <
Menor o Igual que: <=
Igual que: =
Distinto que: <>

Cuadro 2: Operadores de comparación

Operadores de Lógicos
NOT
AND
OR
XOR

Cuadro3: operadores lógicos

En los operadores de comparación se necesitan dos valores para comparar:

Por ejemplo:

Vida>0

También disponemos de operadores lógicos como: **NOT**(compara que no se cumpla una condición), **AND**(compara que dos expresiones son ciertas), **OR** (compara que al menos una de las dos expresiones es cierta) **XOR** (compara que solo una de las dos expresiones es cierta).

Siempre que realicemos estas comprobaciones, obtendremos como resultado los calores 0 (false) o 1 (True). Además se pueden encadenar expresiones condicionales separadas por operadores lógicos y englobar expresiones dentro de otras utilizando paréntesis, por ejemplo:

```
(Vida<1 or energía<100) and puntos = 0
```

Cuadro 4: Expresiones encadenas

En este caso se evalúa primero lo que esta entre paréntesis y el resultado se comparará con la siguiente expresión; así, se cumplirá la condición si puntos vale 0 y vida es menor que 1 o si puntos vale 0 y energía es menor a 100.

Hay que tener en cuenta que los operadores lógicos convierten sus operandos a valores enteros y el resultado que producen también.

Estructura IF ....Then

Su construcción Básica es la siguiente:

If expresión **then** sentencias1
Else sentencias2

o también

```
If expresión
   sentencias1
Else
   sentencias2
Endif
```

Cualquiera de las dos formas es valida, lo que hace esta estructura es evaluar si la expresión es verdadera, en ese caso ejecuta la sentencias1; si llegara a ser falsa ejecuta las sentencias2.

```
If vida<1 Then jugador_muere
Else vida=vida-1
```

Código 6: Ejemplo del uso de la estructura If

Lo que hace el código 6 es Si (If) vida es menor a 1 (vida<1) entonces (Then) “jugador_muere”; Si no (Else) a vida le resta 1 (vida=vida-1)

Ae pueden encadenar varias estructuras if.

If expresión1
  sentencias1
Elseif expresión2
  sentencias2
Else
  sentencias3
Endif

En esta estructura encadenada, las “sentencias3” se ejecutarán solamente si la “expresion1” del if o la “expresion2” del ElseIf Fueran Verdaderas (True)

```
If nombre$=”Juan” Then
   Print “Saludos, Juan Gonzalez”
Else If nombre$=”Pedro”
   Print “Saludos, Pedro Ramirez”
Else
  Print “No me acuerdo tu apellido”
Endif
```

Código 7: Ejemplo del uso de la estructura If encadenada.

En el código 7 el programa se fija si el nombre ingresado por el usuario es **“Juan”**, en caso de que sea verdadero mostrara **“Saludos juan gonzalez”**, caso contrario entrará en el **Else If** y preguntará si es **“Pedro”** si llegara a ser verdader mostrará **“Saludos, Pedro Ramirez”** por último si no es ninguno de los 2 mostrará en pantalla **“No me acuerdo de tu apellido”.**

¡A PROGRAMAR!

Muy bien vamos a ver un ejemplo del uso del If

```
nombre$=Input("Como es tu nombre: ")

If nombre$= "Juan"
Print "Hola Juan"
Else
Print "No te Conozco"
EndIf
```

Ejemplo1: El codigo de nuestro primer programa

En este caso se utiliza una función nueva la cual es Input, esta función lo que hace es pedirle al usuario que ingrese un texto, cuando este lo ingrese se lo asignamos automaticamente a la variable nombre$.

Por último hacemos la comparación si la variable **nombre$** tiene el texto juan, Mostrará **"Hola Juan"**, caso contrario mostrará **"No te conozco"**

Ejecutemos el programa presionando la tecla **F5** o en el botón **"Run"** para ver que es lo que hace nuestro programa.

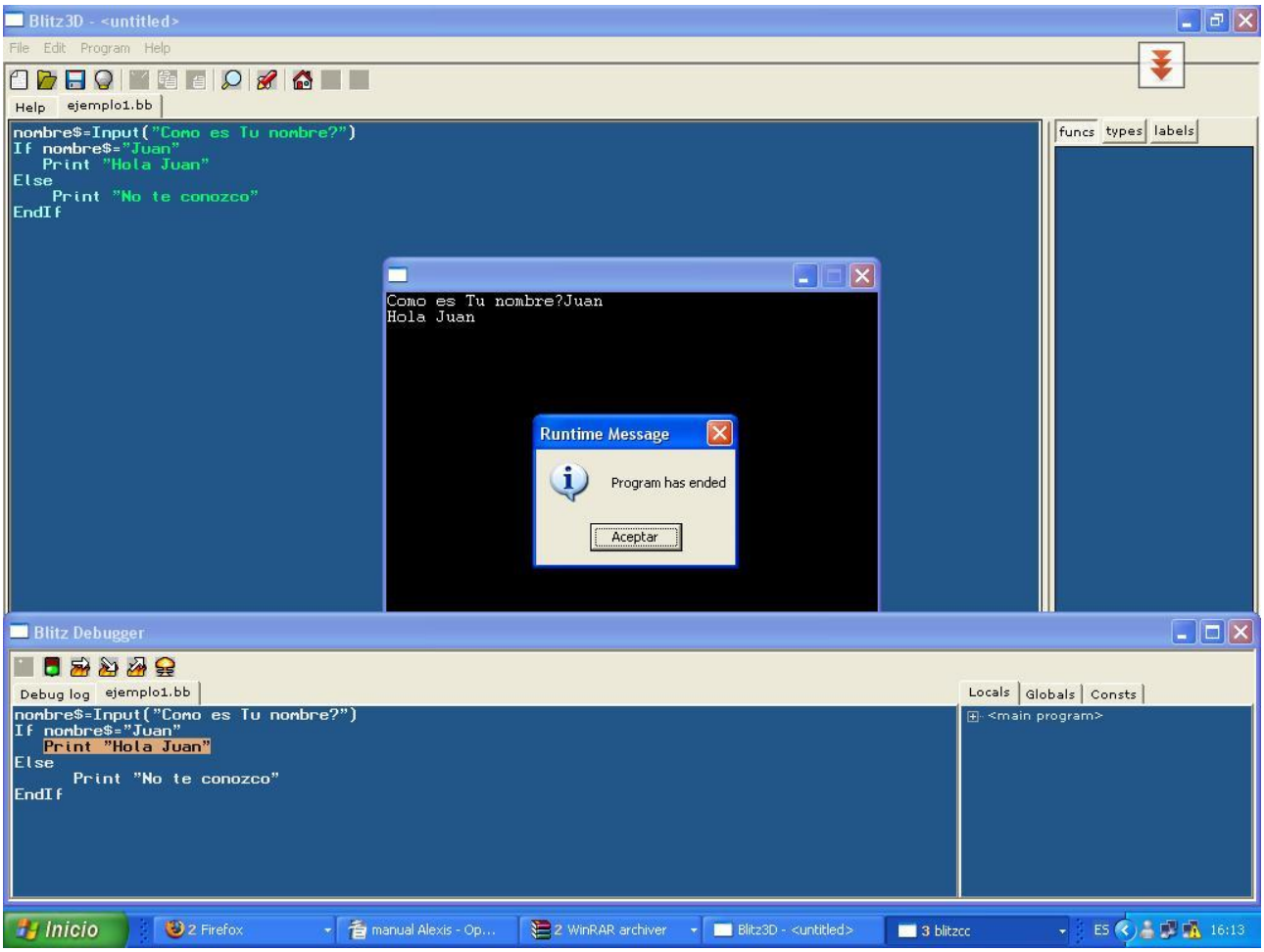


Fig 10: Nuestro primer programa Ejecutandose

Estructura Select ... Case

Select expresión

Case expresiones1
case expresiones2
default expresiones3
End Select

Este tipo de estructuras nos ayudan a evitar un gran numero de estructuras If... Then en nuestro programa. Imaginense que quieren dibujar en pantalla diferentes explosiones dependiendo del tipo de disparo que se haya efectuado. Esto se podría realizar mediante muchos if, por ejemplo,

```
If disparo=1 then
dibujar_explosion1
If disparo2 Then
dibujar_explosion2
If disparo3 then
dibujar_explosion3
```

Realmente este sistema es algo engorroso y poco elegante. La estructura “Select... Case” es una alternativa más apropiada.

```
Select disparo; Según el valor de disparo
Case 1 dibujar_explosion1; En caso de que disparo=1 dibujamos la explosión1
Case 2 dibujar_explosion2; En caso de que disparo=2 dibujamos la explosión2
Default dibujar_explosion3; En el caso restante es decir disparo=3 dibujamos la explosión3
End Select
```

Código 9: Uso de la estructura Case

¡A PROGRAMAR!

Ahora veremos el uso de la estructura case, y aprovecharemos a aplicar lo visto hasta aquí.

```
; Pedimos al usuario que ingrese el número del mes

mes=Input("Ingrese el número de Mes ")

Select mes
    Case 1 nombre_mes$="Enero"
    Case 2 nombre_mes$="Febrero"
    Case 3 nombre_mes$="Marzo"
    Case 4 nombre_mes$="Abril"
    Case 5 nombre_mes$="Mayo"
    Case 6 nombre_mes$="Junio"
    Case 7 nombre_mes$="Julio"
    Case 8 nombre_mes$="Agosto"
    Case 9 nombre_mes$="Septiembre"
    Case 10 nombre_mes$="Octubre"
    Case 11 nombre_mes$="Noviembre"
    Case 12 nombre_mes$="Diciembre"
    Default nombre_mes$="El mes ingresado no existe"
End Select

If (mes >= 1) And (mes <=12)
    Print "El mes "+ mes + " corresponde a " + nombre_mes$
Else
    Print " " + nombre_mes$
End If
```

Ejemplo 2: Uso de la estructura case

En este pequeño ejemplo creamos un programita para decirle al usuario el nombre del mes, a partir de un número ingresado por el.

Lo primero que vemos es que se le pide al usuario que ingrese un número.

mes=**Input**("Ingrese el número de Mes ")

Luego entramos en el Select

Select mes
    Case 1 nombre_mes$="Enero"
    Case 2 nombre_mes$="Febrero"
    Case 3 nombre_mes$="Marzo"
    Case 4 nombre_mes$="Abril"
    Case 5 nombre_mes$="Mayo"
    Case 6 nombre_mes$="Junio"
    Case 7 nombre_mes$="Julio"
    Case 8 nombre_mes$="Agosto"
    Case 9 nombre_mes$="Septiembre"
    Case 10 nombre_mes$="Octubre"
    Case 11 nombre_mes$="Noviembre"
    Case 12 nombre_mes$="Diciembre"
    Default nombre_mes$="El mes ingresado no existe"
End Select

Lo que hace el programa es preguntar el contenido de la variable mes (variable entera) en caso de que su valor sea, por ejemplo, 1 asignamos a la variable **nombre_mes$**, el valor "**Enero**". Entonces en la variable **nombre_mes$** tendremos el nombre del mes dependiendo del numero que haya ingresado la persona, ahora que pasa si la persona ingresa un número erroneo? Por ejemplo el 13?

en este caso entrará en la linea:

```
Default nombre_mes$="El mes ingresado no existe"
```

De esta forma la variable tendrá el texto que avisará de que se ha ingresado mal el número del mes.

Hasta aca el programa esta haciendo lo que queremos, pero aún no nos muestra nada, para que nos muestre deberemos escribir el siguiente codigo:

```
If (mes >= 1) And (mes <=12)
    Print "El mes "+ mes + " corresponde a " + nombre_mes$
Else
    Print " " + nombre_mes$
End If
```

Ups, y aca usamos el if (vean como de apoco vamos aplicando lo aprendido! =))

Bien lo que hacemos en este bloque de codigo, es ver si el número que ingreso es correcto (Es muy probable que el usuario, sufra equivocaciones, asi que nosotros deberemos tratar por todos los medios, que el programa que diseñemos "Avise de estas equivocaciones").

El If , va a preguntar si el valor ingresado en la variable mes, se encuentra en el rango del 1 al 12, si es asi imprime en pantalla un mensaje con el nombre del mes. Ahora....ese print esta bastante raro..no? Expliquemolo.

```
Print "El mes "+ mes + " corresponde a " + nombre_mes$
```

En este caso se esta combinando un texto con variables, el compilador tomará como texto lo que esta entre comillas dobles, el resto lo que esta entre el simbolo mas (+) es la variable de esta forma el mensaje que se mostrará será como el siguiente:

El Mes 1 corresponde a Enero

GoTo Label: Esta Sentencia desvia el flujo del programa hasta la etiqueta definida luego de la palabra reservada.

```
Print "el programa comienza"
Goto etiqueta1
Print "esto no se imprimira nunca en pantalla"

.etiqueta1
Print "¡Saltamos hasta aqui!"

;espera que pulse ESC para acabar
While Not KeyHit(1)
Wend
```

Ejemplo5: Codigo donde se puede ver el uso de la sentencia Goto

Fijemonos que es lo que hace el programa, primero imprime el mensaje **"El programa comienza",** luego usamos la sentencia **goto etiqueta1**, es decir que automaticamente ira a la linea 5 del codigo, luego imprimira **"¡Saltamos hasta aqui!"** el while final se utiliza para que el programa no termine

la ejecución hasta que no le digamos.

```
While Not KeyHit(1)
Wend
```

lo que quiere decir es “Mientras no este presionada la tecla ESC.” se mantiene el codigo en ejecución.

