

Drawing Shapes with SDL

Note: This tutorial assumes that you already know how to display a window and draw a sprite with SDL.

SDL only comes with the ability to draw sprites. If you want to draw primitive shapes like points and lines, you can use the SDL_gfx library. SDL_gfx is a very easy to use library that allows you to do a number cool things. This tutorial focuses on using SDL_gfx to draw shapes.

You can find the SDL_gfx homepage [here](#).

Since I use Visual Studio .NET 2003, I'll show you how to set up SDL_gfx with that IDE. If you use another IDE, you'll have to go to the SDL_gfx homepage and figure out how to get it to work.

Setting Up SDL_gfx in Visual Studio .NET 2003

Since SDL is cross-platform, you often have to compile SDL libraries for your platform yourself. SDL_gfx is one of those libraries. I don't see how it would benefit you to have to do this though, so I've gone and done it for you. [Click here](#) to download the zip containing SDL_gfx. Extract it to a location of your choosing.

Note that everything you're downloading is from the guys who wrote SDL_gfx. I take no credit for anything. All I did was build a project that they provided.

In Visual Studio, select **Tools->Options**. Go to **Projects** and choose **VC++ Directories**. In the **Show directories for:** drop down menu, select **Include files**. Click the **New Line** button (looks like a folder with a star behind it) and then click the ... button that appears. Navigate to where you extracted SDL_gfx to and highlight the **include** folder. Click **Open**.

Select **Tools->Options**. Go to **Projects** and choose **VC++ Directories**. In the **Show directories for:** drop down menu, select **Library files**. Click the **New Line** button (looks like a folder with a star behind it) and then click the ... button that appears. Navigate to where you extracted SDL_gfx to and highlight the **lib** folder. Click **Open**.

Go to the **lib** folder in your SDL_gfx directory and copy-paste the "sdlgfx.dll" file to your project's main directory. Be sure to include this file in the same directory as your .exe file when you distribute your programs.

In Visual Studio, select **Project->Project Name Properties**. Select **Linker** and then **Input**. In **Additional Dependencies**, you need to have "sdl.lib sdlmain.lib sdlgfx.lib".

Drawing with SDL_gfx

Before drawing with SDL_gfx you have to include "SDL_gfxPrimitives.h". That's all the initialization required. Note that primitives are things like points, lines, rectangles, etc.

SDL_gfx provides a number of functions for drawing shapes. I'll only be covering the following because I think they're all you should need:

- points
- lines
- triangles
- filled triangles
- rectangles
- filled rectangles
- ellipses
- filled ellipses
- polygons
- filled polygons

I'll now provide a list of the functions I think are most useful. If you want a complete list, see the SDL_gfx homepage. Note that the functions take Sint16's for coordinates. These are just short's, which are 16 bit signed (can be negative and positive) integers.

The alpha value (always the last parameter) is for transparency. Pass in 255 if you want no transparency. Giving values less than that will make the shapes partially transparent, so they'll blend with anything that's underneath them. Neat, eh?

