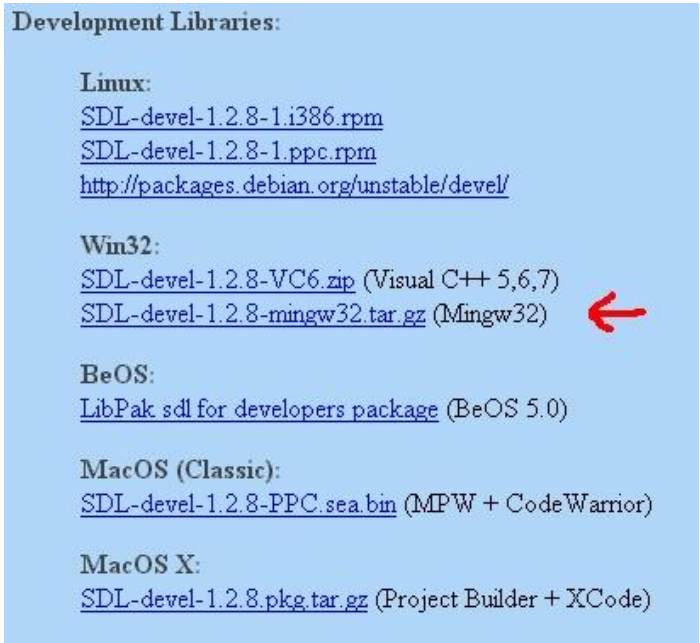
Outline

GUI Development with SDL

Setting up SDL in Eclipse

Setting up SDL in Eclipse

1. First thing you need to do is download SDL headers and binaries from :



Development Libraries:

Linux:
SDL-devel-1.2.8-1.i386.rpm
SDL-devel-1.2.8-1.ppc.rpm
http://packages.debian.org/unstable/devel/

Win32:
SDL-devel-1.2.8-VC6.zip (Visual C++ 5,6,7)
SDL-devel-1.2.8-mingw32.tar.gz (Mingw32)

BeOS:
LibPak sdl for developers package (BeOS 5.0)

MacOS (Classic):
SDL-devel-1.2.8-PPC.sea.bin (MPW + CodeWarrior)

MacOS X:
SDL-devel-1.2.8.pkg.tar.gz (Project Builder + XCode)

http://www.libsdl.org/download-1.2.php

Setting up SDL in Eclipse

2. Copy the contents of the lib subfolder to the MinGW lib folder: C:\MinGW\lib.

3. After that, open the include subfolder in the archive and extract the folder named "SDL" to the MinGW include folder: C:\MinGW\include\SDL.

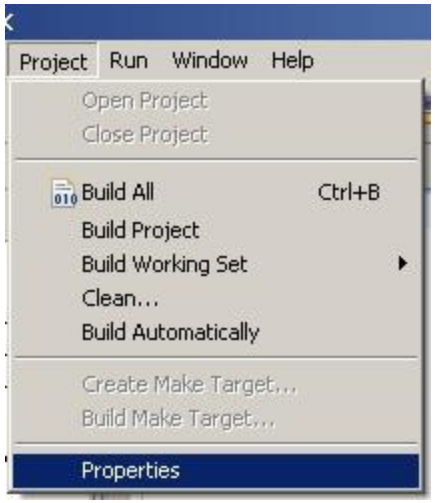
Setting up SDL in Eclipse

4. Now take the SDL.dll from the archive (it should be inside the bin subfolder) and extract it.

   You're going to put this in the same directory as your exe when you compile it.