LA función **KeyHit(numero de tecla)** se utiliza para saber si una tecla es presionada, el parametro número tecla es un valor númerico y cada número representa una tecla, en este caso la tecla 1 es ESC.

Sentencias para Bucles

En este tipo de estructura se realiza un bucle de las sentencias hasta cumplirse determinada condición

Estructura While... Wend

construcción básica

While expresión
Sentencias
Wend

Mientras (While) se cumpla de la expresión, se ejecutan las sentencias.
Esta estructura es utilizada cuando queremos que se realicen operaciones sólo cuando se cumpla cierta condición.
Antes de entrar en la estructura para realizar el grupo de sentencias, se evalua la expresión y mientras sea verdadera, se permanecerá en el interior de la estructurá .
En el ejemplo:

```
vida_jugador=3
While Vida_Jugador>0
        ;Dibuja y mueve jugador
        ;dibuja y mueve los enemigos
        ;Atacan los enemigos al jugador
        If disparo_enemigo colisiona con jugador then
        Vida_jugador=Vida_Jugador-1
Wend
muere_Jugador
```

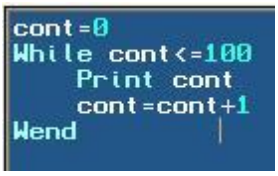
Código 10: ejemplo del uso del While

Asignamos 3 vidas al jugador. Mientras(While) las **“Vidas del Juador”** no sean 0, “Dibujaremos y moveremos al jugador y a los enemigos”, “Atacarán los enemigos”. Si un “Disparo

del enemigo" alcanza "Al jugador", éste pierde una vida. Se vuelve a Preguntar si la "vida del jugador" no es 0; como ahora vale 2 se vuelve a ejecutar las acciones anteriores y asi hasta que las vidas del jugador valgan 0. Se saldrá de la estructura y se ejecutará "Muere Jugador".

¡A PROGRAMAR!

Escribamos el siguiente código en Blitz



ejemplo 3: uso del while

Este ejemplo es bastante sencillo de entender lo que hace el programa es inicializar la variable cont, en 0 luego entra en el bucle, el cual se hará mientras la variable cont sea <=100. una vez que entro en el bucle veremos que imprimirá en pantalla el primer número, luego a la variable cont le sumará 1 y volvera a preguntar si cont es <=100, volviendose a ejecutar la sentencia si es verdadero. En nuestro ejemplo se imprimen los números del 1 al 100, consecutivamente.

Estructura For ... Next

For variable = valor Inicial **To** valor final **Step** salto en el incremento del valor
Sentencias
Next

Este tipo de estructuras para bucles funciona preguntando por la evolución del valor de una variable. Antes de entrar en el bucle, se asigna el "valor inicial" a la "variable" y se entra en el bucle.

Una vez ejecutadas las "sentencias" con la función "next" se regresa al for. Se incrementa una unidad el valor de "variable" (opcionalmente se puede incrementar en las unidades que se indique en "step" (por ejemplo -1, para decrementar el bucle) y se vuelve a entrar en el bucle. Esta operación se repetirá hasta que el valor contenido de la "variable" alcance el "valor final". Por ejemplo, queremos contar del 1 al 10 de dos en dos e imprimirlo en pantalla:

```
For contador = 1 to 10 step 2
        Print contador
Next
```

Codigo 11: Uso del for con Step

Para realizar la cuenta pero al reves, se deben cambiar el "valor Final" por el "Valor Inicial" y colocar "Step" en negativo. Por ejemplo, si queremos hacer una cuenta atrás del 10 al 1:

```
For contador = 10 to 1 step -1
        Print contador
Next
```

Codigo 11: decrementando el contador en un bucle For

También puede haber bucles dentro de otros bucles. El bucle interior se repetirá tantas veces

como indique el bucle exterior.

```
For y = 1 to 600
        For x = 1 to 800
                        dibujar_punto x,y
        Next
Next
```

Codigo 12: Ejemplo de Bucles anidados

```
Print"--------------PRograma Tabla Multiplicar------------"
For Y = 1 To 10
    For x = 1 To 10
            Total_Tabla=x*y
            Print y + "*" + x + "Es igual a" + Total_Tabla
            Continuar$=Input("Presione Enter para continuar")

    Next
Next
```

Ejemplo 3: Uso del bucle For, para generar una tabla de multiplicar.

Muy bien, expliquemos el codigo del ejemplo 3 en este caso la primera linea no deberia ser necesaria explicar a esta altura del manual no? La segunda linea comienza a usarse el bucle, lo que queremos hacer es una tabla de multiplicar del 1 al 10 entonces si pensamos como podriamos hacerla con lapiz y papel, seria:

1 x1
1x 2
1 x 2
....
1x8
1x9
1x10
2 x 1
2x2

Si nos fijamos el operando de la izquierda se mantiene en un número fijo, hasta que el de la derecha va variando del 1 al 10. para representar esto en un programa de Computación, podremos hacerlo usando dos bucles For, anidados.

```
For Y = 1 To 10
    For x = 1 To 10
            Total_Tabla=x*y
            Print y + "*" + x + "Es igual a" + Total_Tabla
            Continuar$=Input("Presione Enter para continuar")
    Next
Next
```

Bucle interno

Bucle Externo

En este bucle fijese que cuando Y = 1, entra al otro bucle donde la variable X valdra 1 tambien, pero cuando termine de ejecutarse el bucle interno, incrementara el valor de X a 2, pero y seguira valiendo 1, hasta que se cumpla que x valga 10 entonces recien ahi Y se incrementa en uno, valiendo de esta forma Y=2, luego vuelve a entrar en el bucle de la X y vuelve a hacer lo mismo que lo anterior, asi sucesivamente hasta que Y=10 con lo que sale del bucle.

Fijense que para podere realizar la multiplicación se hace X*Y y eso se guarda en la variable Total_Tabla.

En la linea siguiente se muestra el resultado de la multiplicación. Por ultimo agregue una sentencia Input, para que el usuario tenga que presionar una tecla para poder continuar y pueda ir viendo los resultados de a poco.

En B3D existe un tipo de bucle For que nos permite recorrer cada uno de los objetos que forman parte de una estructura de datos. Se trata de la estructura For...Each

For variable = Each nombre de la estructura de datos o tipo de datos
    Sentencias
Next

Es muy utilizado para actualizar grandes grupos de objetos. Pero explicaremos el uso de esta sentencia con detalle cuando hablemos de estructuras de datos.

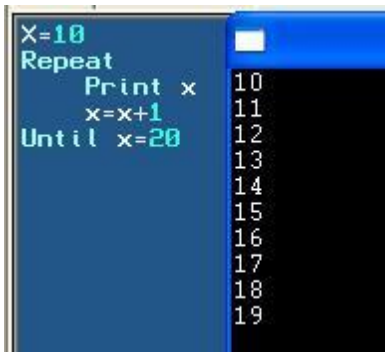
Estructura Repeat... Until

Repear
    Sentencia
Until Expresión

Este bucle de repetición, permite realizar una seria de sentencias hasta que se cumpla una condición dada. La diferencia con el bucle “While” es que en este bucle se ingresa aunque sea una vez.

```
X=10
Repeat
        Print x
        x=x+1
Until X=20
```

Codigo 13: Uso del bucle repeat



Ejemplo 4: aquí vemos el codigo del ejemplo anterior y su salida en la consola.

Muy bien hasta aca tenemos bastante para ir probando, quizas esta sea una de las unidades mas aburridas, porque ni siquiera hemos visto como imprimir un punto de color en pantalla, pero es importante que puedas ver los fundamentos básicos y como tratar los problemas. Ya que si sabemos como solucionar un problema, luego trasladarlo a la pc, es mas facil.

Unidad 3: Arreglos.

ARRAY

Una matriz es en realidad un identificador que se relaciona con muchas posiciones de memoria. Estos array pueden ser de una (Vector) o dos dimensiones (Matriz).

Para definir un array se utiliza la palabra reservada Dim seguida del identificador y entre parentesis la cantidad de celdas de memoria que se le asignará a ese identificador. Por ejemplo, vamos a definir una matriz unidimensional llamada “enemigos” que contenga 5 celdas de memoria:

Dim Tabla(4)

se coloca entre parentesis un 4 en lugar de un 5 porque la primera casilla empieza en 0 y no en el 1, asi pues tenemos 5 valores desde 0 hasta 4.

Podemos asignar valores a cada posición de memoria de la tabla en el momento de la declaración:

Dim Tabla(4) = 12, 30, 40, 50,68

Esto significa que en la posición 0 de tabla hay un 12, en la 1 de Tabla hay un 30 y asi sucesivamente.

Para pasar un valor cualquiera de la tabla a aotra variable, por ejemplo el valor 50 a “fuerza”, escribiriamos:

Fuerza=Tabla(3)
Porque la posición 3 (en realidad es la cuarta) contiene el valor 50.

¡A PROGRAMA!

Bueno veamos un ejemplo un poco más interesante que los anteriores, escribo el codigo y ahora les explico que es lo que vamos a hacer.

```
;ponemos el modo grafico en 800 x 600, para lograr tener la pantalla mas grande
Graphics 800,600

;#############################################################################################
;                                    Modulo correspondiente al menú
;#############################################################################################
.menu
; borramos todo el texto que hay en pantalla
Cls
;ubicamos el cursor en la posición x=30 e y=20 , asi escribimos los textos en ese lugar
Locate 30,20

;imprimimos el menú
Print "------------------------------ Sistema de Alumnos ----------------------------"

 Print " "
 Print "1-Cargar alumnos"
 Print ""
 Print "2 -Consultar datos de alumnos"
 Print " "

;preguntamos al usuario a donde quiere ir, si presiona 1 va a la seccion carga y si presiona 2
;va a la sección lectura

resp$=Input("ingrese número de opción " )
    If resp$="1"
        Goto carga
    Else
        Goto leer
    EndIf
```

```
;#############################################################################################
;                          Modulo correspondiente a la carga de alumnos
;#############################################################################################
.carga
Cls
Locate 30,20
Print "------------------------------ Cargar Alumnos ----------------------------"
Dim alumno$(4)
Print " "
;llenamos el vector desde la posición 0 a la 4
For i=0 To 4
    alumno$(i)=Input("Ingrese nombre de alumno Número: " + i + " ")
Next
;borramos todo lo que hay en pantalla
Cls
Locate 30,20
Print "------------------------------ Listado de Alumnos Cargados --------------- "

; Leemos el vector antes cargado y lo mostramos en pantalla
For i=0 To 4
    Print " "
    Print alumno$(i)
    Next

Print " "
Print "ha cargado todos los alumnos"

Print " "

;Pedimos al usuario que ingrese enter para ir al menú principal

Input("presione la tecla ENTER para volver al menú")
If KeyHit(28)
    Goto menu
EndIf

;#############################################################################################
;                        Modulo correspondiente a la lectura de alumnos
;#############################################################################################

.leer
Cls
Locate 30,20
Print "------------------------------ Leer Alumnos ----------------------------"
;pedimos al usuario que ingrese la posición que quiere leer
pos=Input ("ingrese posición que desea leer")

;mostramos el contenido de la posición ingresada por el usuario
Print "ha leido la posición " + pos + " el alumno que se encuentra en esa posición se llama "
; muestra el texto hasta que la persona presione ESC
While Not KeyHit(1)
Wend
```

Ejemplo 6: El código de Nuestro sistema de Alumnos.

Muy bien, ahora pasemos a explicar lo que hace este pequeño sistema la primera instrucción nueva que aparece es :

```
Graphics 800,600
```

Lo que hace esta sentencia es colocar el modo grafico en una resolución de 800 x 600, lo primero que notaremos al cambiar la resolución, es que la ventana se agranda.

Otra resolución por ejemplo es 640 x 480.

estos dos parametros son los minimos que necesita esta función, pero tambien tiene otros parametros opcionales que pueden ser incluidos o no.

La función se compone de:

Graphics width, height, color depth,[mode]

Width y height: estos valores representan al tamaño de la pantalla expresado en pixeles.

Color Depth: representa la calidad del color expreado en bit (0,16,24 o 32 bit)

Mode: Corresponde al modo de video

0 : auto – Modo ventana en debug, Pantalla completa cuando no se usa el debug (este es el modo por defecto)
1 : Modo pantalla completa
2 : Mode ventana
3 : Modo ventana personalizada

Es decir que si ponemos:

```
Graphics 800,600 ,16,1
```

Estamos indicando que la resolución es de 800x600 con una profundidad de color de 16 bit y en modo fullscreen.

Luego el comando **CLS**, es utilizado para borrar todo el texto que hay en pantalla.

Luego de borrar la pantalla colocamos el cursor en la posición X=30 y=20 de la pantalla

```
Locate 30,20
```

Esto es para que el texto que se vaya escribiendo empiece siempre en esa posición, caso contrario veremos como el texto a medida que aparece se va bajando.

Una vez ubicado el cursor de texto entraremos en el proceso para imprimir el menú, no tiene mucha complicación, fijense que las lineas:

```
resp$=Input("ingrese número de opción " )
    If resp$="1"
        Goto carga
    Else
        Goto leer
    EndIf
```

lo único que hace es tomar del teclado la opción elegida por el usuario y ver, si es 1 salta a la carga de datos y si es 2 (o otro carácter) iria a la sección de lectura de un dato.

Sección Carga de Alumnos

Bien esta sección es la que se encarga de cargar los alumnos al vector.

