A BEGINNERS GUIDE TO PROGRAMING

Book 3 supplemental

SDL2 and SDL_bgi Library

Axle

9 March 2023

Table of Contents

Licence

With many thanks to Daniel Moore for proof reading and corrections as well as the many ideas, examples and suggestions that have been provided.

Revisions

Version	Date	Notes
Draft V 0.1	09/03/2023	Essential SDL2 and SDL_bgi tutorial complete (C language only).

TODO:

Add guide for FreeBASIC and Python3.

Add some basic API coding guides from the Space shooter demo and Quartz Clock demo.

Preface

Description from the Author: Guido Gonzato, PhD.

SDL_bgi is a graphics library (GRAPHICS.H) for C, C++, WebAssembly, and Python. It's based on SDL2 and it's portable on many platforms.

Its name refers to BGI, the Borland Graphics Interface that was the 'de facto' standard in PC graphics back in DOS days; it was made popular by Borland Turbo C/C++ compilers. I wrote SDL_bgi because I wanted a simple to use but fast graphics library for my experiments with fractals and cellular automata, using the BGI syntax I'm used to.

SDL_bgi is functionally compatible with the BGI implementation in Turbo C 2.01 and Borland C++ 1.0; for instance, it compiles and runs the original bgidemo.c. SDL_bgi also provides nearly full compatibility with another BGI implementation, WinBGIm (see links below). One of the aims of SDL_bgi is the preservation of old software written for BGI; but not only that.

SDL_bgi provides graphics primitives, and is much easier to use than plain SDL2; it should be especially useful for beginners, i.e. in introductory programming courses. SDL_bgi is pretty fast, and in addition to BGI compatibility it provides extensions for ARGB colours, mouse support, vector fonts, and multiple windows. Native SDL2 functions can be used alongside SDL_bgi functions. SDL_bgi can also be used in programs written in C++ or Python.

SDL_bgi is written in C, and it should compile on any platform supported by SDL2. It has been tested on GNU/Linux, MS Windows (MSYS2 and Mingw-w64, CodeBlocks, Dev-C++), macOS (High Sierra and Catalina), Raspios (ARM, i386), and WebAssembly (Emscripten). A few example programs in C and Python are provided in the demo/ directory.

From: https://sourceforge.net/projects/sdl-bgi/files/

SDL_bgi is based on SDL2, and is portable to any platform supported by SDL2: Windows, macOS, GNU/Linux, and WebAssembly via Emscripten.

SDL_bgi can be used to port old programs written for Turbo/Borland C to modern systems. And, of course, to write new graphics programs with minimal effort: BGI, once extremely popular, was probably the simplest way to implement presentation graphics in C programs. The same ease of programming can be obtained on modern systems. Programming fractals, cellular automata, geometry, physics models etc. is a breeze with SDL_bgi.

From: https://sdl-bgi.sourceforge.io/

Installing and using SDL2 and SDL-BGI (Graphics.h) Libraries.

NOTE! Many SDL-BGI apps will not run well under VirtualBox clients.

NOTE! It is important to have control over the Frames Per Second (FPS) for video like applications. This can be set via edelay() and the ticks counter that measures the loops per second by measuring the start time and end time for each main loop. edelay() is a busy wait, meaning that the CPU is in a tight busy loop. To make use of the OS task scheduler we can make limited use of the SDL_Delay() function which temporarily will place the application into an idle state reducing the CPU percentage use. SDL_Delay() is not fully compatible with SDL_BGI and should only be used in very limited circumstances as it can cause the application to stop responding.

SDL2 Setup Windows

https://lazyfoo.net/tutorials/SDL/01_hello_SDL/windows/mingw/index.php

https://lazyfoo.net/tutorials/SDL/01_hello_SDL/index2.php

https://github.com/libsdl-org/SDL/releases

Download "SDL2-devel-2.26.2-mingw.zip" (or a later version if you are confident of the compatibility with SDL-BGI). Ensure that it is the development version (-devel-).

DLL runtime redistributables (only exported with your compiled application).

You can just use the DLL from the devel package for testing on other machines.