5. Now start up Eclipse and start a new project.

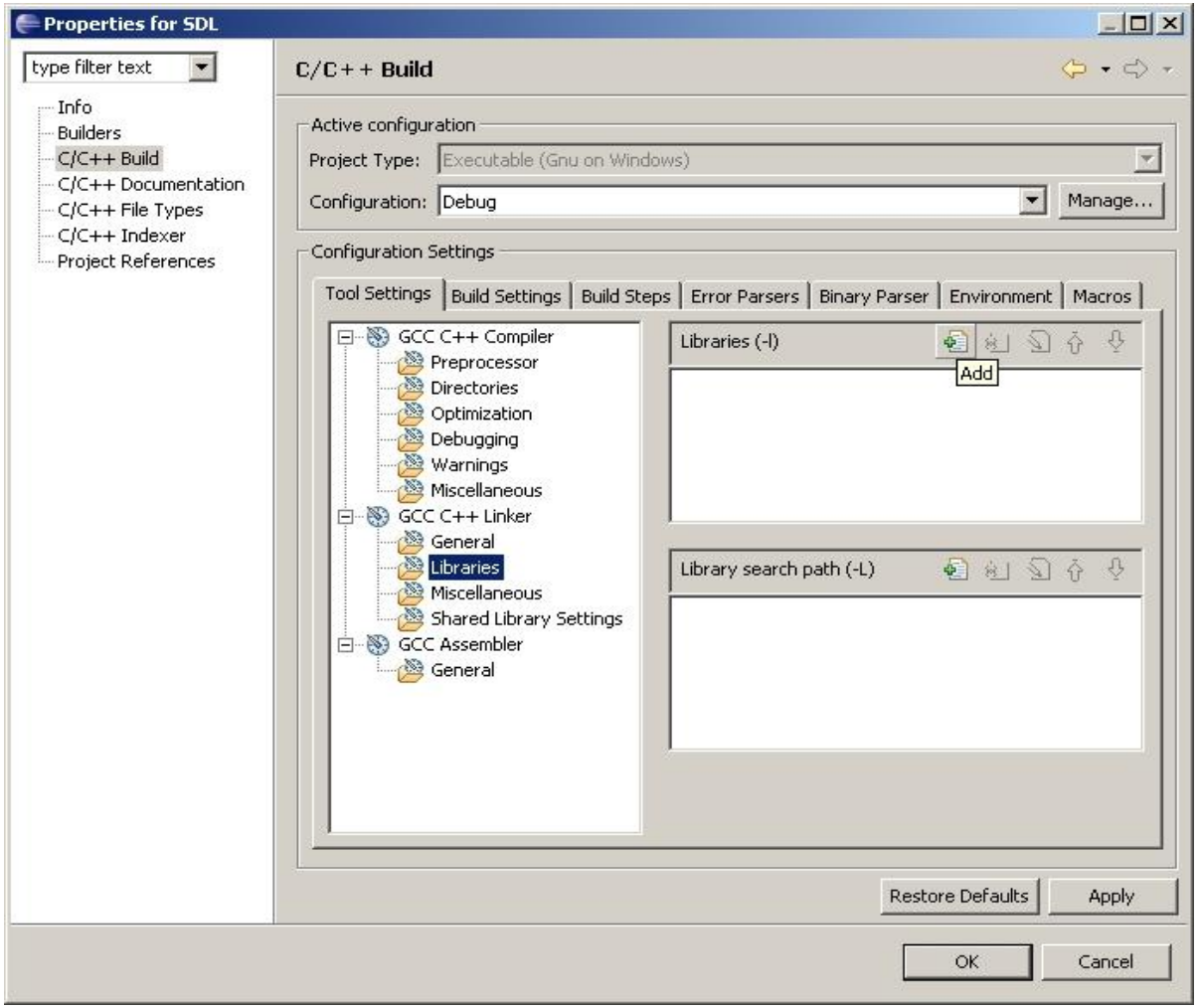
6. Go to the project properties.



Setting up SDL in Eclipse

7. Go to the C/C++ Build menu, then the Libraries submenu.

In the Libraries submenu click add.

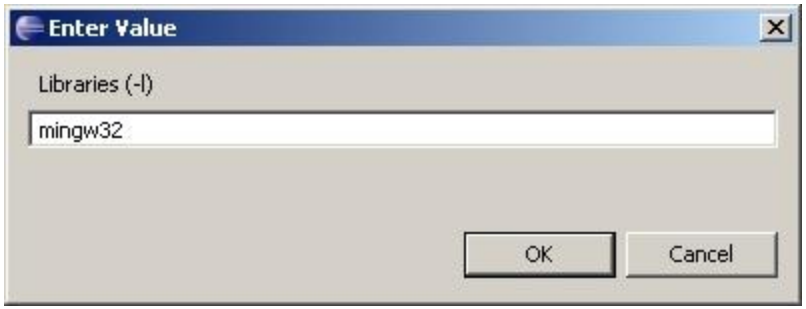


Setting up SDL in Eclipse

11. Then paste in:

“mingw32” and click ok.