Lo primero que se hace es definir el array alumno, en este caso esta definido como string con 5 posiciones (recordemos de la 0 a la 5)

```
Dim alumno$(4)
```

luego procedemos al llenado del array

```
For i=0 To 4
    alumno$(i)=Input("Ingrese nombre de alumno Número: " + i + " ")
Next
```

utilizamos un para (FOR) que repetira 5 veces el bucle. La sentencia que se encuentra dentro del bucle lo unico que hace es leer el dato ingresado y colocarlo en la posición correspondiente del vector, notese que dentro de los parentesis de **“alumno$”** Se encuentra la variable **i,** esto sirve para que cada vez que ingrese en el bucle se llene una posición diferente del vector.

Por ejemplo la primera vez se llenara la posición 0, la segunda vez la 1 y asi susecivamente.

Con ese código ya pudimos cargar el vector, ahora listaremos los nombres cargados, con el siguiente código.

```
For i=0 To 4
   Print " "
    Print alumno$(i)
    Next
```

fijemonos, que este código no nos deberia de ser dificil de entender, lo que hace es nuevamente con un blucle For, recorrer el vector y lo va mostrando (Print alumno$(i))

Modulo Lectura de alumnos

Bueno veremos que para leer el dato de una posición del vector no es muy complicado, lo primero que debemos hacer es pedir al usuario que ingrese la posición que desea leer.

```
pos=Input ("ingrese posición que desea leer")
```

En pos se guardará el número elegido por el usuario.

Luego deberemos mostrar el nombre que se encuentra en una determinada posición.

```
Print "ha leido la posición " + pos + " el alumno que se encuentra en esa posición se llama " + alumno$(pos)
```

Bien para implementar esto, no tenemos más que hacer que colocar un print, con las variable correspondiente (pos) la cual contiene el valor ingresado por el usuario, por ejemplo si el usuario eligio el numero 3 imprimiria en pantalla;

Ha leido la posición **3** El Alumno que se encuentra en esa posición se llama **Juan**

Tarea para el hogar!

Bien el programa esta funcional, pero seria bueno que se implemente lo siguiente:

- Cuando se elige una opción (la 1 o la 2) que permita elegir esas 2 si se presiona alguna otra tecla que me muestre un mensaje de error y me vuelva al menú para ingresar el número correcto.
- Que el menú tenga una tercera opcion que lo que haga sea salir del programa.
- Cuando quiera leer la posición de un alumno, si esta no existiera, me de un mensaje de error.

Y si tienen alguna cosita mas para mejorar el programita posteenlo en el foro asi todos aprendemos!

Array Bidimensional

Un array bidimensional (tabla o matriz) es un array con dos indices.
Para localizar o almacenar un valor en el array se deben especificar dos posiciones (dos subindices), uno para la fila y otro para la columna. Los elementos se referencian con el formato:

Tabla(2,4) elemento de la fila 3 y columna 4.

	1	2	3	4
1	T(1,1)	T(1,2)		T(1,4)
2				
3				
4	T(4,1)			T(4,4)

Fig. 11: Array de dos dimensiones T con 4 filas y 4 columnas.

Como definir un array

Para definir un array de dos dimensiones debemos hacerlo de la siguiente manera:

```
Dim notas(4,3)
```

En este caso estamos definiendo que nuestro array tendra 5 filas (de la 0 a la 4) y 4 columnas (de la 0 a la 3) con lo cual nos da un total de 20 posiciones para almacenar valores (5x4=20)

---------- 2 Parte----------

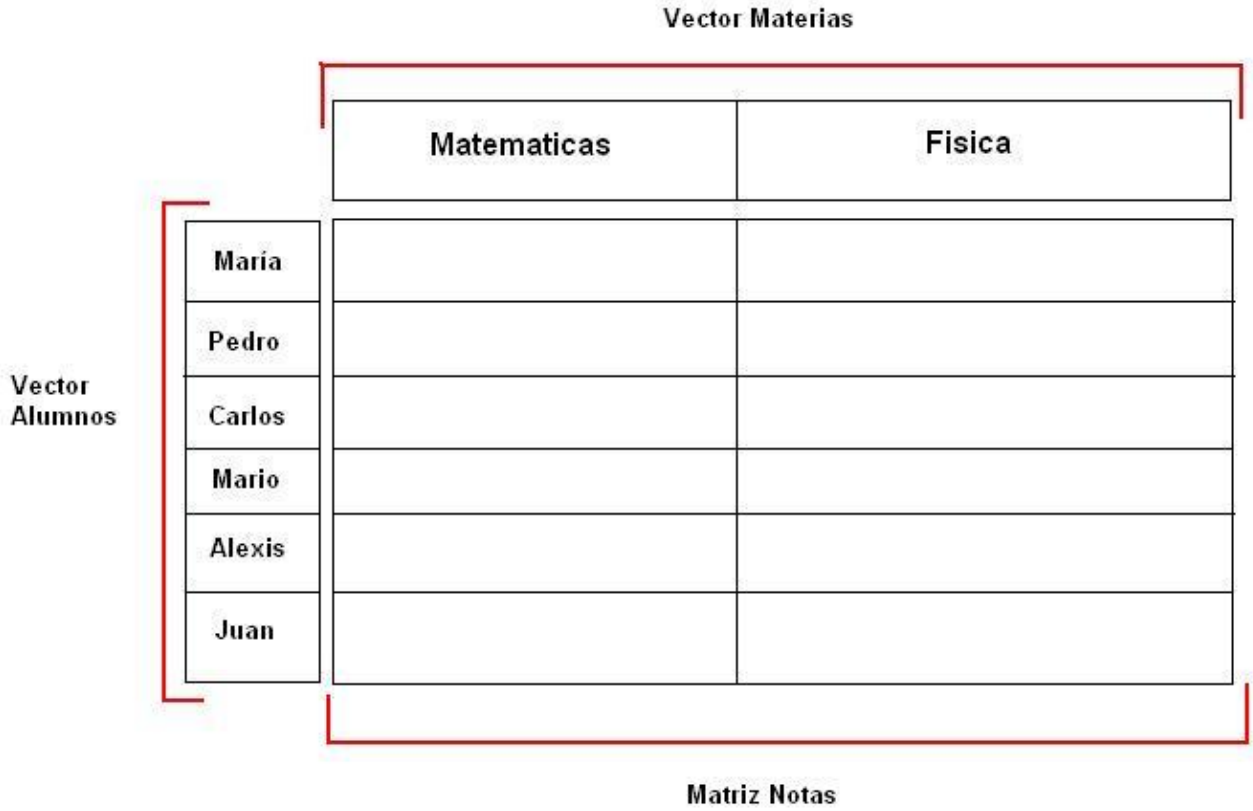
Operaciones con las matrices

Con las matrices podemos realizar diferentes operaciones en este caso veremos la de carga de datos a una matriz y la de Lectura de datos de una matriz

Carga de Datos

Por lo general para cargar un dato se utilizará dos bucle for anidados los cuales recorreran cada uno un indice.

Siguiendo con el sistema de alumnos, veremos ahora un ejemplo utilizando las matrices. Hagamos de cuenta que al sistema que hicimos en el ejemplo6 queremos ahora poder agregarle las notas de una determinada cantidad de alumnos.



Haremos un esquema para entender un poco mejor lo que desarrollaremos.

Fig. 12. Esquema de actualización del Sistema Alumnos

Entonces nuestro pequeño sistema constará de 2 vectores (Materias y Alumnos) y una matriz (notas).
El primer cuadrado vacio contendrá la nota de María en Matematicas el segundo cuadro (si leemos la matriz horizontalmente) contendrá la nota de Maria pero en la materia Fisica. Asi sucesivamente con los diferentes alumnos y materias

Suponiendo que las materias y los alumnos estan cargados (esto lo deberian saber ya, ante cualquier duda recuerden consultar en el foro) ahora escribiremos el codigo para cargar las notas de los alumnos:

```
For i=0 To 5
	For j=0 To 1
		notas(i,j)=Input("Ingrese nota del alumno " + alumnos$(i) + " en la materia " + materia$(j))
	Next
Next
```

El primer for es para recorrer el vector Alumnos, el segundo For es para recorrer el vector Materias, en la primera pasada i=0 y j=0, según el ejemplo el alumno Maria se encuentra en la posición 0 y la materia matematica se encuentra en la posición 0 de su vector, esto quiere decir que cuando pide que ingrese la nota del alumno$(i) el sistema se refiere a Maria, y cuando dice en la materia "materia$(j)" se refiere a matematica.
Ahora, el dato que el usuario ingrese se guardara en notas(i,j) en este caso seria la posición (0,0) de la matriz, es decir el primer cuadrado.

En la segunda pasada i=0 y j=1

Maria habiamos dicho que era la alumna que se encontraba en la primera posición del vecto, ahora la j se incremento en 1 (segunda posición del vector materias) en este caso la materia que se encuentra en esta posición es Fisica. El dato que el usuario ingrese se ubicara en la posición (0,1) del vector. Es decir en el 2 cuadrado del ejemplo visto en la **figura 12**

Luego veremos que J llego hasta el final de su recorrido, entonces i se incrementa en uno y pasa a valer 1(Es decir pasaria a Pedro) y vuelve a entrar en el bucle de las materias con j=0 (matematica) cuando usuario ingrese la nota verá q la nota se guardara en la posición (1,0).

cuando i=5 significará que el bucle llego a su fin, con lo cual se deberian haber cargado las notas de todos los alumnos.

Leer Datos

La lectura de datos es muy parecida a la escritura.

```
For i=0 To 5
	For j=0 To 1

		Print "El alumno " + alumno$(i) + " se saco un " + notas(i,j) + " en la meteria " + materia$(j)
```

Next
Next

Vemos que es la misma estructura que el anterior solo que usamos un print para mostrar el dato obtenido.

Ahora veremos el código completo de la aplicación de Sistema de Alumnos

```
;ponemos el modo grafico en 800 x 600, para lograr tener la pantalla mas grande
Graphics 800,600,16,1

;*************************************************************************************
;                                  Modulo correspondiente al menú
;*************************************************************************************
.menu

; borramos todo el texto que hay en pantalla
Cls
;ubicamos el cursor en la posición x=30 e y=20 , asi escribimos los textos en ese lugar
Locate 30,20

;imprimimos el menú
Print "--------------------------------- Sistema de Alumnos ------------------------------"

 Print " "
 Print "1-Cargar alumnos"
 Print ""
 Print "2- Cargar materias"
 Print " "
 Print "3 -Cargar Notas"
 Print ""
 Print "4- Consultar datos de alumnos "

Print " "
;preguntamos al usuario a donde quiere ir, si presiona 1 va a la seccion carga y si presiona 2
;va a la sección lectura

resp$=Input("ingrese número de opción " )

    If resp$="1"
        Goto carga

        Else If  resp$="2" Then

            Goto carga_materias
        Else If  resp$="3" Then
            Goto carga_notas
        Else
             Goto leer

        EndIf
```

```
;*************************************************************************************************
;                          modulo para ingresar las notas
;*************************************************************************************************

.carga_notas

;Ingresamos notas

;Utilizamos dos for para recorrer los vecores alumnos y materia y luego ingresar las notas
For i=0 To 4
    For j=0 To n-1
        notas(i,j)=Input("Ingrese nota del alumno " + alumno$(i) + " en la materia " + materia$(j))
    Next
Next
Cls
Locate 30,20
Print "-------------------------------- Listado de Alumnos Cargados ---------------- "

;Mostramos los datos de las notas de los alumnos
For i=0 To 4
    For j=0 To n-1

        Print "El alumno " + alumno$(i) + " se saco un " + notas(i,j) + " en la meteria " + materia$(j

    Next
Next



Input("presione la tecla ENTER para volver al menú")
If KeyHit(28)
    Goto menu
EndIf


;*************************************************************************************************
;                    Modulo correspondiente a la lectura de alumnos
;*************************************************************************************************

.leer
Cls
Locate 30,20
Print "------------------------------- Leer Alumnos ------------------------------"
;pedimos al usuario que ingrese la posición que quiere leer
usuario$=Input ("ingrese alumno que desea leer ")
Print " "
materialu$=Input ("ingrese materia que desea leer ")
For i=0 To 4
    For j=0 To n-1

If usuario$ = alumno$(i)
    If materialu$ = materia$(j)
        Print "El Alumno " + alumno$(i) + " se saco un " + notas(i,j) + " En la materia " + materia$(j

        EndIf
    EndIf
    Next
    Next
            ;mostramos el contenido de la posición ingresada por el usuario

; muestra el texto hasta que la persona presione ESC

While Not KeyHit(1)
Wend

Locate 30,20

;Pedimos al usuario que ingrese la cantidad total de materias
n=Input("ingrese cantidad de materias " )
;Ingresamos nombres de Materias
For i=0 To n-1
    materia$(i)=Input("Ingrese nombre de la materia: " + i + " ")
Next

Input("presione la tecla ENTER para volver al menú")
If KeyHit(28)
    Goto menu
EndIf
```

Ejemplo 7: El codigo actualizado del sistema de alumnos

Muy bien a continuación explicaremos alguna partes del codigo. (ten en cuenta que la mayoria del codigo es igual al del ejemplo 6)

```
;Pedimos al usuario que ingrese la cantidad total de materias
n=Input("ingrese cantidad de materias " )
For i=0 To n-1
      materia$(i)=Input("Ingrese nombre de la materia: " + i + " ")
Next
```

Codigo14:

Lo que hacemos el bloque de texto correspondiente al código 14 es pedir al usuario la cantidad de materias que necesitamos ingresar, para luego recorrer el bucle hasta n(cantidad de materias) – 1. ¿Pero por que hacemos n-1? Hacemos eso porque el bucle comienza en 0(podriamos comenzar en 1 pero desaprovecharíamos una posición del Vector). Veamos esto con un ejemplo:

si el usuario quiere ingresar 2 materias, el valor de n seria 2 si el bucle fuera

```
For i=0 To n
      materia$(i)=Input("Ingrese nombre de la materia: " + i + " ")
   Next
```

Le pediria que ingrese la materia 0, la materia 1 y la materia 2 con lo que ingresaria 3 materias en lugar de dos. Es po eso que para evitar este error restamos 1 a n.

