SDL2X TOOLKIT 0.2 (VERY OPTIMIZED!!)

By Waninkoko

This toolkit has been created for make the work of the developers more easy.

With this toolkit you can draw polygons, surfaces, draw texts, load sounds... with only executing a function.

I hope you like it ;)

: Index :

Page 2 - Index

Page 3 - gp2x.h
Page 4 - hw_gp2x.h
Page 6 - sdl_gp2x.h
Page 9 - sdl_gfx_gp2x.h
Page 13 - sdl_image_gp2x.h
Page 14 - sdl_mixer_gp2x.h
Page 18 - sdl_primitives_gp2x.h
Page 19 - sdl_ttf_gp2x.h
Page 22 - sdl_joystick_gp2x.h

Page 23 - Conclusion

:1: gp2x.h

This file includes some basic functions for the GP2X.

void LoadExec(const char *c);

Loads an executable.

c: filename

void ReturnGP2X();

Loads GP2X main menu.

char *Int2Str(int i);

Converts an integer to a char.

i: integer

:2: hw_gp2x.h

This file includes functions about GP2X hardware (overclocking, GPIO...).

int Get920Clock();

Get ARM 920t clock frequency. This function returns clock frequency in MHZ.

void Set920Clock(int Mhz);

Set ARM 920t clock frequency.

Mhz: clock frequency in MHZ

void Disable940();

Disable ARM 940t.

void Enable940();

Enable ARM 940t.

void BatteryLED();

On/Off battery LED.

void ShutdownScreen();

On/Off screen.

int BatteryLEDState();

Get battery LED state. This function returns 1 if battery LED is on and returns 0 if battery LED is off.

int ScreenState();

Get screen state. This function returns 1 if screen is on and returns 0 if screen is off.

```
int GetBatteryCharge();
```

Get battery charge. This function returns 0 if battery is empty, 1 if battery is at medium charge, 2 if battery is full and 3 if there's using AC adapter.

:3: sdl_gp2x.h

This file includes basic functions about SDL, like initialization, draw a surface...

SDL_Surface *InitSDL(int w, int h, int d);

Initialize SDL. This funcion returns the “screen” surface that has set video mode.

w: screen width
h: screen height
d: screen depth

void QuitSDL();

Quit SDL. This function quits SDL and SDL subsystems (SDL_mixer, SDL_ttf).

void JoystickOpen(int device);

Open joystick device. In GP2X the joystick device is 0.

void DrawPixel(SDL_Surface *dst, int x, int y, Uint32 c);

Draw a pixel.

dst: target surface
x: x coordinate
y: y coordinate
c: pixel color (format 0xRRGGBB)

Uint32 GetPixelColor(SDL_Surface *src, int x, int y);

Get pixel color.

src: source surface
x: x coordinate
y: y coordinate

void SetGamma(float r, float g, float b);

Set screen gamma.

r: red

```
    g:   green
    b:   blue
```

void ShowMouse(int i);

```
    Show SDL mouse.

    i:   1 = enable / 0 = disable
```

void MoveMouse(Uint16 x, Uint16 y);

```
    Move SDL mouse.

    x:   x coordinate
    y:   y coordinate
```

void GetMousePosition(int *x, int *y);

```
    Get position of SDL mouse.

    x:   variable where save x coordinate
    y:   variable where save y coordinate
```

void DrawSurface(SDL_Surface *dst, SDL_Surface *src, int x, int y);

```
    Draw a surface.

    dst: target surface
    src: source surface
    x:   x coordinate
    y:   y coordinate
```

void DrawSurfaceCenter(SDL_Surface *dst, SDL_Surface *src, int x, int y);

```
    Like function above but variables “x” and “y” are the
coordinates of the surface's center.
```

void DrawSurfacePortion(SDL_Surface *dst, SDL_Surface *src, int x, int y, int x2, int y2, int w, int h);

```
    Draw portion of a surface.

    dst: target surface
    src: source surface
```

```
    x:   x coordinate
    y:   y coordinate
    x2:  x coordinate of the source surface
    y2:  y coordinate of the source surface
    w:   width of the portion of the source surface
    h:   height of the portion of the source surface
```

```
void SaveScreenshot(SDL_Surface *screen);
```

```
    Save screenshot.

    screen:   screen surface
```

:4: sdl_gfx_gp2x.h

This file includes functions about drawing polygons, drawing transparent polygons, modify surfaces...