Do the same with:
“SDLmain” and “SDL”



Getting an Image on the Screen

- SDL_Surface

  - is a pointers to image

- SDL_Init(SDL_INIT_EVERYTHING);

  - initializes all the SDL subsystems so we can start using SDL's graphics functions.

- SDL_SetVideoMode()

  - set up a 640 pixel wide, 480 pixel high window that has 32 bits per pixel

- (SDL_SWSURFACE)

  - sets up the surface in software memory

  - returns a pointer to the window surface

Getting an Image on the Screen

- SDL_LoadBMP()
  - takes in a path to a bitmap file as an argument
  - returns a pointer
  - returns NULL if there was an error in loading the image
- SDL_BlitSurface()
  - apply image to screen
  - first argument is the surface we're using
  - third argument is the surface we're going to blit on to

Getting an Image on the Screen

- SDL_Flip(screen)

  - update screen

- SDL_Delay(2000)

  - pause from 2000 milliseconds (2 seconds)

- SDL_FreeSurface(hello);

  - free the loaded image

- SDL_Quit()

  - quit SDL

```
1  /*This source code copyrighted by Lazy Foo' Productions (2004-2013)
3
4  //Include SDL functions and datatypes
5  #include "SDL/SDL.h"
6
7  int main( int argc, char* args[] )
8  {
9      //The images
10     SDL_Surface* hello = NULL;
11     SDL_Surface* screen = NULL;
12
13     //Start SDL
14     SDL_Init( SDL_INIT_EVERYTHING );
15
16     //Set up screen
17     screen = SDL_SetVideoMode( 640, 480, 32, SDL_SWSURFACE );
18
19     //Load image
20     hello = SDL_LoadBMP( "hello.bmp" );
21
22     //Apply image to screen
23     SDL_BlitSurface( hello, NULL, screen, NULL );
24
25     //Update Screen
26     SDL_Flip( screen );
27
28     //Pause
29     SDL_Delay( 2000 );
30
31     //Free the loaded image
32     SDL_FreeSurface( hello );
33
34     //Quit SDL
35     SDL_Quit();
36
37     return 0;
38 }
```

Optimized Surface Loading and Blitting

Optimized Surface Loading and Blitting

- The attributes of the screen

  - const int SCREEN_WIDTH = 640;
  - const int SCREEN_HEIGHT = 480;
  - const int SCREEN_BPP = 32;

- Image loading function

  - returns a pointer to the optimized version of the loaded image

  - blit a surface onto another surface that is a different format will have to change the format on the fly which causes slow down

  if(loadedImage != NULL)

  - If nothing went wrong in loading the image

- SDL_DisplayFormat()

  - creates a new version of "loadedImage" in the same format as the screen

Optimized Surface Loading and Blitting

- Surface blitting function
  - takes in the coordinates
  - surface you're going to blit
  - surface you're going to blit it to
- SDL_Rect
  - data type that represents a rectangle
- SDL_BlitSurface()
  - apply image to screen
  - first argument is the surface we're using
  - third argument is the surface we're going to blit on to
  - fourth argument holds the offsets

Optimized Surface Loading and Blitting

- When using SDL, you should always use:

```
int main( int argc, char* args[] )
```

or

```
int main( int argc, char** args )
```

- Initialize all SDL subsystems

```
if( SDL_Init( SDL_INIT_EVERYTHING ) == -1)
    return 1;
```

Optimized Surface Loading and Blitting

- If there was an error in setting up the screen

```
if( screen == NULL )
        return 1;
```