El modulo para ingresar las notas, no lo explicaremos aca por que ya fue explicado en la **página 25** Cuando vimos como podíamos cargar datos en una matriz.

En el ultimo modulo, Lectura de Alumnos, escribimos el siguiente código

```
usuario$=Input ("ingrese alumno que desea leer ")
Print " "
materialu$=Input ("ingrese materia que desea leer ")
For i=0 To 4
      For j=0 To n-1

If usuario$ = alumno$(i)
      If materialu$ = materia$(j)
            Print "El Alumno " + alumno$(i) + " se saco un " + notas(i,j) + " En la materia " +
materia$(j)

            EndIf
```

En la primera línea pedimos que se ingrese el alumno del cual necesitamos saber la nota lo guaramos en una variable nueva llamada **usuario$**

Luego pedimos la materia del alumno en el cual deseamos saber la nota, y se guarda en una variable llamada **materialu$.**

Estos dos valores los necesitaremos para poder comparar con los datos obtenidos del array si son iguales nos mostrará la nota.

Luego que el usuario ingreso el nombre y la materia ingresamos en los **for** para recorrer la matriz.
En la parte de código

```
If usuario$ = alumno$(i)
    If materialu$ = materia$(j)
        Print "El Alumno " + alumno$(i) + " se saco un " + notas(i,j) + " En la materia " +
materia$(j)

        EndIf
    EndIf
```

lo que hacemos es preguntar:

El contenido de la variable usuario$ es igual al contenido de la posición i del vector Alumno$?
El contenido de la variable materialu$ es igual al contenido de la posición j del vector materia$?

Si es asi muestra el nombre del alumno el nombre de la materia y la nota que se saco en esa materia.

Con esto concluimos nuestro primer sistemita, ahora bien esto se puede mejorar mucho y ademas no es la forma mas correcta hacerlo con array porque cada vez que inicamos el programa tenemos que cargar todos los datos de nuevo =(es decir que los datos de los array se mantienen durante el tiempo de ejecución cuando se termina de ejecutar se setean las variables.

Como lo pudimos haber hecho? Con archivos, pero esto es parte de otra historia, de la historia sin ...ups perdón esto lo veremos en otra unidad =) .

Tarea para el Hogar

- El programa muy lindo pero que ocurre si la persona ingresa como nota un 23? existe esa nota? Como podríamos validar el ingreso de una determinada nota para que solo permita valores de 1 a 10?

- Actualizar el modulo de carga de alumnos para que el usuario decida cuantos alumnos ingresar.

- En el Modulo correspondiente a la lectura de alumnos. ¿Que pasa si el alumno que ingreso el usuario no existe? Corregir para que avise al usuario si ingreso un nombre de usuario o materia de forma incorrecta, y que le vuelva a permitir ingresar los datos.

Bueno y a aquellos voluntarios que quieran corregir alguna otra parte bienvenido sea!! si han realizado este manual hasta aca podemos decir que ya tenemos bastante noción de lo que es programación, al menos de la lógica, es interesante que puedan observar como se va combinando todos los comandos que estamos aprendiendo, lo importante es entender todos estos conceptos básicos para más adelante ir aumentando la complejidad.

---------- FIN PARTE I: El Lenguaje Blitz

Ahora bien pasemos a algo que a la mayoria les encanta. **LAS MATEMATICAS!!!**

Unidad 4. Funciones

Una función es un mini programa dentro del programa principal. Las funciones pueden contener varias sentencias bajo un solo nombre, que un programa puede usar una o más veces para ejecutar dichas sentencias. Las funciones ahorran espacio, reduciendo repeticiones y haciendo más fácil la programación, proporcionando un medio de dividir un proyecto grande en módulos pequeños más manejables.

Veremos con un ejemplo como utilizar una función, en este caso vamos a hacer una función llamada Imprimir que cumpla la misma función que la sentencia Print de Blitz, es decir que imprima un texto en pantalla.

Tenga en cuenta que este es un simple ejemplo, en este caso no sirve mucho crear una función ya que no nos estaríamos ahorrando nada de código, pero a fines didácticos sirve para ver cómo funciona una función (valga la redundancia).

```
;programa principal

texto$=Input("Escriba un texto: ")
imprimir(texto$)
Delay(5000)
End


;Funcion Imprimir

Function imprimir(texto$)
Print texto$
End Function
```

En este código lo primero que hacemos solicitar que el usuario ingrese un texto, este se almacena en la variable texto$.

Luego utilizamos la función creada por nosotros, IMPRIMIR(texto$), a esta función le debemos pasar por parámetro la variable que contiene el texto que deseamos mostrar en pantalla.
Luego hacemos un delay para que se muestre el texto en la pantalla por 5 segundos.

Más abajo creamos la función que será llamada desde el programa principal, las funciones empiezan con la palabra Clave FUNCTION, luego debemos colocar un nombre que la identificará en nuestro caso el nombre es Imprimir, por último entre paréntesis deben ir los parámetros que se pasarán para que la función trabaje de forma correcta.

Luego se escriben las sentencias, en nuestro caso usamos Print para mostrar el texto, para cerrar la función es necesario colocar la palabra clave End Function.

Estructura de una función

Las funciones tienen la siguiente estructura

```
Function
nombreFunción(listaparámetros)

Cuerpo de la función

End Function
```

nombreFunción: Indicador o nombre de la función
listaparámetros: lista de declaración de los parámetros de la función, separados por coma.

Llamada a una función

Las funciones para poder ser ejecutada, han de ser llamadas o invocadas. Cualquier expresión puede contener una llamada a una función que redirigirá el control del programa a la función nombrada. Normalmente, la llamada a una función se realizará desde el programa principal, aunque naturalmente también podrá ser desde otra función.

```
; Programa principal

Func1()
Func2()

Function Func1()
…
End Function

Function Func2()
…
End Function
```

Llamadas de funciones

A lo largo de este manual estaremos trabajando constantemente con funciones personalizadas, de esta forma se podrá ir afianzando este tema.

FUNCIONES de BLITZ 3D

Matematica en Blitz 3d

Las funciones matematicas mas de una vez nos facilitaran la programación de juegos, por ejemplo ya lo veran cuando veamos como hacer rebotar una pelotita ;) por eso es necesario que aprendamos matematicas y muchas veces deberemos leer libros especializados para poder solucionar algún problema.

Funciones en Blitz 3d Matematicas

- Pi
- Int
- Float
- Floor
- Ceil
- Sgn
- Abs
- Mod
- Sqr
- Sin
- Cos
- Tan
- ASin
- ACos
- ATan
- ATan2
- Exp
- Log
- Log10
- Xor
- Shl
- Shr
- Sar
- Rnd
- Rand
- SeedRnd
- RndSeed

Tabla 3: Funciones matematicas

PI:

Esta función devuelve el valor PI, muy usado en las rutinas geometricas.

```
Print "el valor Pi es = a" + Pi
While Not KeyHit(1)
```

```
Wend
```

ejemplo8: Este codigo muestra el valor de Pi

INT:

Descripción: Esta funcion redondea un valor númerico o una variable valida al número más cercano

Sintaxis: INT(valor)

Parametros: Valor – algun valor o variable valida

```
N#=2.7
n1#=2.2
Print "El valor redondeado de " + n + " es igual a " + Int(n)
Print "El valor redondeado de " + n1 + " es igual a " + Int(n1)
While Not KeyHit(1)
Wend
```

Ejemplo9: Ejemplo de la funcion INT

FLOAT

Descripción: Convierte un valor entero en uno de coma flotante

Sintaxis: FLOAT valor

Parametro: valor – puede ser un valor entero o variable

```
a=100
b#=2.5
c#= Float a
Print b# + c#
While Not KeyHit(1)
Wend
```

Ejemplo 10: Ejemplo de la funcion FLOAT

FLOOR#

Descripción: Redondea un valor de coma flotante hacia abajo.

Sintaxis: FLOOR# (valor)

Parametro: valor – debe ser un valor o variable de tipo coma flotante.

```
n#=2.7
n1#=2.2
Print "El valor redondeado de " + n + " es igual a " + Floor#(n)
Print "El valor redondeado de " + n1 + " es igual a " + Floor#(n1)

While Not KeyHit(1)
Wend
```

Ejemplo 11: Función FLOOR#

CEIL#

Descripción: Redondea un valor de coma flotante hacia arriba.

Sintaxis: CEIL# (valor)

Parametro: valor – debe ser un valor o variable de tipo coma flotante.

```
n#=2.7
n1#=2.2
Print "El valor redondeado de " + n + " es igual a " + Ceil#(n)
Print "El valor redondeado de " + n1 + " es igual a " + Ceil#(n1)

While Not KeyHit(1)
Wend
```

Ejemplo 12: Función CEIL#

SGN

Descripción: Devuelve el signo del número especificado, se utiliza para saber si un número es mayor a 0, menor a 0 o igual.

Sintaxis: SGN (valor)

Parametro: valor – debe ser un valor o variable de tipo coma flotante o entero.

```
Print Sgn(10)
Print Sgn(5.5)
Print Sgn(0)
Print Sgn(0.0)
Print Sgn(-5.5)
Print Sgn(-10)

While Not KeyHit(1)
Wend
```

Ejemplo 13: Función Sgn (valor)

ABS

Descripción: Devuelve el valor absoluto (positivo de un número)

Sintaxis: ABS (número)

Parametro: valor – debe ser un valor o variable numerico

```
n=-2
n1#=-2.2
n2=2
n3=2.2
Print "El valor absoluto de " + n + " es igual a " + Abs(n)
Print "El valor absoluto de " + n1 + " es igual a " + Abs(n1)
Print "El valor absoluto de " + n2 + " es igual a " + Abs(n)
Print "El valor absoluto de " + n3 + " es igual a " + Abs(n1)
While Not KeyHit(1)
Wend
```

Ejemplo 14: Función Valor absoluto

MOD

Descripción: Devuelve el resto de una división.

```
Print 10 Mod 3
While Not KeyHit(1)
Wend
```

Ejemplo 15: Función MOD

SQR (Float)

Descripción: Devuelve la caiz cuadrada del valor especificado como parametro

Sintaxis: SQR(Float)

Parametro: Float = número de coma flotante (los enteros los convierte automaticamente)

```
n=9
n1#=30.2
Print "La raiz cuadrada de " + n + " es igual a " + Sqr#(n)
Print "La raiz cuadrada de " + n1 + " es igual a " + Sqr#(n1)
While Not KeyHit(1)
Wend
```

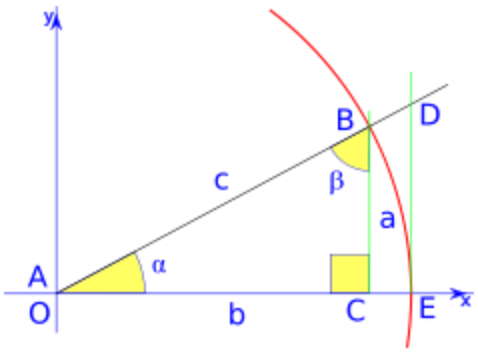
Ejemplo 16: Función SQR

SIN (Número)

Descripción: El comando SIN (Seno),es una función de trigonometria que devuelve un número entre -1 y 1. Este valor representa la coordenada "Y" de un punto. Este comando se usa para trasladar valores de ángulos a coordenadas, pero hay algunas cosas a tener en cuenta a la hora de usarlo. Lo primero de todo, el comando Sin() asume el punto que quieres es de radio 1 (pixel), después usa un circulo donde 0 grados es el punto más al ESTE e incrementa en el sentido contrario de las agujas del reloj, entonces debes tener en cuenta que el eje Y en una pantalla de ordenador es arriba-a -abajo comparado con un sistema de coordenadas matemático normal.

Sintaxis: SIN (Número)

Parametro: Número= número entero o flotante que representa un valor en grados.
Muy bien veamos esto con un ejemplo:



$$\sin(\alpha) = \frac{a}{c}$$

Fig 13. La formula del Seno

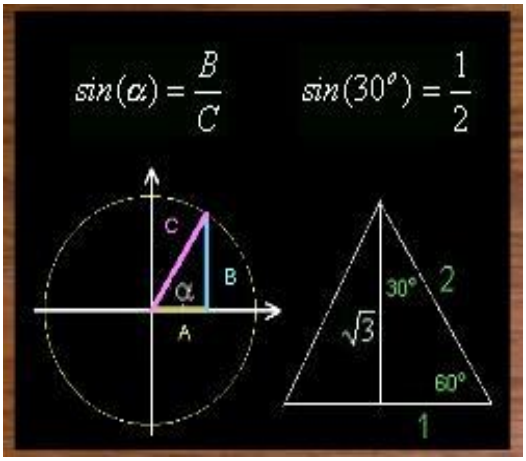


fig: 14 Ejemplo de uso de la formula Seno

Matematicamente el seno de un angulo se calcula dividiendo el cateto opuesto sobre la hipotenusa

Señ (30) = $\frac{\text{Cateto opuesto}}{\text{Hipotenusa}} = \frac{a}{c} = \frac{1}{2} = 0{,}5$

```
;inicializo la variable con el valor en grado para averiguar el seno
n=30
Print "El seno de " + n + "º es:" + Sin(n)
While Not KeyHit(1)
Wend
```

Ejemplo 17. Ejemplo de Seno.

