Computer Labs: Lab 5 - Part 2
XPMs ~~& VBE Function 0x00~~

2º LEIC

Pedro F. Souto (`pfs@fe.up.pt`)

April 5, 2024

Lab5: Video Card in Graphics Mode - 2nd Lab Class

- Write a set of functions:
  ```
  int video_test_xpm(const char *xpm, uint16_t xi, uint16_t yi)
  int video_test_move(const char *xpm, uint_t xi,
                             uint_t yi, ...)
  ```
  ~~`int video_test_controller()`~~
- ~~Develop your own implementation of `vbe_get_mode_info()`, which must call VBE function 0x01, Return VBE Mode Information.~~

Lab 5: `video_test_xpm()`

```
int video_test_xpm(const char *xpm, uint16_t xi,  uint16_t yi)
```

What Display at the screen coordinates `(xi,yi)` the pixmap in XPM format passed in `xpm` array

- Use VBE mode `0x105`

Pixmaps and XPM

pixmap is a short term for “pixel map”, the representation of a digital image as an array of pixel color values

- I.e. it is a map of screen coordinates to color values
- **bitmap** is a more generic term used in other fields of computer science to denote the mapping from one domain to a bit string

XPM X Pixmap is an image format where each color value of a pixmap is represented by **character sequences/strings**

- An XPM for a given pixmap can be stored:
  - either in a text file,
  - or as an array in a C source file
- In Lab 5, we use a simplified version of the XPM format:
  - It uses only one character per color value

Terminology we will use:

(pix)map to refer to a pixmap that uses the color encodings of a given graphical mode

xpm to refer to the XPM representation of a pixmap

- In Lab 5, we use an array of strings to store an XPM

Example: Using C Arrays to Store XPMs ("legacy")

```
static char *pic1[] = {
"32 13 4", /* number of columns and rows, in pixels, and colors */
". 0", /* ’.’ denotes color value 0 */
"x 2", /* ’x’ denotes color value 2 */
"o 14", /* .. and so on */
"+ 4",
"................................", /* the map  */
"..............xxx...............",
"............xxxxxxx.............",
".........xxxxxx+xxxxxx..........",
"......xxxxxxx+++++xxxxxxx.......",
"....xxxxxxx+++++++++xxxxxxx.....",
"....xxxxxxx+++++++++xxxxxxx.....",
"......xxxxxxx+++++xxxxxxx.......",
".........xxxxxx+xxxxxx..........",
"..........ooxxxxxxxoo...........",
".......ooo...........ooo........",
".....ooo...............ooo......",
"...ooo...................ooo...."
};
```

Question How many elements does an XPM array have?

Example: Using C Arrays to Store XPMs

```
static xpm_row_t const minix3_xpm[] = {
   "196 196 950 2",
   "      c None",
   ".     c #C1C1C1",
   "+     c #323232",
   "@     c #090909",
   "#     c #010101",
   "$     c #161616",
   "%     c #9C9C9C",
   [...]
   "                                           + @ # $ %
   "                                          & * = - ;
    , ' )       ! ~ { ] ^ / ( _ : <
   "                                         [ } | |
   | | | 1 2 3 4 5     6 7 8 9
    0 a / | | b c d e f ] g h @ g i i i j k
   [...]
}
```

Lab 5: `video_test_xpm()`

```
int video_test_xpm(char *xpm, uint16_t xi, uint16_t yi)
```

What Display at the screen coordinates `(xi,yi)` the pixmap in XPM format passed in `xpm` array

- Use VBE mode `0x105`

Issue How to convert the xmp to a pixmap?

Answer Use the `xpm_load()` function

Note In the following pages, about the `xpm_load()` function, most functions and types start with the substring **`xpm_`** because they are defined in module XPM. Thus we use:

`map` to refer to pixmap in XPM format

`image` to refer to the image pixmap, i.e. with the colors encoded for a specific graphics mode

Reading a Pixmap from its XPM: `xpm_load()` (1/2)

```
#define TRANSPARENCY_COLOR_1_5_5_5 0x8000
#define TRANSPARENCY_COLOR_8_8_8_8 0xFF000000
#define CHROMA_KEY_GREEN_888 0x00b140
#define CHROMA_KEY_GREEN_565 0x0588
enum xpm_image_type {
  XPM_INDEXED, // for Minix def. pallette in VBE mode  0x105
  XPM_1_5_5_5,
  XPM_5_6_5,
  XPM_8_8_8,
  XPM_8_8_8_8,
  INVALID_XPM
};
typedef struct {
  enum xpm_image_type type;
  uint16_t width;
  uint16_t height;
  size_t size;       // size of the pixmap in bytes
  uint8_t *bytes;    // pointer to memory with pixmap
} xpm_image_t;
uint8_t *(xpm_load)(xpm_map_t map, enum xpm_image_type type,
                    xpm_image_t *img);
```

Reading a Pixmap from its XPM: `xpm_load()` (2/2)

```
xpm_map_t xmap;      // XPM
xpm_image_t img;     // pixmap and metadata
uint8_t *map;        // pixmap itself
// get the pixmap from the XPM
map =  xpm_load(xmap, XPM_INDEXED, &img);
// copy it to graphics memory
```

```
uint8_t *xpm_load(xpm_map_t xmap,
            enum xpm_image_type type,
            xpm_image_t *img);
```

reads an XPM pixmap `xmap`, and returns the pixmap as a two-dimensional `uint8_t` array. The color encoding of the (output) pixmap is the one specified in `type`. It assumes that the XPM uses:

- Either one char per color and one byte per color – only for Minix default pallette in VBE-mode `0x105`
- Or 3 bytes per color, with no restriction on the number of chars per color (this is the format generated by GIMP)

Lab 5: `video_test_move()` (1/4)

```
int video_test_move( xpm_map_t *xpm[],,
                     uint16_t xi, uint16_t yi,
                     uint16_t xf, uint16_t yf
                     int8_t speed, uint8_t frame_rate)
```

What? Move an image on the screen (only along the x or y axes)

`xpm` XPM for the sprite

`(xi,yi)` initial coordinates (of the upper left corner (ULC))

`(xf,yf)` final coordinates (of ULC)

`speed` speed

If non-negative number of pixels between consecutive frames

If negative number of frames required for a 1 pixel movement

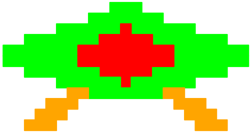
`frame_rate` number of frames per second. This is crucial for a smooth movement of images on the screen

- Again, this has to do with the human vision
- But it also depends on how fast this movement is
- Use the Timer 0 interrupts for setting the frame-rate

Lab 5: `video_test_move()` (2/4)

Question? How can we give the illusion of movement of an image on the screen?

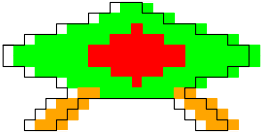
Answer Just draw it repeatedly, in successive positions, starting from its initial position until its final position.



Lab 5: `video_test_move()` (2/4)

Question? How can we give the illusion of movement of an image on the screen?

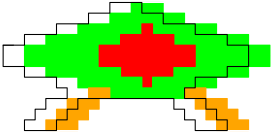
Answer Just draw it repeatedly, in successive positions, starting from its initial position until its final position.



Lab 5: `video_test_move()` (2/4)

Question? How can we give the illusion of movement of an image on the screen?

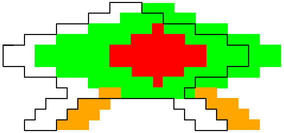
Answer Just draw it repeatedly, in successive positions, starting from its initial position until its final position.



Lab 5: `video_test_move()` (2/4)

Question? How can we give the illusion of movement of an image on the screen?

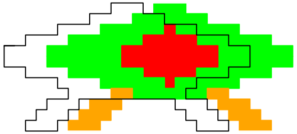
Answer Just draw it repeatedly, in successive positions, starting from its initial position until its final position.



Lab 5: `video_test_move()` (2/4)

Question? How can we give the illusion of movement of an image on the screen?

Answer Just draw it repeatedly, in successive positions, starting from its initial position until its final position.



Lab 5: `video_test_move()` (3/4)

Example Consider an horizontal movement from column 200 to column 800, i.e. of 600 pixels

Question If we move the image by 1 pixel on every Timer 0 interrupt, how long does it take?

Lab 5: `video_test_move()` (3/4)

Example Consider an horizontal movement from column 200 to column 800, i.e. of 600 pixels

Question If we move the image by 1 pixel on every Timer 0 interrupt, how long does it take?

Answer 600/60 = 10 s

- We need 600 Timer 0 interrupts to move the image by the 600 pixels
- By default, the Timer 0 generates 60 interrupts per second

Question How can you reduce this time?

Lab 5: `video_test_move()` (3/4)

Example Consider an horizontal movement from column 200 to column 800, i.e. of 600 pixels

Question If we move the image by 1 pixel on every Timer 0 interrupt, how long does it take?

Answer 600/60 = 10 s

- We need 600 Timer 0 interrupts to move the image by the 600 pixels
- By default, the Timer 0 generates 60 interrupts per second

Question How can you reduce this time?

Answer Two alternatives:

Using a higher interrupt rate need to configure Timer 0 (**bad**)

Moving the image by more than 1 pixel between frames

`speed` `video_text_move()`'s argument is used for that

- Using negative values for specifying the number of frames required for a 1 pixel movement, we can make ultra-slow movements

Lab 5: `video_test_move()` (4/4)

Issue If you just redraw the image in its successive positions, every movement will lead to a trail

Fix Ensure that the screen's pixels occupied in a position but not occupied in the next position are reset to their previous value

How can we do this?

Lab 5: `video_test_move()` (4/4)

Issue If you just redraw the image in its successive positions, every movement will lead to a trail

Fix Ensure that the screen's pixels occupied in a position but not occupied in the next position are reset to their previous value

How can we do this?

- **Alternative 1** Every time we move an image:
    1. Delete the image, i.e. reset the color of the screen pixels in its current position
    2. Redraw the image in its next position

    Optimization only reset those screen pixels that are not overlapped in the new position
- **Alternative 2** Redraw the entire screen at the desired frame-rate
    1. On a **second buffer**, create a **frame**, a complete screen, with the image in its position in the next frame
    2. Copy the contents of the second buffer to the **frame-buffer**

    Optimization You can omit these steps, if the position of the image on the next frame is the same as in the current frame