This functions are too slow at the moment so, if you can, use functions above.

void DrawLineRGBA(SDL_Surface *dst, int x, int y, int x2, int y2, Uint32 c);

Draw a line.

dst: target surface
x: x coordinate
y: y coordinate
x2: x2 coordinate
y2: y2 coordinate
c: color (format 0xRRGGBBAA)

void DrawRectangleRGBA(SDL_Surface *dst, int x, int y, int w, int h, Uint32 c);

Draw a rectangle (not filled).

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBBAA)

void DrawBoxRGBA(SDL_Surface *dst, int x, int y, int w, int h, Uint32 c);

Draw a filled rectangle.

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBBAA)

void DrawBoxWithBorderRGBA(SDL_Surface *dst, int x, int y, int w, int h, Uint32 c, int a, int s, Uint32 c1, int a1);

Draw a filled rectangle with border.

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBBAA)
s: border size
c1: border color (format 0xRRGGBBAA)

void DrawCircleRGBA(SDL_Surface *dst, int x, int y, int radius, Uint32 c);

Draw a circle (not filled).

dst: target surface
x: x coordinate
y: y coordinate
radius: circle radius
c: color (format 0xRRGGBBAA)

void DrawEllipseRGBA(SDL_Surface *dst, int x, int y, int rx, int ry, Uint32 c);

Draw an ellipe (not filled).

dst: target surface
x: x coordinate
y: y coordinate
rx: x radius
ry: y radius
c: color (format 0xRRGGBBAA)

void DrawFilledCircleRGBA(SDL_Surface *dst, int x, int y, int radius, Uint32 c);

Draw a filled circle.

dst: target surface
x: x coordinate
y: y coordinate
radius: circle radius
c: color (format 0xRRGGBBAA)

void DrawFilledEllipseRGBA(SDL_Surface *dst, int x, int y, int rx, int ry, Uint32 c);

Draw a filled ellipse.

```
    dst: target surface
    x:   x coordinate
    y:   y coordinate
    rx:  x radius
    ry:  y radius
    c:   color (format 0xRRGGBBAA)
```

```
void DrawTriangleRGBA(SDL_Surface *dst, int x1, int y1, int x2,
int y2, int x3, int y3, Uint32 c);
```

```
    Draw a triangle (not filled).

    dst: target surface
    x1:  x1 coordinate
    y1:  y1 coordinate
    x2:  x2 coordinate
    y2:  y2 coordinate
    x3:  x3 coordinate
    y3:  y3 coordinate
    c:   color (format 0xRRGGBBAA)
```

```
void DrawFilledTriangleRGBA(SDL_Surface *dst, int x1, int y1, int
x2, int y2, int x3, int y3, Uint32 c);
```

```
    Draw a filled triangle.

    dst: target surface
    x1:  x1 coordinate
    y1:  y1 coordinate
    x2:  x2 coordinate
    y2:  y2 coordinate
    x3:  x3 coordinate
    y3:  y3 coordinate
    c:   color (format 0xRRGGBBAA)
```

```
void DrawRegularTriangleRGBA(SDL_Surface *dst, int x, int y, int
d, Uint32 c);
```

```
    Draw a regular triangle (not filled).

    dst: target surface
    x:   x coordinate
    y:   y coordinate
    d:   side longitude
    c:   color (format 0xRRGGBBAA)
```

```
void DrawRegularFilledTriangleRGBA(SDL_Surface *dst, int x, int y,
```

```
int d, Uint32 c);
```

Draw a regular filled triangle.

```
dst: target surface
x:   x coordinate
y:   y coordinate
d:   side longitude
c:   color (format 0xRRGGBBAA)
```

```
SDL_Surface *RotateSurface(SDL_Surface *src, double angle, int
smooth);
```

Rotate a surface. This function returns the rotated surface.

```
src:      source surface
angle:    rotation angle
smooth:   antialiasing (1 = on, 0 = off)
```

```
SDL_Surface *ZoomSurface(SDL_Surface *src, double zoomx, double
zoomy, int smooth);
```

Zoom a surface. This function returns the zoomed surface.

```
src:      source surface
zoomx:    x zoom
zoomy:    y zoom
smooth:   antialiasing (1 = on, 0 = off)
```

```
SDL_Surface *RotoZoomSurface(SDL_Surface *src, double angle,
double zoomx, double zoomy, int smooth);
```

Rotate and zoom a surface. This function returns the rotated and zoomed surface.

```
src:      source surface
angle:    rotation angle
zoomx:    x zoom
zoomy:    y zoom
smooth:   antialiasing (1 = on, 0 = off)
```

:5: sdl_image_gp2x.h

This file includes functions about images.