COS (Número)

Descripción: El comando COS (Coseno), es una función de trigonometria que devuelve un número entre -1 y 1. Este valor representa la coordenada “Y” de un punto (x,y). Este comando se usa para trasladar valores de ángulos a coordenadas, pero hay algunas cosas a tener en cuenta a la hora de usarlo. Lo primero de todo, el comando Cos() asume el punto que quieres es de radio 1 (pixel), después usa un circulo donde 0 grados representa la derecha y se incrementa en el sentido contrario de las agujas del reloj, entonces debes tener en cuenta que el eje Y en una pantalla de ordenador es arriba-a -abajo comparado con un sistema de coordenadas matemático normal.

Sintaxis: COS (Número)

Parametro: Número= número entero o flotante que representa un valor en grados.

Matematicamente La función Coseno se obtiene de dividir el cateto adyacente por la hipotenusa de un angulo.

$$\cos(\alpha) = \frac{b}{c}$$

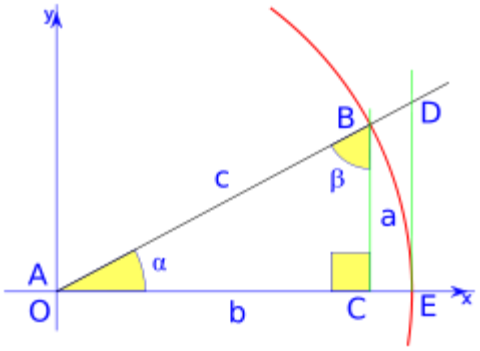
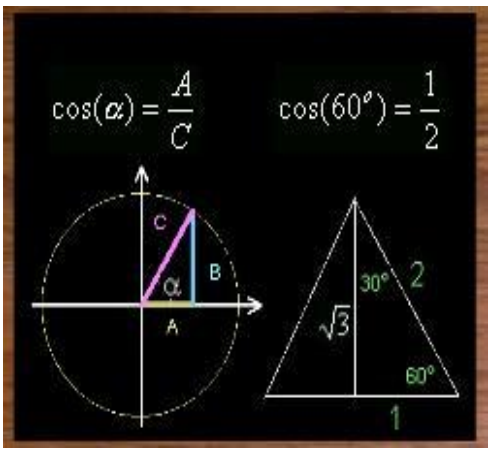


Fig 13. La formula del Coseno



Ejemplo del uso de la Formula Coseno

```
;inicializo la variable con el valor en grado para averiguar el coseno
n=60
Print "El coseno de " + n + "º es:" + cos(n)
While Not KeyHit(1)
Wend
```

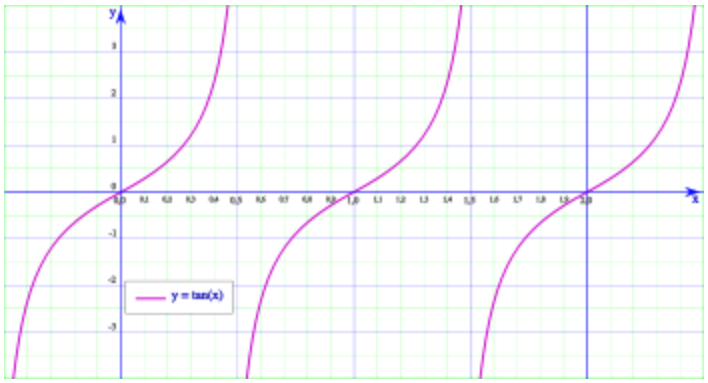
Ejemplo 18. Ejemplo de coseno.

TAN (número)

Descripción: Esta función devuelve la tangente de un ángulo. La funcion tangente obtiene un angulo y devuelve la relación de dos lados de un triangulo rectángulo. La relación es el resultado de la longitud del lado opuesto al ángulo dividido por el lado adyacente.
Esto es útil por ejemplo para juegos que usan armas donde las balas serán disparadas hacia un barril en un cierto ángulo. Puedes usar el valor Tan en tu formula de potencia/ángulo para ayudar a determinar la trayectoria del disparo.

Sintaxis: TAN (Número)

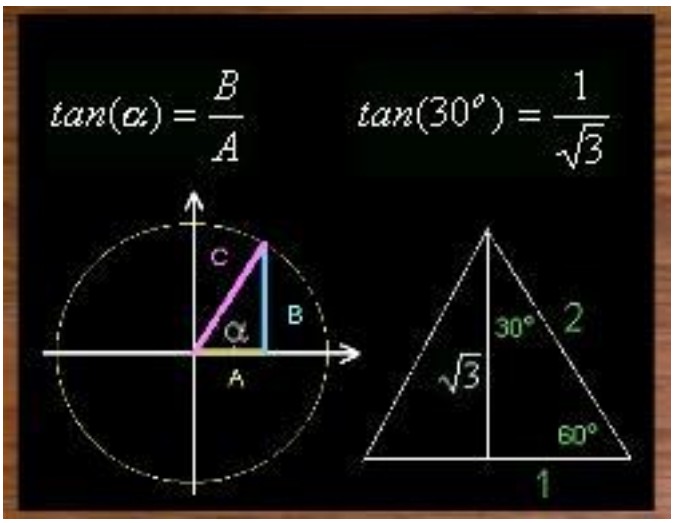
Parametro: Número= número entero o flotante que representa un valor en grados.



$$\tan x = \frac{\operatorname{sen} x}{\cos x}$$

Fig. 14: Gráfico y función tangente

Ejemplo del Uso de la Formula Tangente




```
Angulo=30
Print "La tangente del ángulo 1 es: " + Tan(angulo)
WaitKey
```

Ejemplo 19. Ejemplo de Tangente. El comando Waitkey se utiliza para que al presionar cualquier tecla el programa finaliza.

ASIN (número)

Descripción: Esta función devuelve el arcoseno del parametro especificado. Se utiliza para

trasladar los valores de las coordenadas X/Y a ángulos. Recuerda que la pantalla del ordenador usa un eje Y invertido (los valores son mayores cuanto mas abajo estén).

Sintaxis: ASIN (Número)

Parametro: Número= número entero o flotante que representa un radio de la coordenada “Y” al desplazamiento tagencial.

ACOS (número)

Descripción: Esta función devuelve el arcocoseno del parametro especificado. Se utiliza para trasladar los valores de las coordenadas X/Y a ángulos. Recuerda que la pantalla del ordenador usa un eje Y invertido (los valores son mayores cuanto mas abajo estén).

Sintaxis: ACOS(Número)

Parametro: Número= número entero o flotante que representa un radio de la coordenada “x” al desplazamiento tagencial.

ATAN (flotante)

Descripción: Devuelve el ángulo desde el eje x hasta un punto (y,x). Se usa en la trigonometria básica, este comando te permitira derivar un angulo basado en la velocidad X e Y de un objeto en movimiento.

Sintaxis: ATAN(flotante)

Parametro: flotante = punto en valor flotante (grados).

ATAN2 (valorx,valory)

Descripción: Devuelve el ángulo desde el eje x hasta un punto (x,y). Se usa en la trigonometria básica, este comando te permitira derivar un angulo basado en la velocidad X e Y de un objeto en movimiento.

Sintaxis: ATAN2(Flotante)

Parametro: (valorx,valory)

---------- 3 PArte ----------

EXP (numero)

Descripción: Devuelve el valor de un logaritmo de 'e' elevado a una potencia. La función Exp() es una función exponencial, esta es la función inversa del logaritmo.

Sintaxis: EXP(numero)
Parametros: Numero: real o entero

```
Print Exp(1) ;Muestra el valor e^1
Print Exp(5) ;muestra el valor de e^5
```

Ejemplo 20. calculo del exponencial del número 1 y 5

LOG(numero)

Descripción: Devuelve el logaritmo natural del argumento especificado. Esta funcion es la inversa de la exponencial.

Sintaxis: Log(número)

Parametros: número=real o entero

```
;para encontrar el valor de Log(e^5)
e# = Exp(1)
x# = 5
y# = e#^x#
Print y; i.e.  e^5 = e*e*e*e*e: devuelve 148.413131
Print Log(y) ; ie. Log(e^5) = Log(10000): devuelve 5.000000
```

Ejemplo 21. uso de la función logaritmo

LOG10 (numero)

Descripción: Devuelve el logaritmo común del argumento especificado (base 10).

Sintaxis: Log10(número)

Parametros: número=real o entero

```
;para encontrar el valor de Log10(10^5)

x = 5
y = 10^x
Print y; i.e.  10^5 = 10*10*10*10*10: devuelve 100000
Print Log10(y) ; ie. Log10(10^5) = Log10(10000): devuelve 5.000000
```

Ejemplo 22 Calculo del Logaritmo

XOR

Descripción: Realiza un OR Exclusivo a nivel de bit entre dos valores. Se utiliza a menudo para encriptaciones ligeras, es decir sin necesidad de mucha seguridad, tomará dos valores y realizará un OR exclusivo con cada bit siguiendo las reglas básicas del XOR. Las reglas básicas son:

0 Xor 0 = 0
1 Xor 0 = 1

0 Xor 1 = 1
1 Xor 1 = 0

RND (Start#, end#)

Descripción: Devuelve un número real o entero generado aleatoriamente. El tipo de número dependerá de la variable a la que se lo asignes. Usando el comando SeedRnd, nos aseguramos que los números generados aleatoriamente no serán iguales cada vez que se ejecute el programa.

Parametros
start# = limite inferior
end#= limite superior

```
y=Rnd(0,10); Y contendrá un valor entero aleatorio entre 0 y 10
Print y
x#=Rnd(0,5) ; x contendrá un valor real entre 0.00000 y 10.00000
Print x
```

Ejemplo 23. Uso de la Función RND

RAND ([limite inferior] , Limite superior)

Descripción: Este comando se diferencia del comando RND, ya que devuelve solo valores enteros. El valor más bajo es por defecto 1 si no se especifica ninguno. El valor más alto es el limite del número generado. Si necesitas generar números flotantes, usa RND.

Parametros:

Limite inferior: este valor es opcional por defecto vale, representa el valor mas bajo.
Limite superior: Este valor representa el valor más alto.

```
;Establece un valor entero, llamado "semilla" para que cada vez que calcule un número aleatorio este no se repita.
SeedRnd (MilliSecs())

;genera 20 números aleatorios entre 1 y 100
For t=1 To 20
```

```
Print Rand(1,100)
Next
```

Ejemplo 24. Uso de la Función RAND

SeedRnd seed

Descripción: Los generadores de número aleatorios en los ordenadores no son realmente aleatorios. Generan números basados en un valor "semilla" (un número entero). Si no cambias esta 'semilla', siempre devolverá la misma serie de números. Usa este comando para asegurarte de que los números que obtienes son siempre distintos. Normalmente establecerás el valor 'semilla' al reloj del sistema para asegurarte de que es distinto en cada ejecución

En el ejemplo anterior se uso:

```
SeedRnd (MilliSecs())
```

para de esta forma cambiar la "semilla" con el tiempo de sistema actual en milesima de segundo.

FUNCIONES PARA TRABAJO CON CADENAS

El uso de cadenas es muy extendido en todos tipos de programas, no quedando excento de esto la programación de videojuegos es por ello que en este apartado vamos a explicar las diferentes funciones que trae blitz para el manejo de las mismas, de esta forma iremos preparandonos para el primer juego que realizaremos el cual se tratara de una aventura gráfica conversacional.

LEFT$ (cadena$, Longitud)

Descripción: Este comando sirve para obtener un determinado número de letras de una cadena de texto, empezando a contar por la izquierda.

Parametros

cadena$ = una cadena de texto valida
Longitud = un número entero que indica la longitud de la cadena a extraer.

```
nombre$ = "mario Bros"
Print "Las tres primeras letras de tu nombre son: " + Left$(nombre,3)
```

Ejemplo 26: Ejemplo del uso de la Función Left, para sacar los 3 primeros caracteres de una cadena.

RIGHT$ (cadena$, Longitud)

Descripción: Este comando sirve para obtener un determinado número de letras de una cadena de texto, empezando a contar por la derecha.

Parametros

cadena$ = una cadena de texto valida
Longitud = un número entero que indica la longitud de la cadena a extraer.

```
nombre$ = "mario Bros"
Print "Las tres ultimas letras de tu nombre son: " + Right$(nombre,3)
```

Ejemplo 27: Ejemplo del uso de la Función Right$ para sacar los 3 últimos caracteres de una cadena.

MID$ (cadena$, offset, carácteres)

Descripción: Este comando se utiliza para obtener una cantidad de caracteres del interior de una cadena de texto. Puedes elegir Donde se empezará a leer dentro de la cadena, y cuantos caracteres recogerás. Probablemente usarás este comando para descodificar una cadena y obtener cada carácter para realizar una conversión o una validación.