"SDL2-2.26.2-win32-x64.zip"

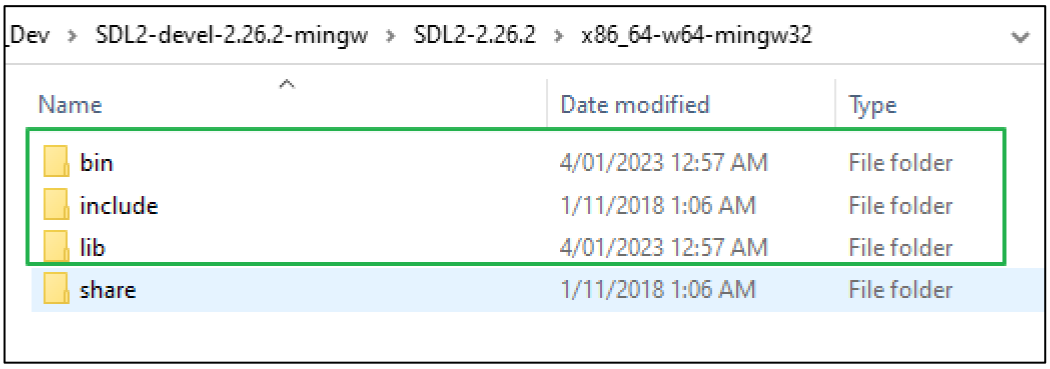
Unpack "SDL2-devel-2.26.2-mingw.zip"

Copy the contents of "x86_64-w64-mingw32" to corresponding "C:\Dev-Cpp-Embarcadero\*.*"

\bin

\include

\lib



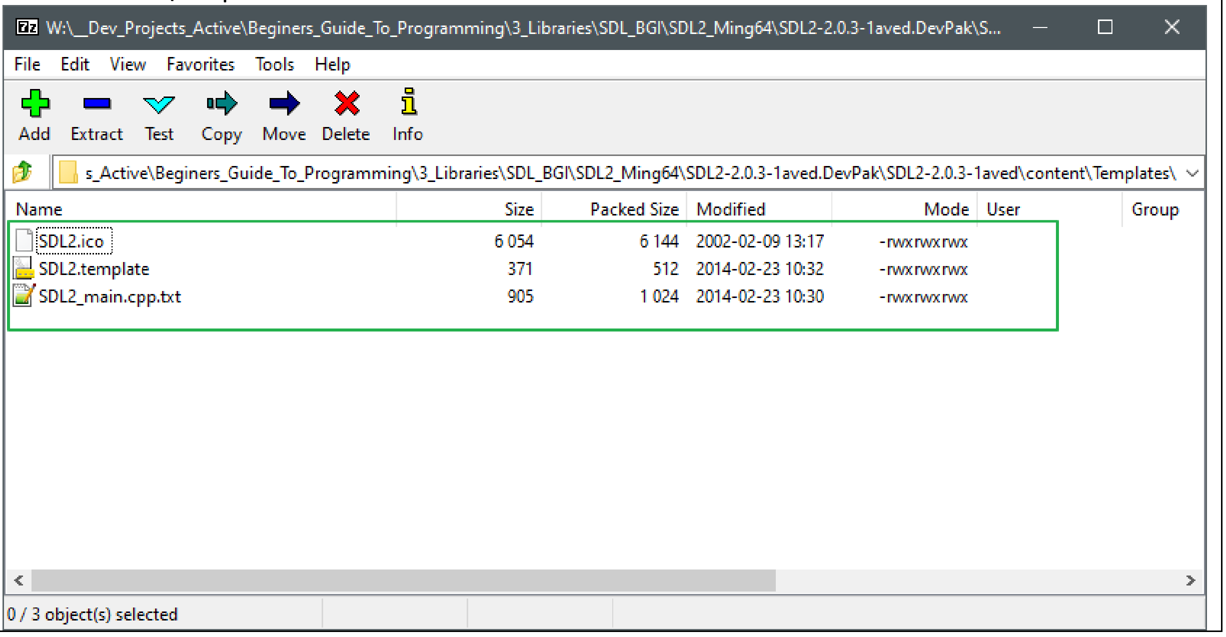
Dev > SDL2-devel-2.26.2-mingw > SDL2-2.26.2 > x86_64-w64-mingw32

Name	Date modified	Type
bin	4/01/2023 12:57 AM	File folder
include	1/11/2018 1:06 AM	File folder
lib	4/01/2023 12:57 AM	File folder
share	1/11/2018 1:06 AM	File folder

Note that SDL2 will be in a sub directory of include.