- Set the window caption

```
SDL_WM_SetCaption( "Hello World", NULL );
```

- Update the screen

```
if( SDL_Flip( screen ) == -1 )
        return 1;
```

```
1  /*This source code copyrighted by Lazy Foo' Productions (2004-2013)
3
4  //The headers
5  #include "SDL/SDL.h"
6  #include <string>
7
8  //The attributes of the screen
9  const int SCREEN_WIDTH = 640;
10 const int SCREEN_HEIGHT = 480;
11 const int SCREEN_BPP = 32;
12
13 //The surfaces that will be used
14 SDL_Surface *message = NULL;
15 SDL_Surface *background = NULL;
16 SDL_Surface *screen = NULL;
17
18 SDL_Surface *load_image( std::string filename )
19 {
20     //Temporary storage for the image that's loaded
21     SDL_Surface* loadedImage = NULL;
22
23     //The optimized image that will be used
24     SDL_Surface* optimizedImage = NULL;
25
26     //Load the image
27     loadedImage = SDL_LoadBMP( filename.c_str() );
28
29     //If nothing went wrong in loading the image
30     if( loadedImage != NULL )
31     {
32         //Create an optimized image
33         optimizedImage = SDL_DisplayFormat( loadedImage );
34
35         //Free the old image
36         SDL_FreeSurface( loadedImage );
37     }
38
```

```
    //Return the optimized image
    return optimizedImage;
}

void apply_surface( int x, int y, SDL_Surface* source, SDL_Surface* destination )
{
    //Make a temporary rectangle to hold the offsets
    SDL_Rect offset;

    //Give the offsets to the rectangle
    offset.x = x;
    offset.y = y;

    //Blit the surface
    SDL_BlitSurface( source, NULL, destination, &offset );
}

int main( int argc, char* args[] )
{
    //Initialize all SDL subsystems
    if( SDL_Init( SDL_INIT_EVERYTHING ) == -1 )
    {
        return 1;
    }

    //Set up the screen
    screen = SDL_SetVideoMode( SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP, SDL_SWSURFACE );

    //If there was an error in setting up the screen
    if( screen == NULL )
    {
        return 1;
    }

    //Set the window caption
    SDL_WM_SetCaption( "Hello World", NULL );
```

```
75
76      //Load the images
77      message = load_image( "hello.bmp" );
78      background = load_image( "background.bmp" );
79
80      //Apply the background to the screen
81      apply_surface( 0, 0, background, screen );
82      apply_surface( 320, 0, background, screen );
83      apply_surface( 0, 240, background, screen );
84      apply_surface( 320, 240, background, screen );
85
86      //Apply the message to the screen
87      apply_surface( 180, 140, message, screen );
88
89      //Update the screen
90      if( SDL_Flip( screen ) == -1 )
91      {
92          return 1;
93      }
94
95      //Wait 2 seconds
96      SDL_Delay( 2000 );
97
98      //Free the surfaces
99      SDL_FreeSurface( message );
100     SDL_FreeSurface( background );
101
102     //Quit SDL
103     SDL_Quit();
104
105     return 0;
106 }
```

Setup SDL Extension Libraries

Setup SDL Extension Libraries

1. SDL_image is located on:

http://www.libsdl.org/projects/SDL_image/release-1.2.html

Binary:

- Linux
  - SDL_image-1.2.4-1.i386.rpm
  - SDL_image-1.2.4-1.ppc.rpm
  - SDL_image-devel-1.2.4-1.i386.rpm
  - SDL_image-devel-1.2.4-1.ppc.rpm
- Win32
  - SDL_image-1.2.4-win32.zip
  - SDL_image-devel-1.2.4-VC6.zip
- MacOS X
  - SDL_image-1.2.4.pkg.tar.gz

Setup SDL Extension Libraries

Every extension libary has 3 essential parts:

- The header file.
- The lib file.
- The *.dll file(s)

Setup SDL Extension Libraries

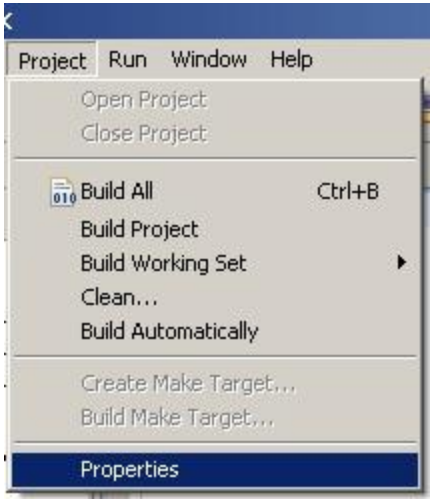
2. extract the header file inside to the SDL to C:\MinGW\include\SDL

