1 Tipos de datos.

Se ha definido el tipo Pict como puntero a una estructura de tamaño reducido que contiene toda la información necesaria para poder manejar las imágenes deseadas. Esta definición, así como el resto de definiciones y prototipos, se encuentra en el fichero "pict.h".

También se encuentran definidos los tipos de datos correspondientes a los códigos de operación utilizados[1], aunque por lo general serán transparentes al usuario de la biblioteca de funciones.

2 Funciones de acceso a disco.

Las funciones de alto nivel programadas para leer y escribir ficheros PICT2 son las siguientes:

```
Pict   pict_open(char *,int,int,int,int,C_MAP);
int    pict_put_line(Pict,U_char *);
int    pict_get_line(Pict,U_char *);
int    pict_close(Pict);
int    pict_free(Pict);
Pict   pict_read_pict(char *);
Pict   pict_read_pict_c(char *);
```

La función pict_open deberá llamarse antes de abrir cualquier imagen (tanto para lectura como para escritura) que se desee procesar. En caso de que se desee leer de disco, tras esta llamada se tendrá una estructura Pict con todos sus campos inicializados a los valores correctos (excepto el puntero a pixels, que se reserva para uso del programador); mientras que en caso de escritura, con esta llamada se creará todo el encabezamiento del archivo en el formato adecuado al tipo de imagen.

Tras esta llamada el fichero está listo para leer (escribir) linea por linea la imagen, lo que puede hacerse mediante pict_get_line (pict_put_line).

---

[1] Estas estructuras corresponden byte a byte con el contenido del archivo PICT2 asociado a la imagen, por ello debe verificarse que sizeof(short) = 2 bytes y que sizeof(long int) = 4 bytes..

Una vez finalizado el proceso, pict_close coloca el EndOfFile si es necesario, cierra el archivo, y pict_free libera la memoria utilizada por la estructura Pict utilizada.

Como alternativa a esta secuencia, para lectura, se dispone de pict_read_pict; que con una sola llamada carga toda la imagen (utilizando el campo "pixels"). En esta llamada, se almacenan de forma consecutiva los pixels completos, mientras que con pict_read_pict_c, son las lineas de cada canal las que aparecen como elementos diferenciados.

3 Funciones auxiliares.

Las siguientes funciones permiten el acceso a los campos de la estructura Pict sin necesidad de tener conocimiento de la composición de dicha estructura:

```
int     pict_height(Pict);
int     pict_width(Pict);
int     pict_mode(Pict);
int     pict_n_comp(Pict);
U_char      *pict_pixels(Pict);
```

Todas ellas devuelven el valor correspondiente a su nombre.

```
void  pict_pixel_rgba();
```

Para el caso de que se hubiera empleado pict_read_pict(), esta función permite el acceso al color de cada pixel por sus coordenadas.

Para el manejo de imágenes con tabla de colores se dispone de tres funciones adicionales:

```
Pict   pict_24_to_8(Pict, int, int);
int    pict_get_map(Pict, C_MAP);
int    pict_set_map(Pict, C_MAP);
```

La función pict_24_to_8 (adaptada de la librería AlephTracer) convierte un imagen de color real a 256 colores con minimizando la pérdida de exactitud.

4 Funciones de uso interno.

Estas funciones pueden ser utilizadas para programar a un nivel más bajo; y son las utilizadas internamente por la biblioteca, aunque por lo general serán transparentes al programador.

```
S_int       pack_bytes(U_char *,S_int,FILE *);
S_int       unpack_bytes(U_char *,S_int,FILE *);

int   pict_write_header_9A(Pict,US_int,US_int,US_int);
int   pict_read_header_9A(Pict);
int   pict_write_header_98(Pict,US_int,US_int,US_int,C_MAP);
int   pict_read_header_98(Pict,C_MAP);
int   pict_ctable_98(Pict,C_MAP);
```

La descripción de todas ellas se encuentra en el código fuente.

5 Referencia.

A continuación se detalla la guía de uso de las librerías desarrolladas.

"pict.h"

DESCRIPCION

Funciones para el manejo ficheros PICT2.