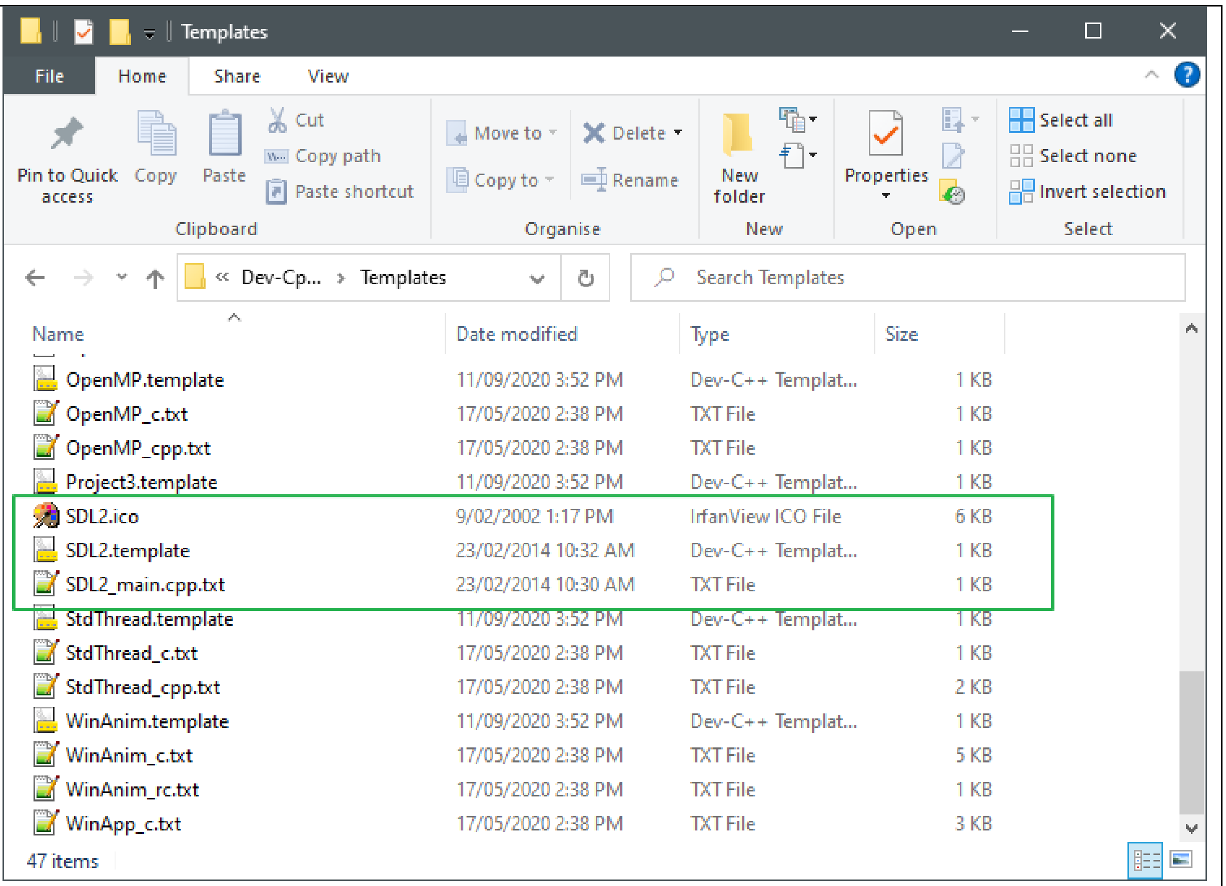
“MyDev-C++\include\SDL2\*.h”

Optional
Download “SDL2-2.0.3-1aved.DevPak” from https://sourceforge.net/projects/devpacks/files/
or direct https://sourceforge.net/projects/devpacks/files/SDL2-2.0.3-1aved.DevPak/download
Open SDL2-2.0.3-1aved.DevPak (or unpack) using 7-Zip file manager.
Copy the contents from SDL2-2.0.3-1aved.DevPak “.\templates” to “C:\Dev-Cpp-Embarcadero\templates”.