Parametros

Cadena$: Una cadena Valida
Offset: Este dato es de tipo entero.Posición dentro de la cadena para empezar a leer.
Caracteres: Este dato es de tipo entero. Cuantos caracteres se leeran desde la posición de comienzo

```
nombre$= "La Leyenda de Zelda"
For T=1 To Len(nombre$)
Print Mid$(nombre$,t,1)
Next
```

Ejemplo 28: Ejemplo del uso de la Función Mid$

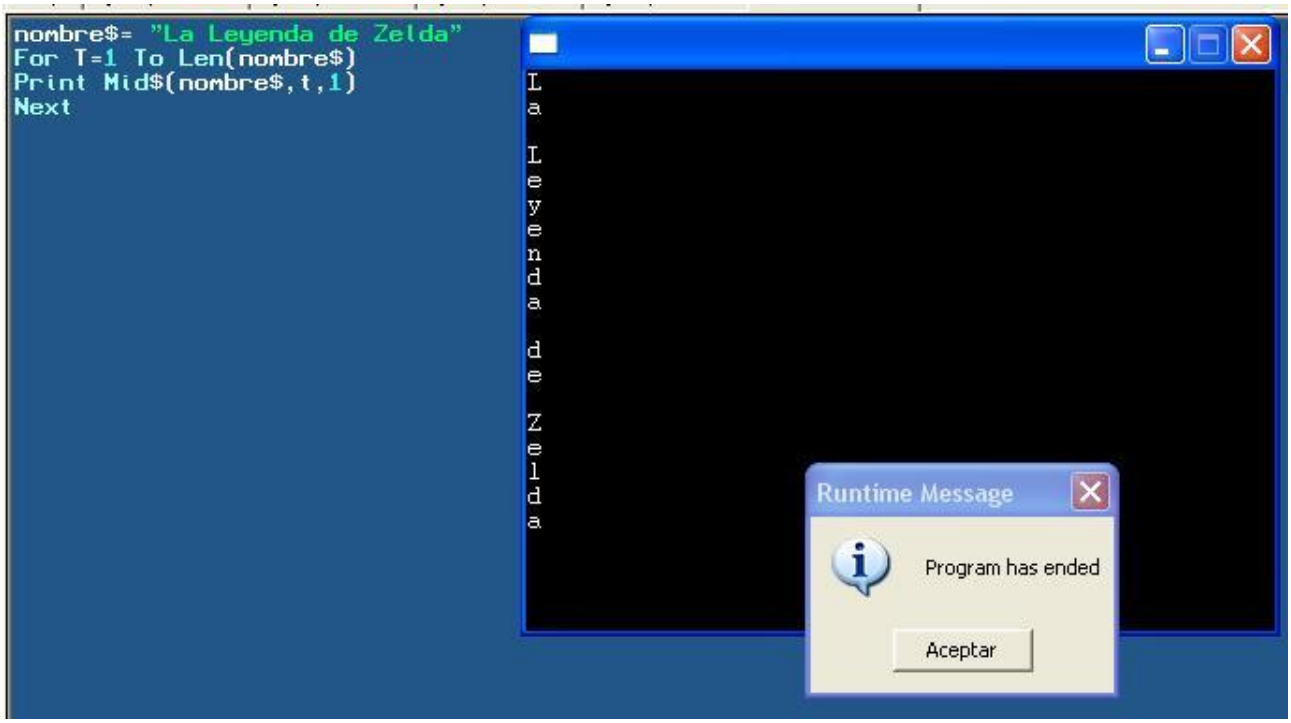


Fig. 15: aquí podemos observar el efecto que se realiza al ejecutar el programa escrito en el ejemplo 24.

Replace$ (cadena$, buscado$,Reemplazo$)

Descripcion: Este comando permitirá reemplazar caracteres dentro de una cadena dentro de otra. Úsalo para convertir letras de tus cadenas de texto (como quitar espacios o convertirlos en subrayados '_').

Parametros:

Cadena$ = cadena de Texto valida
buscado$ = cadena de Texto valida
Reemplazo$ = cadena de Texto valida

```
nombre$ = "Bruno Diaz"
Print "Tu nombre antes del reemplazo: " + nombre$
Print "Tu nombre con la R cambiada por la L: " + Replace$(nombre$,"r","L")
```

Ejemplo 29: Reemplazamos el carácter R por el L

Instr (cadena1$,cadena2$,offset)

Descripcion: Este comando permitirá buscar un texto en otro. El comando devolverá la localización (número de letras desde la izquierda) de la cadena que estás buscando. El comando devuelve 0 si no encuentra nada.

Parametros:

Cadena1$ = texto en el que quieres buscar
cadena2$ = texto que quieres buscar
offset$ = valor entero que indica la posición desde donde se comenzará a buscar (opcional)

```
nombre$ = "Juan perez"
Print nombre$
ubicacion= Instr(nombre$,"p",1)
Print "Tu nombre tiene la 'p' en la posición número" + ubicacion + " !"
```

Ejemplo 30: busqueda de un caracter en una cadena.

Upper$(cadena$)

Descripcion: Este comando toma una cadena y la convierte en mayúscula.

Parametros:

Cadena$ = cadena de Texto valida

```
nombre$ = "Gerson jeansalle"
Print "Tu nombre en mayúsculas es: " + Upper$(nombre$)
```

Ejemplo 31: convirtiendo un texto a mayúsculas

lower$(cadena$)

Descripcion: Este comando toma una cadena y la convierte en minusculas.

Parametros:

Cadena$ = cadena de Texto valida

```
nombre$ = "Gerson jeansalle"
Print "Tu nombre en minusculas es: " + lower$(nombre$)
```

Ejemplo 32: convirtiendo un texto a minusculas

Trim$ (cadena$)

Descripcion: Quita los espacios iniciales y finales de una cadena

Parametros:

Cadena$ = cadena de Texto valida

```
texto$ = "   Habia una vez   "
Print "Tu texto antes del arreglo es: " + texto$ + "..."
```

```
Print "Tu texto despues del arreglo es: " + Trim$(texto$) + "..."
```

Ejemplo 33: Quitando espacios al principio y final de un texto.

Len$ (cadena$)

Descripcion: Devuelve el número de caracteres de una cadena de texto

Parametros:

Cadena$ = cadena de Texto valida

```
nombre$= "Mortal Kombat"
Print "Hay " + Len(nombre$) + " letras en el nombre."
```

Ejemplo 34: La función Len cuenta la cantidad de caracteres que tiene un cadena de texto.

FUNCIONES DE TEXTO

Write cadena$

Descripcion: Escribe una cadena de texto en la pantalla, pero no cambia de línea (al contrario de lo que hace print)

Parametros:

Cadena$ = cadena de Texto valida

```
Write "Blitz "
Write "Basic"
```

Ejemplo 35: Uso de la función Write

Locate X,Y

Descripcion: Posiciona los comandos de texto en la pantalla

Parametros:
X = coordenada x de la pantalla.
Y = coordenada Y de la pantalla.

```
nombre$ = Input("¿Como te llamas? ")
```

```
Locate 100,200
Print "Hola, "+ nombre$
While Not KeyHit(1)
Wend
```

Ejemplo 36: En este ejemplo vemos como el saludo se ubica en la posición indicada por "Locate"

Text x,y,cadena$, [centrado X], [Centrado Y]

Descripcion: Imprime un texto en las coordenadas de la pantalla especificadas. Puedes centrar el texto en las coordenadas estableciendo "Center x / center y" en True. El texto se dibujará en el color actual del dibujo.

Parametros:
X = coordenada x para comenzar a escribir
Y = coordenada Y para comenzar a escribir
Cadena$= una cadena Valida
Center X= Opcional; true = Centra horizontalmente
Center Y= Opcional; true = Centra Verticalmente

```
Graphics 800,600,16

While Not KeyHit(1)
Text 400,0,"Hola Mundo",True,False
Wend
```

Ejemplo 37: Aquí vemos el uso de la función "Text" para ubicar el texto en las coordenadas elegidas.

LoadFont (fontname$, height, bold, italic, underlined)

Descripcion: Carga una fuente para poder ser utilizada.

Parametros:
Fontname$ - nombre de la fuente a cargar, por ejemplo "arial"
Height – altura en pixeles de la fuente
Bold – true para cargarla en negrita, false para el caso contrario
Italic – true para cargarla en cursiva, false para el caso contrario
underline – true para cargarla en subrayada, false para el caso contrario

```
; Activa el modo Gráfico
Graphics 800,600,16
; Crea las variables globales para las fuentes
Global fntArial,fntArialB,fntArialI,fntArialU

; Carga las fuentes
fntArial=LoadFont("Arial",24,False,False,False)
fntArialB=LoadFont("Arial",18,True,False,False)
```

```
fntArialI=LoadFont("Arial",32,False,True,False)
fntArialU=LoadFont("Arial",14,False,False,True)
; Establece la fuente e imprime el texto

SetFont fntArial
Text 400,0,"Esta es la Arial 24 puntos",True,False

SetFont fntArialB
Text 400,30,"Esta es la Arial 18 puntos negrita",True,False

SetFont fntArialI
Text 400,60,"Esta es la Arial 32 puntos cursiva",True,False

SetFont fntArialU
Text 400,90,"Esta es la Arial 14 puntos subrayada",True,False
; Espera a que el usuario pulse ESC para salir

While Not KeyHit(1)
Wend
; ¡Borra todas las fuentes de la memoria!
FreeFont fntArial
FreeFont fntArialB
FreeFont fntArialI
FreeFont fntArialU
```

Ejemplo 38: Aquí vemos un programa utilizando la loadFont, para poder ver un texto con diferentes características.

SetFont Fonthandle

Descripción: Establece una fuente cargada previamente para operaciones de escritura

Parametros:
Fontname$ - Cadena de texto que contiene el nombre de la fuente.
Height – altura en pixeles de la fuente
Bold – true para cargarla en negrita, false para el caso contrario
Italic – true para cargarla en cursiva, false para el caso contrario
underline – true para cargarla en subrayada, false para el caso contrario

En el **ejemplo 38** se observa el uso de esta función

Muy bien hasta aquí tenemos bastante información como para comenzar a crear nuestro primer videojuego.

JUEGO 1: Aventura Conversacional – Indiana Jones and The Tomb of Pharaon

Como primera medida necesitaremos realizar un pequeño análisis y diseño del videojuego que construiremos.

El juego que haremos será una aventura conversacional, en que consiste?
Veamos que nos dice la wikipedia al respecto:

aventura conversacional es un juego en el que la descripción de la situación en la que se encuentra el jugador proviene principalmente de un texto. A su vez, el jugador debe teclear la acción a realizar. El juego interpreta la entrada -normalmente- en lenguaje natural, lo cual provoca una nueva situación y así sucesivamente. A veces existen gráficos en estos juegos, que sin embargo son tan sólo situacionales o que ofrecen ayuda complementaria en algunos casos.

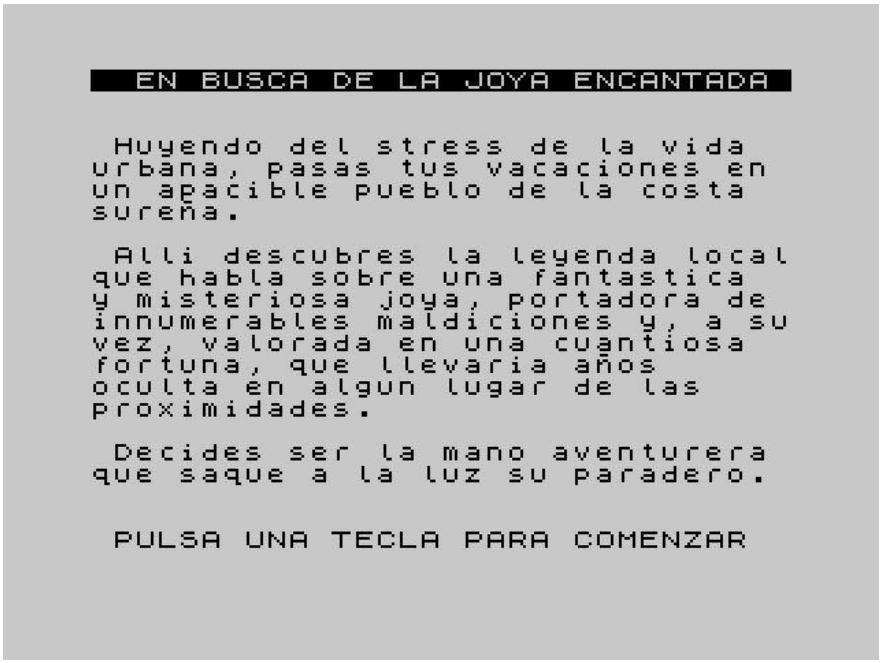


Fig. 16: Ejemplo de Aventura conversacional usando solo texto.

Fig. 17: Ejemplo de Aventura conversacional usando texto e imágenes.

Muy bien según la definición ya tenemos un poco más claro que es una aventura conversacional, se debe tener en cuenta que para esta clase de juegos mucho la parte grafica no interesa, pero si la historia, los problemas a resolver y la descripción de determinadas áreas las cuales ayudarán al jugador a resolver los puzzles.

Como excusa para el curso de programación de videojuegos, crearemos un juego de indiana Jones, pero antes de comenzar a escribir el código será fundamental que nos organicemos:

Cosas que debemos hacer

1. Crear la historia, los personajes los lugares que el héroe visitará.
2. Crear las áreas y un pequeño mapa de conexión entre las mismas
3. crear los ítems que el jugador podrá recoger
4. crear los puzzles que deberá resolver el jugador.

Historia

Haremos una historia corta, que nos permitirá ver los fundamentos de la programación de aventuras conversacionales.