Each function returns 0 for success and -1 for error.

```
// Draw a pixel at (x, y) with RGBA color (r, g, b, a)
int pixelRGBA(SDL_Surface* dst,
              Sint16 x, Sint16 y,
              Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a line from (x1, y1) to (x2, y2) with RGBA color (r, g, b, a)
int lineRGBA(SDL_Surface* dst,
             Sint16 x1, Sint16 y1,
             Sint16 x2, Sint16 y2,
             Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a triangle with vertices (x1, y1), (x2, y2), (x3, y3) and
```

```

// RGBA color (r,g,b,a)
int trigonRGBA(SDL_Surface* dst,
               Sint16 x1, Sint16 y1,
               Sint16 x2, Sint16 y2,
               Sint16 x3, Sint16 y3,
               Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a filled triangle with vertices (x1, y1), (x2, y2), (x3, y3)
// and RGBA color (r, g, b, a)
int filledTrigonRGBA(SDL_Surface* dst,
                    Sint16 x1, Sint16 y1,
                    Sint16 x2, Sint16 y2,
                    Sint16 x3, Sint16 y3,
                    Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a rectangle with top left corner at (x1, y1) and
// bottom right corner at (x2, y2), and RGBA color (r,g,b,a)
int rectangleRGBA(SDL_Surface* dst,
                 Sint16 x1, Sint16 y1,
                 Sint16 x2, Sint16 y2,
                 Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a filled rectangle with top left corner at (x1, y1) and
// bottom right corner at (x2, y2), and RGBA color (r,g,b,a)
int boxRGBA(SDL_Surface* dst,
            Sint16 x1, Sint16 y1,
            Sint16 x2, Sint16 y2,
            Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw an ellipse with center at (x, y), and x axis radius rx,
// and y axis radius ry, and RGBA color (r,g,b,a). rx can also
// be thought of as the distance from the center to the left/right
// most point. ry is the same, only from the top/bottom most point.
int ellipseRGBA(SDL_Surface* dst,
               Sint16 x, Sint16 y,
               Sint16 rx, Sint16 ry,
               Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a filled ellipse with center at (x, y), and x axis radius rx,
// and y axis radius ry, and RGBA color (r,g,b,a). rx can also
// be thought of as the distance from the center to the left/right
// most point. ry is the same, only from the top/bottom most point.
int filledEllipseRGBA(SDL_Surface* dst,
                    Sint16 x, Sint16 y,
                    Sint16 rx, Sint16 ry,
                    Uint8 r, Uint8 g, Uint8 b, Uint8 a);

// Draw a polygon with RGBA color (r,g,b,a). vx and vy are arrays of
// x and y values. n is the size of vx and vy.
int polygonRGBA(SDL_Surface* dst,
               Sint16* vx, Sint16* vy,
               int n,
               Uint8 r, Uint8 g, Uint8 b, Uint8 a);

```

```

// Draw a filled polygon with RGBA color (r,g,b,a). vx and vy are
arrays of
// x and y values. n is the size of vx and vy.
int filledPolygonRGBA(SDL_Surface* dst,
                      Sint16* vx, Sint16* vy,
                      int n,
                      Uint8 r, Uint8 g, Uint8 b, Uint8 a);

```

Here is some working code for drawing shapes with `SDL_gfx`. I've bolded the parts that relate to `SDL_gfx`.

```

#include "SDL.h"
#include "SDL_gfxPrimitives.h"

const int WINDOW_WIDTH = 640;
const int WINDOW_HEIGHT = 480;
const char* WINDOW_TITLE = "SDL Start";

int main(int argc, char **argv)
{
    SDL_Init( SDL_INIT_VIDEO );

    SDL_Surface* screen = SDL_SetVideoMode( WINDOW_WIDTH,
WINDOW_HEIGHT, 0,
    SDL_HWSURFACE | SDL_DOUBLEBUF );
    SDL_WM_SetCaption( WINDOW_TITLE, 0 );

    SDL_Event event;
    bool gameRunning = true;
    while (gameRunning)
    {
        if (SDL_PollEvent(&event))
        {
            if (event.type == SDL_QUIT)
            {
                gameRunning = false;
            }
        }
        SDL_FillRect(screen, NULL, SDL_MapRGB(screen->format, 0, 0,
0));
        pixelRGBA(screen,
            10, 15,
            255, 255, 255, 255);
        lineRGBA(screen,
            20, 10,
            70, 90,
            255, 0, 0, 255);
    }
}

```

```

trigonRGBA(screen,
            500, 50,
            550, 200,
            600, 150,
            0, 255, 255, 255);

filledTrigonRGBA(screen,
                200, 200,
                300, 50,
                400, 200,
                0, 0, 255, 255);

rectangleRGBA(screen,
              -10, 300,
              100, 380,
              0, 255, 0, 255);

boxRGBA(screen,
        210, 76,
        325, 300,
        255, 0, 0, 150);

ellipseRGBA(screen,
            600, 400,
            50, 90,
            255, 255, 0, 200);

filledEllipseRGBA(screen,
                  600, 400,
                  25, 150,
                  0, 255, 0, 255);

short x[6] = { 350, 275, 300, 325, 350, 400 };
short y[6] = { 325, 325, 390, 390, 375, 300 };

polygonRGBA(screen,
            x, y,
            6,
            255, 255, 255, 155);

short s[5] = { 400, 450, 450, 425, 300 };
short t[5] = { 400, 410, 450, 425, 500 };

filledPolygonRGBA(screen,
                  s, t,
                  5,
                  255, 0, 255, 155);

    SDL_Flip(screen);
}
SDL_Quit();
return 0;
}

```