TIPOS

U_char	US_int	Pict	tHeader
S_int	C_MAP	Rect	tclipRect
L_int	PictStruct	fixedpoint	colorTable
tpixMap9A	tpixMap98	Header9A	Header98

MACROS

PICT_H	PICTF	M_MONO
P_READ	PICTR	M_RGB8
P_WRITE	PICTG	M_RGB16
GRAYSCALE	PICTB	M_RGB24
OPTIMAL	PICTA	M_RGBA32
M_ALPHA		

FUNCIONES

pict_read_pict()	pict_read_pict_c()	pict_24_to_8()
pict_get_map()	pict_set_map()	pict_height()
pict_width()	pict_mode()	pict_n_comp()
pict_pixels()	pict_pixel_rgba()	pict_put_line()
pict_get_line()	pict_close()	pict_free()
pack_bytes()	unpack_bytes()	
pict_write_header_9A()	pict_read_header_9A()	
pict_write_header_98()	pict_read_header_98()	
pict_ctable_98()		

pict_read_pict, pict_read_pict_c

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
Pict  pict_read_pict(char *file);
Pict  pict_read_pict_(char *file);
```

DESCRIPCION

Lee el fichero PICT2 cuyo nombre viene dado por **file* y asigna dinámicamente una variable PictStruct, inicializándola con los valores correspondientes al archivo.

Además asigna dinámicamente el campo 'pixels' de dicha estructura, almacenando en este espacio de memoria los valores correspondientes a los colores de los pixels de la imagen.

pict_read_pict:

Los datos se almacenan por lineas de pixels, ajustándose los pixels al siguiente esquema:

M_MONO, M_RGB8:	II
M_RGB24	RRGGBB
M_RGBA32	RRGGBBAA

pict_read_pict_c:

Los datos se almacenan por lineas de componentes, ajustándose los datos al siguiente esquema:

M_MONO, M_RGB8:	II II ... II
M_RGB24	RR RR ... RR
	GG GG ... GG
	BB BB ... BB
M_RGBA32	AA AA ... AA
	RR RR ... RR
	GG GG ... GG
	BB BB ... BB

Donde cada mayúscula representa un dígito hexadecimal, y existen pict_width() pixels por linea.

RESULTADO

Puntero a la estructura PICT generada.

En caso de error (por reserva de memoria o por error de acceso a fichero) el programa aborta.

VER

pict_free, pict_width

pict_24_to_8

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
Pict  pict_24_to_8(Pict pict_24,int mode, int bits);
```

DESCRIPCION

Genera un una estructura PICT en modo M_MONO o M_RGB8 a partir de una de modo M_RGB24 o M_RGBA32.

pict_24 debe ajustarse al formato generado por **pict_read_pict()** para imágenes de 24 o 32 bits.

mode puede tener uno de los siguientes valores:

- GRAYSCALE (escala de grises)
- OPTIMAL (paleta adaptativa)
- M_ALPHA (canal alpha de la imagen original)

bits debería tener el valor 5

NOTA: Esta función es una adaptación de un trabajo anterior de Juan Antonio Magallón [].

RESULTADO

Puntero a la estructura PICT generada.

En caso de error (por reserva de memoria) el programa aborta.

VER

pict_free, pict_read_pict

pict_get_map, pict_set_map

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int   pict_get_map(Pict pict,C_MAP map);
int   pict_set_map(Pict pict,C_MAP map);
```

DESCRIPCION

pict_get_map asigna a la variable *map* los valores correspondientes a la tabla de colores de *pict*

pict_set_map asigna a la tabla de colores de *pict* los valores almacenados en la variable *map*

RESULTADO

0 En caso de que *pict* no sea de modo indexado

1 En caso contrario

VER

pict_height, pict_width, pict_mode, pict_n_comp, pict_pixels

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int   pict_height(Pict pict);
int   pict_width(Pict pict);
int   pict_mode(Pict pict);
int   pict_n_comp(Pict pict);
U_char      *pict_pixels(Pict pict);
```

DESCRIPCION

Estas funciones permiten obtener las características de un Pict.

La estructura apuntada por *pict* debe haber sido obtenida por alguna de las siguientes funciones:

pict_read_pict(), pict_read_pict_c(), pict_open(), pict_24_to_8()

RESULTADO

pict_height: Altura de la imagen.

pict_width: Anchura de la imagen.

pict_mode: Modo de color de la imagen.

pict_n_comp: Número de componentes de cada pixel.

pict_pixels: Dirección de memoria a partir de la cual almacenan los datos **pict_read_pict**, **pict_read_pict_c** y **pict_24_to_8**.