W:_Dev_Projects_Active\Beginers_Guide_To_Programming\3_Libraries\SDL_BGI\SDL2_Ming64\SDL2-2.0.3-1aved.DevPak\S...
File
Edit
View
Favorites
Tools
Help
Add
Extract
Test
Copy
Move
Delete
Info
s_Active\Beginers_Guide_To_Programming\3_Libraries\SDL_BGI\SDL2_Ming64\SDL2-2.0.3-1aved.DevPak\SDL2-2.0.3-1aved\content\Templates\

Name	Size	Packed Size	Modified	Mode	User	Group
SDL2.ico	6 054	6 144	2002-02-09 13:17	-rwxrwxrwx		
SDL2.template	371	512	2014-02-23 10:32	-rwxrwxrwx		
SDL2_main.cpp.txt	905	1 024	2014-02-23 10:30	-rwxrwxrwx		

0 / 3 object(s) selected

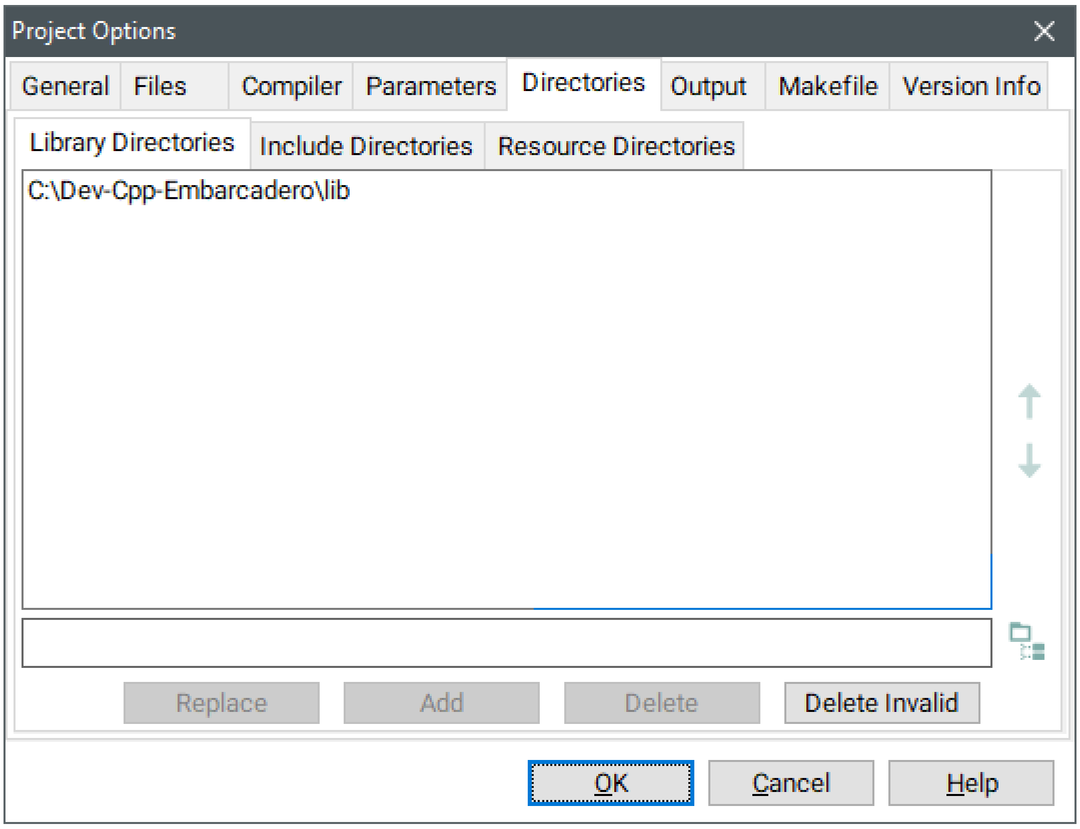


Name	Date modified	Type	Size
OpenMP.template	11/09/2020 3:52 PM	Dev-C++ Templat...	1 KB
OpenMP_c.txt	17/05/2020 2:38 PM	TXT File	1 KB
OpenMP_cpp.txt	17/05/2020 2:38 PM	TXT File	1 KB
Project3.template	11/09/2020 3:52 PM	Dev-C++ Templat...	1 KB