3. Copy lib folder from arhive to C:\MinGW\lib

4. extract all of the *.dll file(s) from the archive and put them in the same directory as your exe

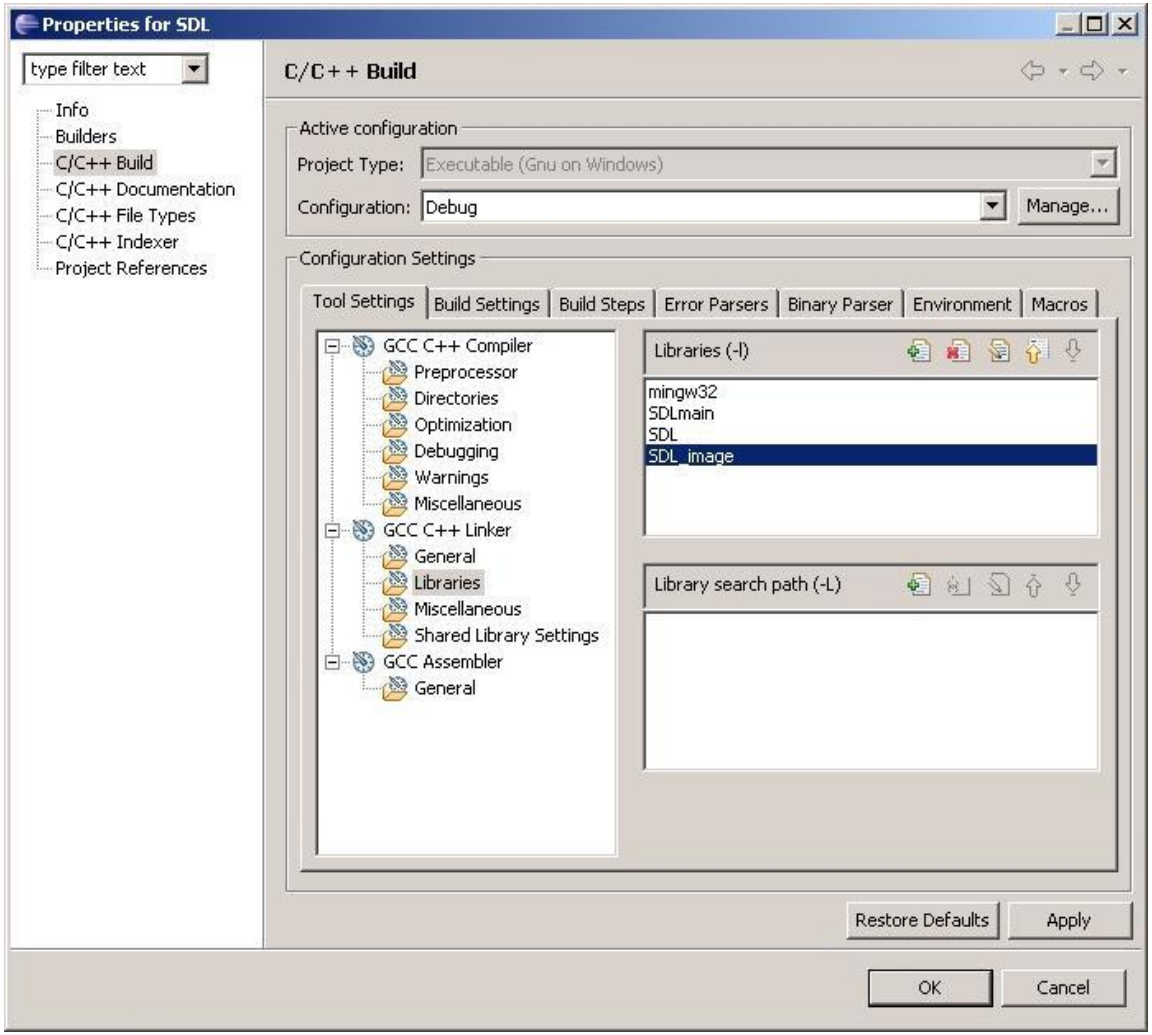
Setup SDL Extension Libraries

5. open up your SDL project and go to the project properties



Setup SDL Extension Libraries

6. In the libraries section add in: SDL_image



Setup SDL Extension Libraries

include the header file

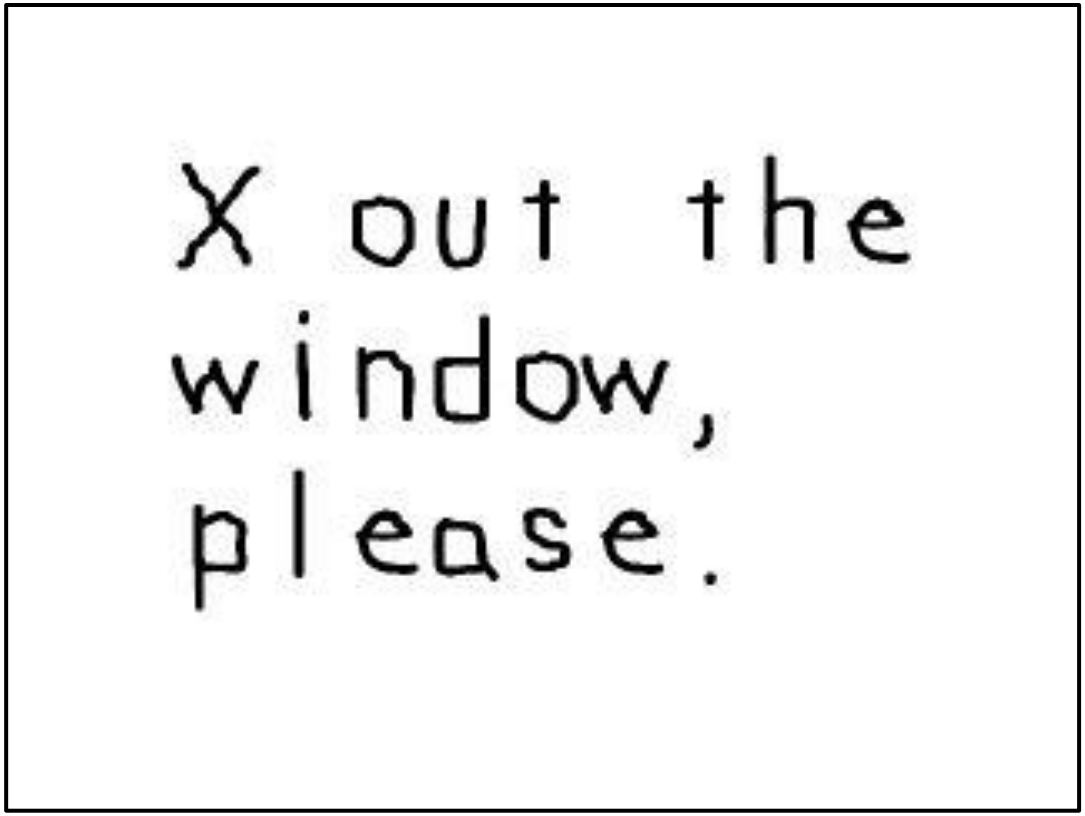
- #include "SDL/SDL_image.h“

IMG_Load()

- same as SDL_LoadBMP()
- can load BMP, PNM, XPM, LBM, PCX, GIF, JPEG, TGA and PNG files

Event Driven Programming

Event Driven Programming



Event Driven Programming

- Make sure the program waits for a quit
  - bool quit = false;
- SDL_Event
  - A SDL_Event structure stores event data for us to handle
- SDL_PollEvent()
  - take an event from the queue and sticks its data in our event structure

