LED-Matrix cheat sheet

(for use with Frank's framework)

Overview

- Prototype for an animation
- Time schedule
- Reference: Functions and constants

Prototype for an animation

```
void PrototypeAnimation(int len_s)  // len_s = length in seconds
{
  // your variables
  int x=0;                              // x-position of the dot

  // standard loop part (copy this part, no need to understand)
  int wait=100;                      // defines 100 ms steps
  int counter=1000/wait*len_s;       // len_s = length in seconds
  while (counter--)                  // loop
  { frame_delay(wait);               // waits to fill the 100 ms
    swap_buffers(1);                 // show last frame, start the next frame

    // simulation step
    paint(x,0,  255,0,0);      // red (255,0,0) dot at x,0 (0=y is top of screen)
    x = x+1;                   // move right
    if (x >= SCRdx) x=0;       // if at border (SCRdx) restart at 0
  }       // SCRdx / SCRdy is screen size given by framework
}
```

Frame and Time schedule

The programming environment is the IDE of Arduino. Arduino programs are use a „void loop() { … }“ function that will be called indefinitely. Into this function you can call your animations, resulting in a time schedule. Some lines of code in the beginning are needed anway, you don't need to change them, but they must remain in the code.

```
#include "LightMatrixKernelLib.h"
int time_turbo=1; // std=1 more(n)= n times faster
int slowmotion=1; // std=1 more(n)= n times slower
int DO_show_as_ascii=0; // std=0  1=show as ASCII output pseudo graphics
void user_init()  // arduino init() is already called in lib
{ // if you want to init something, do it here
}
// → insert here your animation functions ←
void loop()                       // repeated indefinitely:
{ string_anim("HELLO",400,1, 64,255,64); // 1. Write HELLO in light green
  PrototypeAnimation(6);          // 2. PrototypeAnimation should run 6 seconds
  Fireworks(8);                   // 3. fireworks animation should run 8 seconds
}
```

Reference: Functions and constants

paint(x,y, r,g,b);

  set a pixel at x,y (int) with the color r,g,b (int)

  0 <= r, g, b <= 255   0=dark 255=max.bright r=red g=green b=blue

  0 <= x < SCRdx (horizontal screen size given by framework)

  0 <= y < SCRdy (vertical screen size)

  It's ok to set pixel outside this area (without effect).

paint_hi(x,y, r,g,b);

  similar to paint(...), but

  1. (x,y) are fixed point numbers (int value FIXP is 1.0)
     This one is bit tricky: to set Pixel at Position (1,2)
     you must set (int) numbers 1*FIXP and 2*FIXP. The nice thing is
     you can set the pixel between the coordinates (1,2) and (1,3)
     as an example. You would write a yellow pixel e.g.:

```
paint_hi(1*FIXP,2*FIXP+FIXP/2, 255,255,0);
// Fireworks(...) example animation does it like this
```

     or – more intuitively - by use of float numbers:

```
paint_hi(1*FIXP,2.5*FIXP, 255,255,0); // easy, isn't it?
```

  2. r,g,b can grow above 255! Result: Light floods a bit in nearby
     pixels

string_anim(char *string_to_write, int time_to_show_the_string,
            int set_1_if_to_animate_print_of_chars_slowly__else_0,
            unsigned char r, unsigned char g, unsigned char b);

  Print the string an then wait some time, so the spectator can see
  it for a while, e.g. 400 ms. The color will be r,g,b (0..255).
  If the text is too long, it wil scroll automatically.
  Character allowed: A-Z, space

analogRead – ask for potentiometer value

```
int mypos = analogRead(A1);
```

  If you want to make something interactive,
  you can ask for the position value of a rotary knob.
  Resulting values are in the range 0..1023.

Reference: Functions and constants continued
(more special, usually not needed)

paint_add(int x, int y, int r, int g, int b);
  Basically the same as paint, but it adds the r,g,b values instead
  of overwriting them into the pixel cell.
  (paint_hi(...) does this also.)

frame_delay(int ms);
  Waits until ms milliseconds elapsed relative to its last call.

swap_buffers(int mode);
  Write the current image into the LED-Matrix and if mode==1, then
  clear the current image for a clean start of the next frame.

char_paint(int x, int y, char c,
          unsigned char r, unsigned char g, unsigned char b);
  Paints a char (A-Z) at position x,y.

void get_pixel(int x, int y)
  Read the current pixel, returns r,g,b as: return_rgb.r, …

int time_turbo = acceleration; // global variable, default 1
  Set this acceleration to speed up the animations (might not work)
  int slowmotion=slowdown; // the same for slower animation

int DO_show_as_ascii = on_off; // global variable, 1=on 0=off
  Writes the animation to serial output in ASCII graphics if on;
  useful to check for hardware defects or to write an animation,
  if you have no physical LED matrix to test with.