SDL_Surface *LoadImage(const char *file, Uint32 c);

Load an image and returns a surface with the image loaded.

file: image filename
c: surface color key (-1 = no color key)

char *GetImageFormat(const char *file);

Get image format. This function returns the image format in char.

file: image filename

:6: sdl_mixer_gp2x.h

This file includes functions about mixer like mixer initialization, play a file, play a channel, sound effects...

void InitSDL_mixer(int freq, int channels, int buffer);

Initialize SDL_mixer with the parsed frequency, channels and buffer.

freq: frequency
channels: number of channels (1 – mono, 2 – stereo)
buffer: sound buffer

void QuitSDL_mixer();

Quit SDL_mixer.

void AllocateChannels(int channels);

Allocate mixing channels.

channels: number of mixing channels

Mix_Chunk *LoadChannel(const char *c);

Load a channel. This function returns the channel loaded.

c: sound file

Mix_Music *LoadMusic(const char *c);

Loads music. This function returns the music loaded.

c: music file

void CloseChannel(Mix_Chunk *chunk);

Close a channel.

chunk: loaded sound

void CloseMusic(Mix_Music *music);

Close music.

music: loaded music

```
char *GetMusicFormat(Mix_Music *music);
```

Get music format.

music: loaded music

```
void SetChannelVolume(int channel, int volume);
```

Set channel volume.

channel: channel
volume: channel volume

```
void SetMusicVolume(int volume);
```

Set music volume.

volume: music volume

```
void PauseChannel(int channel);
```

Pause channel.

channel: channel

```
void PauseMusic();
```

Pause music.

```
void StopChannel(int channel);
```

Stop channel.

channel: channel

```
void StopMusic();
```

Stop music.

```
void SetMusicPosition(double pos);
```

Set music position.

pos: music position

void PlayChannel(Mix_Chunk *chunk, int channel, int loops);

Play a channel.

chunk: loaded sound
channel: channel to play sound
loops: number of loops (-1 = infinite)

void PlayChannelTimed(Mix_Chunk *chunk, int channel, int loops, int ticks);

Play channel during indicated time.

chunk: loaded sound
channel: channel to play sound
loops: number of loops (-1 = infinite)
ticks: playing duration

void FadeInChannel(Mix_Chunk *chunk, int channel, int loops, int ms);

Play a channel with fade-in effect.

music: loaded sound
channel: channel to play sound
loops: number of loops (-1 = infinite)
ms: effect duration in milliseconds

void FadeInChannelTimed(Mix_Chunk *chunk, int channel, int loops, int ms, int ticks);

Play a channel during indicated time with fade-in effect.

music: loaded sound
channel: channel to play sound
loops: number of loops (-1 = infinite)
ms: effect duration in milliseconds
ticks: playing duration

void FadeOutChannel(int channel, int ms);

Stop channel with fade-out effect.

channel: channel
ms: effect duration in milliseconds

```
void PlayMusic(Mix_Music *music, int loops);
```

Play music.

music: loaded music
loops: number of loops (-1 = infinite)

```
void FadeInMusic(Mix_Music *music, int loops, int ms);
```

Play music with a fade-in effect.

music: loaded music
loops: number of loops (-1 = infinite)
ms: effect duration in milliseconds

```
void FadeInMusicPos(Mix_Music *music, int loops, int ms, double
pos);
```

Play music from a position with a fade-in effect.

music: loaded music
loops: number of loops (-1 = infinite)
ms: effect duration in milliseconds
pos: music position

```
void FadeOutMusic(int ms);
```

Stop music with a fade-out effect.

ms: effect duration in milliseconds

:7: sdl_primitives_gp2x.h

This file includes some function about drawing primitives (like SDL_gfx but faster).

void DrawRectangle(SDL_Surface *dst, int x, int y, int w, int h, int s, Uint32 c);

Draw a rectangle (not filled).

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBB)

void DrawBox(SDL_Surface *dst, int x, int y, int w, int h, Uint32 c);

Draw a filled rectangle.