Event Driven Programming

```
131
132     //While the user hasn't quit
133     while( quit == false )
134     {
135         //While there's an event to handle
136         while( SDL_PollEvent( &event ) )
137         {
138             //If the user has Xed out the window
139             if( event.type == SDL_QUIT )
140             {
141                 //Quit the program
142                 quit = true;
143             }
144         }
145     }
146
```

Event Driven Programming

Replace

- SDL_LoadBMP(filename.c_str())

with

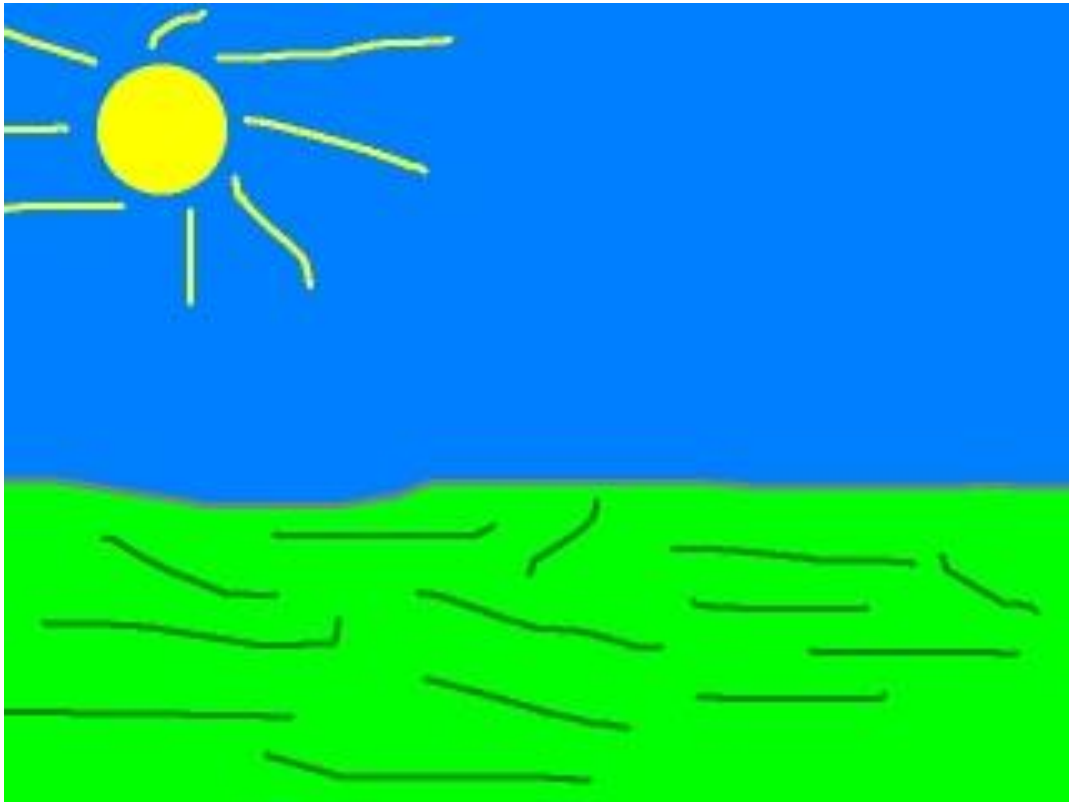
- IMG_Load(filename.c_str())