VER

pict_read_pict(), pict_read_pict_c(), pict_open(), pict_24_to_8()

pict_pixel_rgba

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
void  pict_pixel_rgba(
      Pict pict,int i,int j,
      U_char *r,U_char *g,U_char *b,U_char *a  );
```

DESCRIPCION

Esta función permite obtener los valores de las componentes de un pixel determinado de un Pict.

(*i*,*j*) son las coordenadas del pixel (vertical y horizontal respectivamente, siendo la esquina superior izquierda el punto 0,0)

La estructura apuntada por *pict* debe haber sido obtenida por alguna de las siguientes funciones:

pict_read_pict(), pict_read_pict_c(), pict_24_to_8()

En caso de que *pict* sea de tipo M_RGBA32, si el valor de *a* es distinto de NULL, la función asignará al contenido de *a* el valor de la componente alpha.

A la salida de la función, los contenidos de *r, g* y b, tenrán los valores de las componentes roja, verde y azul respectivamente del pixel en cuestión.

RESULTADO

Ninguno

VER

pict_read_pict(), pict_read_pict_c(), pict_24_to_8()

pict_put_line, pict_get_line

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int   pict_put_line(Pict pict,U_char *pixels);
int   pict_get_line(Pict pict,U_char *pixels);
```

DESCRIPCION

pict_put_line permite escribir una imagen fila por fila, sin necesidad de cargar los datos de la imagen entera en memoria.

pict debe haber sido obtenido como resultado de una llamada a **pict_open** en modo P_WRITE.

pixels debe apuntar a una posición de memoria en la que esten almacenados los valores de las componentes de los pixels de la linea a escribir, según el formato descrito en **lpict_read_pict_c**

pict_get_line permite leer una imagen fila por fila, sin necesidad de cargar los datos de la imagen entera en memoria.

pict debe haber sido obtenido como resultado de una llamada a **pict_open** en modo P_READ.

pixels debe apuntar a una posición de memoria en la que se almacenarán los valores de las componentes de los pixels de la linea a escribir, según el formato descrito en **lpict_read_pict_c**.

El tamaño que deberá reservarse en *pixels* puede determinarse por medio de la expresión

```
(size_t)(pict_width() * pict_n_comp())
```

teniendo en cuenta que en el caso de que el modo sea M_MONO o M_RGB8 esta cantidad debe incrementarse en 1 byte.

RESULTADO

-1 en caso de que el modo no sea el adecuado, en otro caso:

pict_put_line devuelve el número de bytes añadidos al archivo (byteCount)

pict_get_line devuelve el número de pixels leidos del archivo (byteCount)

VER

pict_open(), pict_read_pict_c(), pict_width(), pict_n_comp().

pict_open, pict_close

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
Pict  pict_open(
      char *name,int file_mode,
      int img_H,int img_W,int img_mode,C_MAP p_map);
int   pict_close(Pict pict);
```

DESCRIPCION

pict_open permite abrir un fichero PICT2 con objeto de procesarlo manualmente.

name debe ser un nombre válido de fichero.

file_mode puede ser uno de los siguientes:

P_READ	modo de lectura
P_WRITE	modo de escritura

img_H es el número de lineas de la imagen

img_W es el número de pixels por linea

img_mode puede tener uno de los siguientes valores:

M_MONO	Escala de grises
M_RGB8	256 colores, indexado
M_RGB24	RGB, 256 niveles por componente
M_RGBA32	RGB + canal alfa.

El PICT así generado es almacenado en una estructura asignada dinámicamente por la propia función.

pict_close cierra un fichero PICT. En caso de que el modo sea P_WRITE, previamente se añade el código de operación ***EndOfPicture***

RESULTADO

pict_open devuelve un puntero a una estructura PICT inicializada correctamente, salvo error:

En caso de error de acceso al fichero, el programa aborta tras imprimir un mensaje.

En case de error en los datos o de falta de memoria, el programa imprime un mensaje y devuelve NULL

pict_close devuelve 0

VER

pict_free().

pict_free

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int     pict_free(Pict pict);
```

DESCRIPCION

Libera la memoria ocupada por *pict*, así como la reservada en el campo 'pixels'