Todo comienza con un llamado a Indiana Jones de su buen amigo Marcus Brody este le pide que urgentemente lo vea en su museo.

El puzzle en este caso tiene que llamar por telefono un taxi, para que lo lleve hasta el museo
Comando: Usar Teléfono, despliega 3 opciones

- A marcus Brody
- A una chica de las anteriores películas
- Taxi

Debe examinar la mesa de luz y tomar tu billetera, ella contiene el dinero para pagarle al taxista.

Una vez que se sube al taxi debe hablar con el taxista y decirle a donde quiere ir:

- A ver a marcus
- A ver a tu chica
- No sabe a donde ir

Cuando llega al museo, el taxista le pedira el dinero, deberá primero abrir la billetera (De esta forma puede obtener el dinero) y luego debe dar billete a taxista.

Cuando indiana llega al museo Marcus le cuenta la historia de que los nazis se encuentran en Egipto buscando un artefacto, el cual creen que tiene poderes extraordinarios, y ellos mismos piensan que esta en la tumba del Faraón Keops

Se debe utilizar Hablar Marcus, entonces Marcus comienza a hablar. El dialogo es el siguiente:

MB - Indy, debes ir urgente a Egipto, los nazis están cerca de apoderarse del anillo de Keops, según la leyenda dice que aquel que posea ese anillo tendrá poderes increíbles y podrá dominar a cualquier ser viviente de la tierra,

Indiana puede elegir 3 respuestas:

- Sígueme contando que me interesa la historia
  - En este caso Marcus Cuenta lo siguiente
    " Los arqueólogos cuando descubrieron las cámaras secretas en la pirámide de Keops, pensaron que descubrirían la momia de Keops, pero no fue así, según parece esta en un lugar distante a esa pirámide o quizás en la pirámide pero en una cámara aún mas secreta que la anterior, ahora los nazis están buscando la tumba donde yace la momia de Keops, porque creen que en ella se encuentra el anillo.

    IJ: Marcus parece que estamos frente a una nueva aventura, saca los pasajes para que ya mismo emprendamos el viaje a "El Cairo".
- No perdamos mas tiempo, sácame los pasajes para ir a "El Cairo"
  - Ok indy muy bien!! Toma los había sacado con anterioridad porque sabia de tu ansiedad!
- Oye Marcus... de nuevo con las fábulas! (umhhhhhh tan aventurero que salió y se pierde en su propio museo)
  - Pero indy es cierto, sino mira este telegrama del jefe del ejercito de E.E.U.U, pidiendo nuestra colaboración en este evento.
  - Ok tenias razón, necesito que me saque el pasaje para el cairo

Aeropuerto:

En el aeropuerto se ve un gran tumulto de personas que van de un lado al otro, indiana se debe dirigir al embarque para el cairo, esto es Ir NORTE, IR NORTE , IR OESTE, IR OESTE, IR NORTE. Si le va preguntando a las personas estas le van indicando hacia donde ir.

Una vez que llega Indy debe mostrar el pasaje en la ventanilla de embarque, para poder pasar al avión.

Y luego muestra un avión en viaje y finaliza la demo!

Bien para hacer el juego de toda la historia contada solo haremos hasta que el taxi llega a buscar a indy, todo lo demás quedará para tarea para el hogar =).

Antes de empezar con la programación veremos algunas funciones que utilizaremos y que no fueron explicadas con anterioridad.

Estas son

- Color
- Drawimage
- Loadimage
- Loadsound

Y además veremos como crear nuestras propias funciones

---------- Parte 4 ----------

Código completo del Juego

```
Graphics 800,600
Global inicio
Global panelder,panel
Global fuente,fuenteder

;array para guardar las imagenes del nivel1
Dim nivel1$(5)
;me cuenta la cantidad de intentos para que me muestre dialogos diferentes cuando el jugador le erra
intento=0

;cargamos las fuentes a utilizar
Fuenteder = LoadFont ("adventure",12,True,False,True)
Fuente = LoadFont ("adventure",12,False,False,False)
;Cargamos las Imagenes del nivel1, usamos un array para realizar esta acción

nivel1$(0)=LoadImage("img/pantallas/pantalla1/hab1.jpg")
nivel1$(1)=LoadImage("img/pantallas/pantalla1/tel.jpg")
nivel1$(2)=LoadImage("img/pantallas/pantalla1/marcus.png")
nivel1$(3)=LoadImage("img/pantallas/pantalla1/taxi.jpg")



;Cargamos las imagenes para utilizar al comienzo y en la interface grafica.
panel=LoadImage("img/panel.bmp")
panelder=LoadImage("img/paneldere.bmp")
momia=LoadImage("img/momia.bmp")
inicio=LoadImage("img/inicio0.bmp")
inicio1=LoadImage("img/inicio.bmp")
inicio2=LoadImage("img/portadaindy2.bmp")
musica = LoadSound("musica/indiana.mp3")

PlaySound musica
```

```
;modo debug salto nivel, esto me permite chequear los niveles avanzados.
nivel=Input("Elija Nivel: ")
If nivel= 0
      Goto nivel0
ElseIf nivel=1
           Goto nivel1
Else
           Goto nivel2

EndIf

;Imagenes del nombre de la compania y presentación del juego
.nivel0
DrawImage inicio,0,0
Delay 3700

DrawImage inicio1,0,0
Delay 7000

Cls
DrawImage inicio2,0,0
WaitKey

;Bucle principal
While Not KeyHit(1)
.nivel1
;inicio el backbuffer y el frontbuffer
SetBuffer FrontBuffer()
SetBuffer BackBuffer()
;Borro lo que hay en pantalla
Cls
;comienzo por la pantalla1
dibujarpaneles(nivel1$(1),50,50)

;Escribo el texto en pantalla
Text 50,430,"Ring Ring Ring!!!"

Locate 50,480
;muestro el prompt y espero que la persona escriba un comando para guardarlo
res$=Input("> ")

;si el comando escrito es tomar telefono entonces pasa al nivel 2
If res$ = "tomar telefono"
  .nivel2
;vamos a la pantalla 2
      Goto pantalla2
```

```
    ElseIf intento = 0

    Text 50,500, "mmmmmmm esto no lo puedes hacer"
    ;mostramos el mensaje por dos segundos
    Delay 2000
    ;usamos una bandera para mostrar diferentes mensajes
    intento=1
    Else
        Text 50,500,"Oyeeeee porque no dejas de probar cosas y atiendes el telefono?"
        intento=0

    Delay 2000

EndIf

.pantalla2
    Cls
    dibujarpaneles(nivel1$(2),50,15)

    Text 50,430,"Ufff indiana menos mal que atendiste! necesito que vengas "
    Text 50,445,"urgente al museo, es cuestion de vida o muerte"
    SetFont fuente
    ;cambio el color de la fuente
    Color 100,155,50
    Text 50,460,"1 - Pero que ocurre marcus?"
    Text 50,475,"2 - Estas loco marcus, recien me acuesto de una noche dificil?"
    Text 50,490,"3 - Marcus,luego nos vemos tengo que preparar el examen para mañana!"
    Locate 50,510
    res$=Input("> ")

    If res$=1
        Cls
        dibujarpaneles(nivel1$(2),50,15)
      Color 0,0,0
        Text 50,430,"Los Nazis estan realizando excavaciones en el cairo "
        Text 50,445,"se encuentran buscando una reliquia milenaria la cual"
        Text 50,460,"otorga poderes a aquel que la posea, necesito que vengas"
        Text 50,475,"al museo"
        ElseIf res$=2
            Cls
            dibujarpaneles(nivel1$(2),50,15)
        Color 0,0,0
            Text 50,430,"Oye indy, en verdad que es importante tomate un "
            Text 50,445,"cafe y ven urgente"
            Else
            Cls
                dibujarpaneles(nivel1$(2),50,15)
            Color 0,0,0

                Text 50,430,"Esperaaa!!!! si no vienes urgente la humanidad corre "
```

```
                Text 50,445,"Peligro!!!, tomales el examen otro dia esto en verdad es"
                Text 50,460,"importante!"

EndIf
Locate 50,495
res$=Input("> ")
If res$= "ok"
    Goto pantalla3
Else
   Goto pantalla2
EndIf
;aqui comienza la pantalla pantalla 3
.pantalla3

Cls
    dibujarpaneles(nivel1$(0),50,15)

  Color 0,0,0
    Text 50,430,"Muy bien es mejor ir urgente al museo"
    Text 50,445,"para que marcus me pueda contar con lujos de detalle "
    Text 50,460,"lo que esta ocurriendo"
    Text 50,475,"justo mi auto esta en el taller y debo llegar lo mas rapido"
    Text 50,490,"posible"
        .inicio3
    Locate 50,500

    res$=Input("> ")
    If res$="tomar telefono"
    .tele
        Cls
        dibujarpaneles(nivel1$(0),50,15)

        Text 50,430,"mmmmmmm a ver a quien podre llamar que me ayude"

        Color 100,155,50
        Text 50,445,"1 - Llamar a MArcus"
        Text 50,460,"2 - Llamar a Marlyn waitt"
        Text 50,475,"3 - Llamar un taxi"
        Locate 50,500
        res$=Input("> ")
            If res$=1
                Cls
                dibujarpaneles(nivel1$(2),50,15)
            Color 0,0,0
                Text 50,430,"Oye indy todavia estas ahi, no te puedo explicar "
                Text 50,445,"por el telefono ven urgente!!!"
                Delay 5000
                Goto tele
            ElseIf res$=2
                Cls
```

```
        Color 0,0,0
        Text 50,430,"Indiana Jones, luego de lo de anoche no quiero "
        Text 50,445,"ni verte"
        Delay 5000
        Goto tele

    Else
        dibujarpaneles(nivel1$(0),50,15)
        Color 0,0,0
        Text 50,430,"Ok senor jones en 10 minutos le enviamos un coche"
        ;pone el puzle 1 en 1 para decir que se activo
        prueb1=1
        Goto inicio3
    EndIf


    .cajon
    ElseIf res$="abrir cajon"
        Cls
        dibujarpaneles(nivel1$(0),50,15)
            Color 0,0,0
        Text 50,445,"El cajon fue abierto y se observa una Billetera"
        prueb3=1
        Goto prueba3

            Else
               Cls
                  dibujarpaneles(nivel1$(0),50,15)

                  Color 0,0,0
                  Text 50,445,"Oye eso no sirve"
                  Goto inicio3




            .prueba3
            Locate 50,460
            res$=Input("> ")
            If (res$="tomar billetera" And prueb3=1)
                Cls
                dibujarpaneles(nivel1$(0),50,15)
                Color 0,0,0
                Text 50,475,"Muy bien indy es necesario para pagar el taxi"
                prueb2 = 1
                Goto tiempo
                Else
                    Cls
                    dibujarpaneles(nivel1$(0),50,15)
                    Color 0,0,0
```

```
                                    Text 50, 430,"Oye estas loco?? eso no se puede hacer!"
                                    Goto prueba3


                EndIf

        EndIf


        .tiempo
If (prueb1=1 And prueb2=1 )
        ;truco para que pase el tiempo
    WaitKey
        Cls
        dibujarpaneles(nivel1$(0),50,15)
        Text 50,430,"pasaron 5 minutos"

        WaitKey
        Cls
        dibujarpaneles(nivel1$(0),50,15)
        Text 50,430,"pasaron 10 minutos"
        WaitKey
        Cls
        dibujarpaneles(nivel1$(3),50,15)
        Text 50,430,"PIIII PIIIIII"
        Text 50,445,"Al fin llego el taxi"
        WaitKey
        Goto pantalla4

        EndIf
    .pantalla4

        Cls

        Text 50,430,"FIN????? "







    WaitKey



Flip
Wend

.fin
```

```
WaitKey

;Función que permitira dibujar los paneles e imagenes del juego

Function dibujarpaneles(imagen,AHorizontal,AVertical)

DrawImage imagen,AHorizontal,AVertical
DrawImage panel,10,415
DrawImage panelder,615,10

SetFont fuenteder
Text 625,50,"Abrir"
Text 730,50,"usar"

SetFont fuente
;menú Abrir
Text 625,60,"tomar"
Text 625,70,"golpear"

;Menú Usar
Text 730,60,"Hablar"
Text 730,70,"empujar"

End Function
```

Paso a Paso: El código explicado

Muy bien, ahora pasaremos a explicar cada parte del código para que sepamos que es lo que hace (o debería hacer) el juego.

```
Graphics 800,600
Global inicio
Global panelder,panel
Global fuente,fuenteder

;array para guardar las imagenes del nivel1
Dim nivel1$(5)
;me cuenta la cantidad de intentos para que me muestre dialogos diferentes cuando el jugador le erra
intento=0
```

Lo primero que hacemos es inicializar el modo grafico en 800X600, luego crearemos variables globales las cuales utilizaremos a lo largo del programa: Las variables que inicializamos son: Inicio

(para mostrar la pantalla del creador del juego), Panelder y Panel (Tendrá la imágenes de los paneles gráficos), Declaramos un array llamado nivel1, donde guardaremos las imágenes para mostrar en el nivel 1, también iniciamos una variable contador llamada intento la cual contará las veces que una persona pregunto una misma cosa, esto servirá para que la computadora arroje distintas respuestas simulando una inteligencia artificial.

```
;cargamos las fuentes a utilizar
Fuenteder = LoadFont ("adventure",12,True,False,True)
Fuente = LoadFont ("adventure",12,False,False,False)
;Cargamos las Imagenes del nivel1, usamos un array para realizar esta acción

nivel1$(0)=LoadImage("img/pantallas/pantalla1/hab1.jpg")
nivel1$(1)=LoadImage("img/pantallas/pantalla1/tel.jpg")
nivel1$(2)=LoadImage("img/pantallas/pantalla1/marcus.png")
nivel1$(3)=LoadImage("img/pantallas/pantalla1/taxi.jpg")


;Cargamos las imagenes para utilizar al comienzo y en la interface grafica.
panel=LoadImage("img/panel.bmp")
panelder=LoadImage("img/paneldere.bmp")
momia=LoadImage("img/momia.bmp")
inicio=LoadImage("img/inicio0.bmp")
inicio1=LoadImage("img/inicio.bmp")
inicio2=LoadImage("img/portadaindy2.bmp")
musica = LoadSound("musica/indiana.mp3")

PlaySound musica
```

Lo siguiente que hacemos es Cargar las fuentes, utilizando la instrucción LoadFont, también cargamos el array con las imágenes que utilizaremos en el primer nivel, para realizar esta tarea deberemos utilizar la función LoadImage, por último cargamos el tema de fondo utilizando Load Sound y luego ejecutamos la música utilizando PlaySound.

```
;modo debug salto nivel, esto me permite chequear los niveles avanzados.
nivel=Input("Elija Nivel: ")
If nivel= 0
      Goto nivel0
ElseIf nivel=1
            Goto nivel1
Else
            Goto nivel2

EndIf
```

Lo que hacemos en este código es una especie de herramienta propia para saltar los niveles sin tener que ejecutar el programa desde el principio, esto sirve para poder corregir y probar los diferentes niveles.