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBB)

void DrawBoxWithBorder(SDL_Surface *dst, int x, int y, int w, int h, Uint32 c, int s, Uint32 c1);

Draw a filled rectangle with border.

dst: target surface
x: x coordinate
y: y coordinate
w: width
h: height
c: color (format 0xRRGGBB)
s: border size
c1: border color (format 0xRRGGBB)

:8: sdl_ttf_gp2x.h

This file includes functions about initialize truetype api, drawing texts...

void InitSDL_TTF();

Initialize SDL_ttf.

void QuitSDL_TTF();

Quit SDL_ttf;

TTF_Font *LoadFont(const char *c, int s);

Load a font. This function returns the loaded font.

c: font file
s: font size

void CloseFont(TTF_Font *f);

Close font.

f: font

SDL_Surface *RenderTextSolid(TTF_Font *f, SDL_PixelFormat *fmt, const char *text, Uint32 c);

Render text solid. This function returns a surface with rendered text.

f: font
fmt: surface format
text: text
c: color (format 0xRRGGBB)

SDL_Surface *RenderTextShaded(TTF_Font *f, SDL_PixelFormat *fmt, const char *text, Uint32 c, Uint32 bg);

Render text shaded. This function returns a surface with rendered text.

f: font
fmt: surface format

```
text:     text
c:   color (format 0xRRGGBB)
bg:  background color (format 0xRRGGBB)
```

SDL_Surface *RenderTextBlended(TTF_Font *f, SDL_PixelFormat *fmt, const char *text, Uint32 c);

Render text blended. This function returns a surface with rendered text.

```
f:   font
fmt: surface format
text:     text
c:   color (format 0xRRGGBB)
```

void DrawTextSolid(SDL_Surface *dst, TTF_Font *f, const char *text, int x, int y, Uint32 c);

Draw a text solid with selected font.

```
dst: target surface
f:   font
text:     text
x:   x coordinate
y:   y coordinate
c:   color (format 0xRRGGBB)
```

void DrawTextShaded(SDL_Surface *dst, TTF_Font *f, const char *text, int x, int y, Uint32 c, Uint32 bg);

Draw a text solid with selected font.

```
dst: target surface
f:   font
text:     text
x:   x coordinate
y:   y coordinate
c:   color (format 0xRRGGBB)
bg:  background color (format 0xRRGGBB)
```

void DrawTextBlended(SDL_Surface *dst, TTF_Font *f, const char *text, int x, int y, Uint32 c);

Draw a text blended with selected font.

```
dst: target surface
f:   font
text:     text
```

```
    x:   x coordinate
    y:   y coordinate
    c:   color (format 0xRRGGBB)
```

```
void DrawTextWithShadow(SDL_Surface *dst, TTF_Font *f, const char
*text, int x, int y, Uint32 c, Uint32 c1);
```

```
    Draw a text with selected font and with a shadow.

    dst: target surface
    f:   font
    text:     text
    x:   x coordinate
    y:   y coordinate
    c:   color (format 0xRRGGBB)
    c1:  shadow color (format 0xRRGGBB)
```

:9: sdl_joystick_gp2x.h

This file includes functions about joystick handling.

void ReadJoystick(int davec);

Read joystick with one of the DaveC's configurations.

davec: DaveC configuration

1 – configuration 1
2 – configuration 2
3 – configuration 3
4 – configuration 4
5 – configuration 5
6 – configuration 6

When you want to know if a key is pressed or the joystick's position (up, down, left, right) you have to call this function and check if the correspondent button variable is 1 or 0.

joy1 – joystick upleft
joy2 – joystick up
joy3 – joystick upright
joy4 – joystick right
joy5 – joystick downright
joy6 – joystick down
joy7 – joystick downleft
joy8 – joystick left

joya – A button
joyb – B button
joyx – X button
joyy – Y button
joyl – L button
joyr – R button
joyselect – SELECT button
joystart – START button
joyvoldown – VOLDOWN button
joyvolup – VOLUP button
joyclick – CLICK button

IMPORTANT: You have to call function PrepareJoystick() before handling joystick. This function initializes needed variables.

:10: Conclusion

If you find a bug, you have a suggestion or anything else you can contact me at waninkoko@gmail.com

I know that this document has some gramatical errors (I think, lol) so you can send me corrections to my email address.