RESULTADO

indefinido

VER

pack_bytes

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
short int pack_bytes( U_char *unpacked,
      short int usz, FILE *file );
```

DESCRIPCION

Esta función es de un nivel relativamente bajo, se utiliza para realizar la compresión RLE; y el usuario final generalmente no necesitará utilizarla.

unpacked contiene la dirección de memoria a partir de la que se encuentran los datos de partida.

usz es el número de bytes de *unpacked* que se desean comprimir.

file es el puntero al fichero en el que se escribirán los datos comprimidos. La escritura se realiza internamente en bloques de tamaño BUFSIZ.

Nota: El esquema RLE utilizado se diferencia del utilizado por Apple (siendo compatibles, no obstante) en que Apple, para evitar fenómenos de expansión no comprime los datos hasta que no se encuentran más de dos bytes iguales, mientras que en esta función se comprimen a partir del segundo dato repetido, pero se chequea posteriormente el tamaño, utilzándose un esquema alternativo en caso de que se haya producido expansión.

RESULTADO

Número de bytes escritos en el fichero.

VER

unpack_bytes().

unpack_bytes

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
short int unpack_bytes( U_char *unpacked,
      short int psz, FILE *file );
```

DESCRIPCION

Esta función es de un nivel relativamente bajo, se utiliza para realizar la descompresión RLE; y el usuario final generalmente no necesitará utilizarla.

unpacked contiene la dirección de memoria a partir de la que se almacenarán los datos obtenidos por la descompresión. Debe tener suficiente espacio para contener el número de bytes que se obtendrán. Por lo general esta cantidad es conocida de antemano; pero en el caso más general, puede ser hasta 65 veces el tamaño de los datos comprimidos.

psz es el número de bytes comprimidos que se desea procesar.

file es el puntero al fichero en del que se leerán los datos comprimidos. La lectura se realiza internamente en bloques de tamaño BUFSIZ.

RESULTADO

Número de bytes almacenados en *unpacked*.

VER

pack_bytes().

pict_ctable_98

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int   pict_ctable_98( Pict pict, C_MAP c_table);
```

DESCRIPCION

Esta función permite asignar valores a la tabla de colores de la variable *pict* y escribir los mismos en el fichero correspondiente.

pict debe estar abierto; y debe de ser de tipo color indexado (o escala de grises).

c_table es la tabla de color a asignar. En caso de que su valor sea NULL, se asignarán los valores correspondientes a la escala de grises.

RESULTADO

0.

VER

pict_write_header_98, pict_write_header_9A

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"

int pict_write_header_9A( Pict pict, US_int im_H,
US_int im_W, US_int modo );

int pict_write_header_98( Pict pict, US_int im_H,
US_int im_W, US_int modo, C_MAP c_table);
```

DESCRIPCION

Estas funciones son de un nivel relativamente bajo, se utiliza para preparar y escribir los datos asociados a los códigos 98 y 9A; y el usuario final generalmente no necesitará utilizarla.

pict es la variable a tratar.

im_H es el número pixels que tiene cada fila de la imagen.

im_W es el número de filas que tiene la imagen.

modo puede tener uno de los siguientes valores:

M_MONO	Escala de grises
M_RGB8	256 colores, indexado
M_RGB24	RGB, 256 niveles por componente
M_RGBA32	RGB + canal alfa.

c_table:Para **pict_write_header_98**, debe contener la tabla de colores que se desea asignar, salvo en el caso de que el modo sea M_MONO, en cuyo caso puede tener el valor NULL

RESULTADO

Valor del campo ***rowBytes***. En caso de fallo por falta de memoria, imprime un mensaje de error y aborta la ejecución.

En caso de darse algún otro tipo de error, devuelve -1

VER

pict_read_header_9A(), pict_read_header_98().

pict_read_header_98, pict_read_header_9A

LIBRERIAS

libpict.

SINTAXIS

```
#include "pict.h"
int   pict_read_header_9A(Pict pict);
int   pict_read_header_98(Pict pict, C_MAP c_table);
```

DESCRIPCION

Estas funciones son de un nivel relativamente bajo, se utiliza para leer los datos asociados a los códigos 98 y 9A; y el usuario final generalmente no necesitará utilizarla.

pict es la variable a tratar.

c_table:Para **pict_read_header_98**, debe contener la tabla de colores que se desea asignar.

RESULTADO

Valor del campo ***rowBytes***. En caso de fallo por falta de memoria, imprime un mensaje de error y aborta la ejecución.

En caso de darse algún otro tipo de error, devuelve -1

VER

pict_write_header_9A(), pict_write_header_98().