El programa en si es muy sencillo, lo que hace es preguntarle al usuario que ingrese el nivel al que desea ir, luego dependiendo del nivel saltamos a un sector determinado.

```
;Imagenes del nombre de la compania y presentación del juego
.nivel0
DrawImage inicio,0,0
Delay 3700

DrawImage inicio1,0,0
Delay 7000

Cls
DrawImage inicio2,0,0
WaitKey
```

En esta parte del código, lo único que hacemos es mostrar las imágenes de la compania y de la presentación del juego, para mostrar una imagen utilizamos la instrucción DrawImage, luego hacemos un delay para que la imagen se muestre por un instante después del delay, (retardo) pasa a la siguiente imagen.

```
;Bucle principal
While Not KeyHit(1)
.nivel1
;inicio el backbuffer y el frontbuffer
SetBuffer FrontBuffer()
SetBuffer BackBuffer()
;Borro lo que hay en pantalla
Cls
;comienzo por la pantalla1
dibujarpaneles(nivel1$(1),50,50)

;Escribo el texto en pantalla
Text 50,430,"Ring Ring Ring!!!"

Locate 50,480
;muestro el prompt y espero que la persona escriba un comando para guardarlo
res$=Input("> ")

;si el comando escrito es tomar telefono entonces pasa al nivel 2
If res$ = "tomar telefono"
   .nivel2
;vamos a la pantalla 2
      Goto pantalla2



      ElseIf intento = 0
```

```
    Text 50,500, "mmmmmmm esto no lo puedes hacer"
    ;mostramos el mensaje por dos segundos
    Delay 2000
    ;usamos una bandera para mostrar diferentes mensajes
    intento=1
    Else
        Text 50,500,"Oyeeeee porque no dejas de probar cosas y atiendes el telefono?"
        intento=0

    Delay 2000

EndIf
```

Aquí comienza el bucle principal de juego, debemos iniciar el frontbuffer y backbuffer para poder mostrar los gráficos, para hacer esto utilizamos la instrucción:

```
SetBuffer FrontBuffer()
SetBuffer BackBuffer()
```

En la siguiente instrucción dibujaremos el panel, para eso utilizamos la función creada por nosotros (más adelante explicamos esta función)

```
dibujarpaneles(nivel1$(1),50,50)
```

Una vez dibujado el panel, mostramos el texto en las coordenadas 50,580

```
Text 50,430,"Ring Ring Ring!!!"
```

Luego ubicamos el cursos (>) en la posición 50,480 además esperamos que la persona escriba alguna instrucción.

```
Locate 50,480
res$=Input("> ")
```

lo que la persona escribe se guardará en la variable res$

Ahora consultamos la respuesta que ingreso el jugador

```
If res$ = "tomar telefono"
```

si la persona escribió “Tomar Télefono” entonces va a la pantalla 2

```
Goto pantalla2
```

Sino si intento es igual a 0

```
ElseIf intento = 0
```

Muestra el texto indicador de que el jugador ingreso mal una instrucción.

```
    Text 50,500, "mmmmmmm esto no lo puedes hacer"
```

Mostramos el mensaje por dos segundos

```
    Delay 2000
```

Ponemos intento en 1, para que me muestre un texto diferentes cuando el usuario se vuelva a equivoca.

```
    intento=1
```

Si intento es igual a 1 me muestra un mensaje diferente. Y volvemos a poner la variable intento en 0.

```
Else
        Text 50,500,"Oyeeeee porque no dejas de probar cosas y atiendes el telefono?"
        intento=0
```

Código correspondiente a la pantalla 3

Muy bien ahora mostraremos el código de la pantalla 3, ya que este es la parte más compleja y vamos a poder ver el manejo de puzles

Dibujamos el nivel

```
dibujarpaneles(nivel1$(0),50,15)
```

Asignamos el color del texto

```
Color 0,0,0
```

Mostramos el texto.

```
Text 50,430,"Muy bien es mejor ir urgente al museo"
        Text 50,445,"para que marcus me pueda contar con lujos de detalle "
        Text 50,460,"lo que esta ocurriendo"
        Text 50,475,"justo mi auto esta en el taller y debo llegar lo mas rapido"
        Text 50,490,"posible"
```

Bueno, la siguiente parte no la explicamos, pero a esta altura deberías saber que es lo que hace ;)

```
Locate 50,500

        res$=Input("> ")
        If res$="tomar telefono"
        .tele
            Cls
            dibujarpaneles(nivel1$(0),50,15)

            Text 50,430,"mmmmmmm a ver a quien podre llamar que me ayude"
```

Indicamos el color del texto

```
Color 100,155,50
```

Ahora damos 3 opciones para que el jugador pueda llamar

```
    Text 50,445,"1 - Llamar a MArcus"
            Text 50,460,"2 - Llamar a Marlyn waitt"
            Text 50,475,"3 - Llamar un taxi"
```

Damos la opción de que la persona elija a quien llamar, dependiendo de la persona que desea llamar será el texto que mostrará.

```
Locate 50,500
            res$=Input("> ")
                If res$=1
                    Cls
                    dibujarpaneles(nivel1$(2),50,15)
                Color 0,0,0
                    Text 50,430,"Oye indy todavia estas ahi, no te puedo explicar "
                    Text 50,445,"por el telefono ven urgente!!!"
                    Delay 5000
                    Goto tele
                ElseIf res$=2
                    Cls
                    Color 0,0,0
                    Text 50,430,"Indiana Jones, luego de lo de anoche no quiero "
                    Text 50,445,"ni verte"
                    Delay 5000
                    Goto tele

                Else
```

Si selecciona la tercer opción lo lleva a la siguiente pantalla

```
dibujarpaneles(nivel1$(0),50,15)
                    Color 0,0,0
                    Text 50,430,"Ok senor jones en 10 minutos le enviamos un
coche"
```

Ponemos el indicador Prueb1 en uno para avisar que se activo el primer puzzle

```
prueb1=1
                Goto inicio3
                    EndIf
```

Ahora el jugador deberá escribir “abrir cajón para buscar la billetera.

```
ElseIf res$="abrir cajon"
                        Cls
                        dibujarpaneles(nivel1$(0),50,15)
                            Color 0,0,0
                        Text 50,445,"El cajon fue abierto y se observa una Billetera"
```

Una vez que el cajón es abierto, ponemos la prueb3 en uno para indicar que paso el puzzle

```
prueb3=1
```

y luego nos dirijimos a la etiqueta prueba3.

```
Goto prueba3

                        Else
                           Cls
                             dibujarpaneles(nivel1$(0),50,15)

                             Color 0,0,0
                             Text 50,445,"Oye eso no sirve"
                             Goto inicio3
```

Aquí inicia la etiqueta Prueba 3

```
.prueba3
                        Locate 50,460
                        res$=Input("> ")
```

Ahora nos fijamos si la prueb3 = 1 y el usuario escribio tomar billetera entonces mostrará el mensaje indicando que realice una operación valida

```
If (res$="tomar billetera" And prueb3=1)
                             Cls
                             dibujarpaneles(nivel1$(0),50,15)
                             Color 0,0,0
                             Text 50,475,"Muy bien indy es necesario para pagar el
taxi"
```

Prueb2 le ponemos 1 (para ya sabes que)

```
prueb2 = 1
```

Nos dirijimos a la etiqueta tiempo.

```
Goto tiempo
```

Sino va a mostrar el mensaje incorrecto.

```
                                Else
                                    Cls
                                    dibujarpaneles(nivel1$(0),50,15)
                                    Color 0,0,0
                                    Text 50, 430,"Oye estas loco?? eso no se puede
hacer!"
                                    Goto prueba3
                            EndIf
                    EndIf
```

ETIQUETA TIEMPO

```
.tiempo
```

Si cumplimos con la prueba 1 y 2

```
If (prueb1=1 And prueb2=1 )
```

Hacemos que pase el tiempo, cada vez que presionamos una tecla

```
WaitKey
        Cls
        dibujarpaneles(nivel1$(0),50,15)
        Text 50,430,"pasaron 5 minutos"

        WaitKey
        Cls
        dibujarpaneles(nivel1$(0),50,15)
        Text 50,430,"pasaron 10 minutos"
```

A la tercera vez que presionamos una tecla el taxi llega

```
WaitKey
        Cls
        dibujarpaneles(nivel1$(3),50,15)
        Text 50,430,"PIIII PIIIII"
        Text 50,445,"Al fin llego el taxi"
        WaitKey
        Goto pantalla4

        EndIf
```

Muy bien aquí hemos podido ver como se puede manejar el tema de los puzles, para que pase a un determinado lugar, es importante el tener un buen diseño del juego ya que es muy probable que sino nos mareemos entre tantas bifurcaciones, fíjense que nuestro juego tiene apenas 3 pantallas, si hubiéramos tenido 50 sería un caos.

Ahora Crearemos la función que permite dibujar los paneles.

Los parámetros que usamos son **Image** (esta recibe la dirección de la imagen) y **Ahorizontal** y **Avertical**, con lo cual ingresamos las coordenadas para alinear la imagen

Function dibujarpaneles(imagen,AHorizontal,AVertical)

Dibujamos la imagen

DrawImage imagen,AHorizontal,AVertical

Dibujamos el panel inferior e izquierdo

```
DrawImage panel,10,415
DrawImage panelder,615,10
```

Elegimos la fuente para los títulos del panel de la derecha y escribimos los commandos que se pueden utilizar

```
SetFont fuenteder
Text 625,50,"Abrir"
Text 730,50,"usar"
```

Ingresamos los ítems dentro del menú abrir

```
SetFont fuente
;menú Abrir
Text 625,60,"tomar"
Text 625,70,"golpear"

;Menú Usar
Text 730,60,"Hablar"
Text 730,70,"empujar"

End Function
```

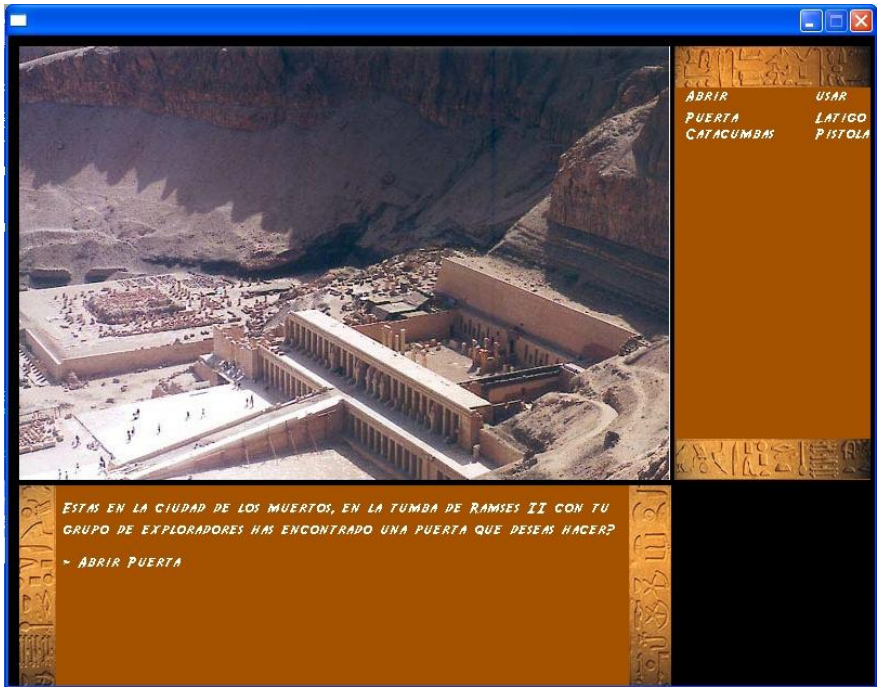


FIG. Screenshot de nuestro juego

En el próximo capítulo veremos algunas funciones graficas que usamos en este juego que no hemos explicado anteriormente.