to load various images

Color Keying

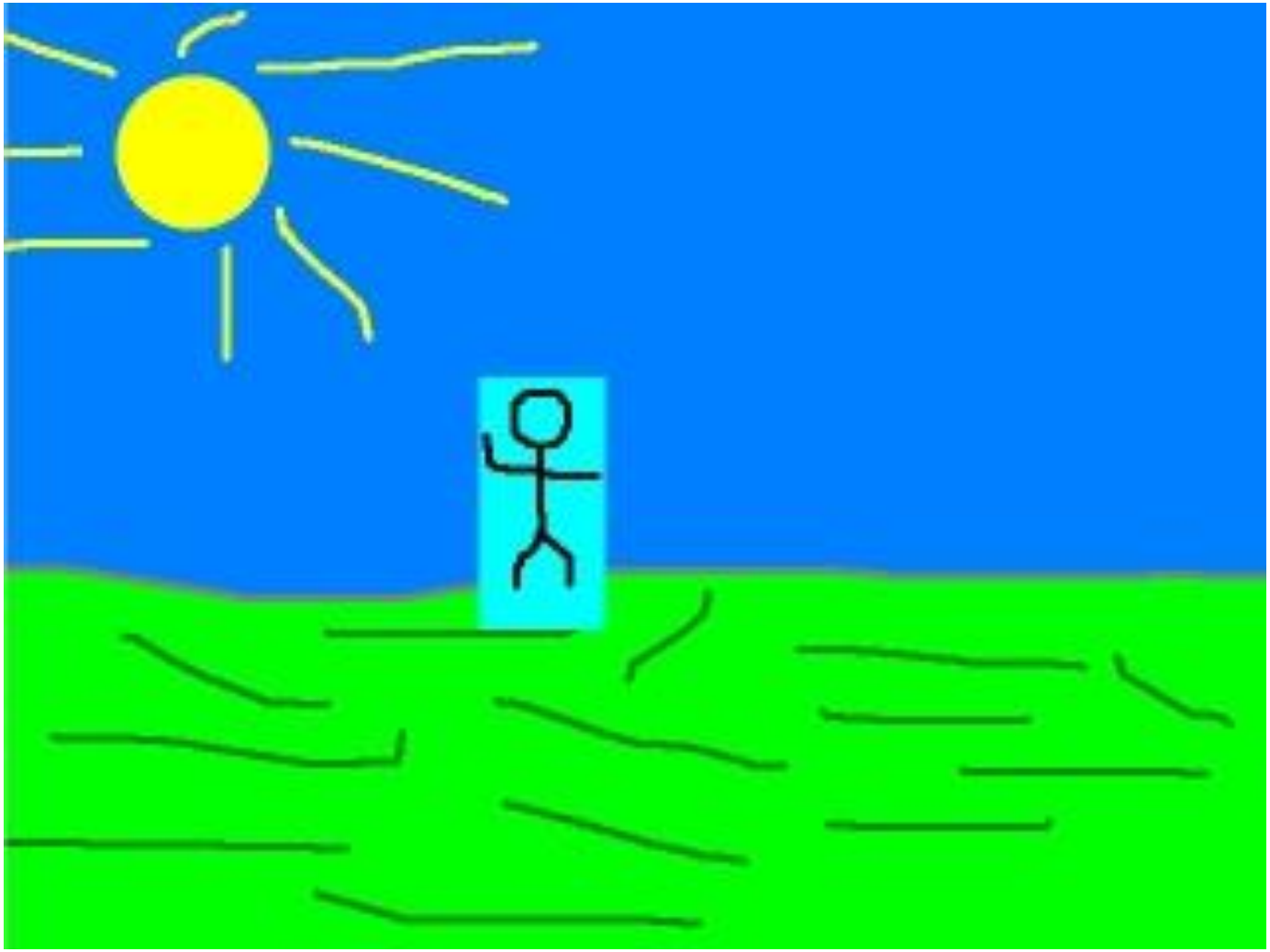
Color Keying

Say you wanted to apply this stick figure named "Foo“ to this background



Color Keying

Result



Color Keying

The color key is typically set when the image is loaded

```
41
42          //If the image was optimized just fine
43          if( optimizedImage != NULL )
44          {
45              //Map the color key
46              Uint32 colorkey = SDL_MapRGB( optimizedImage->format, 0, 0xFF, 0xFF );
47
48              //Set all pixels of color R 0, G 0xFF, B 0xFF to be transparent
49              SDL_SetColorKey( optimizedImage, SDL_SRCCOLORKEY, colorkey );
50          }
51      }
52
```

Color Keying

- SDL_MapRGB()

  - take the red, green, and blue values and give us the pixel value back in the same format as the surface

- SDL_SetColorKey(optimizedImage, SDL_SRCCOLORKEY, colorkey)

  - first argument is the surface we want to set

  - second argument holds the flags we want to place,

  SDL_SRCCOLORKEY flag makes sure that we use the color key when blitting this surface onto another surface

  - third argument is the color we want to set as the color key