SDL2.ico	9/02/2002 1:17 PM	IrfanView ICO File	6 KB
SDL2.template	23/02/2014 10:32 AM	Dev-C++ Templat...	1 KB
SDL2_main.cpp.txt	23/02/2014 10:30 AM	TXT File	1 KB
StdThread.template	11/09/2020 3:52 PM	Dev-C++ Templat...	1 KB
StdThread_c.txt	17/05/2020 2:38 PM	TXT File	1 KB
StdThread_cpp.txt	17/05/2020 2:38 PM	TXT File	2 KB
WinAnim.template	11/09/2020 3:52 PM	Dev-C++ Templat...	1 KB
WinAnim_c.txt	17/05/2020 2:38 PM	TXT File	5 KB
WinAnim_rc.txt	17/05/2020 2:38 PM	TXT File	1 KB
WinApp_c.txt	17/05/2020 2:38 PM	TXT File	3 KB

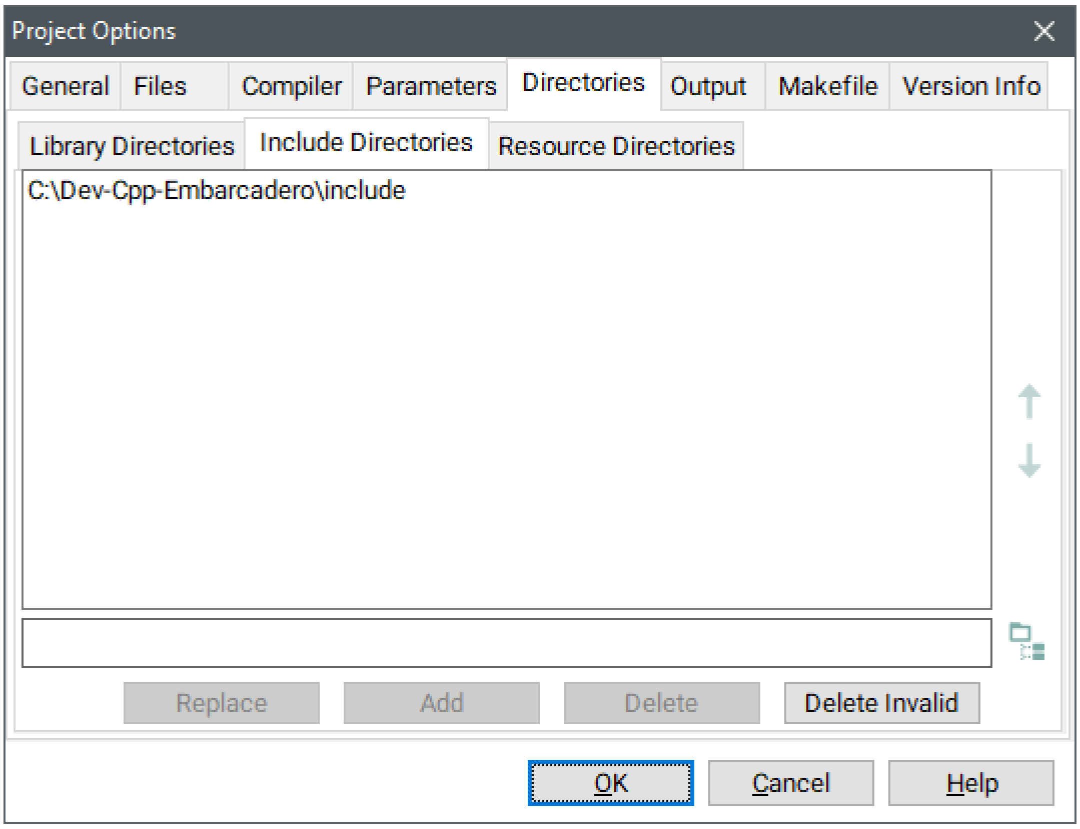
47 items

This will give you an SDL2 project option when start Dev-C++ but is not required, You can use the set up that I have described in this document for a better outcome using C language.

Start at new project and add the SDL2 paths to the project under Project -> Project Options...



C:\Dev-Cpp-Embarcadero\lib



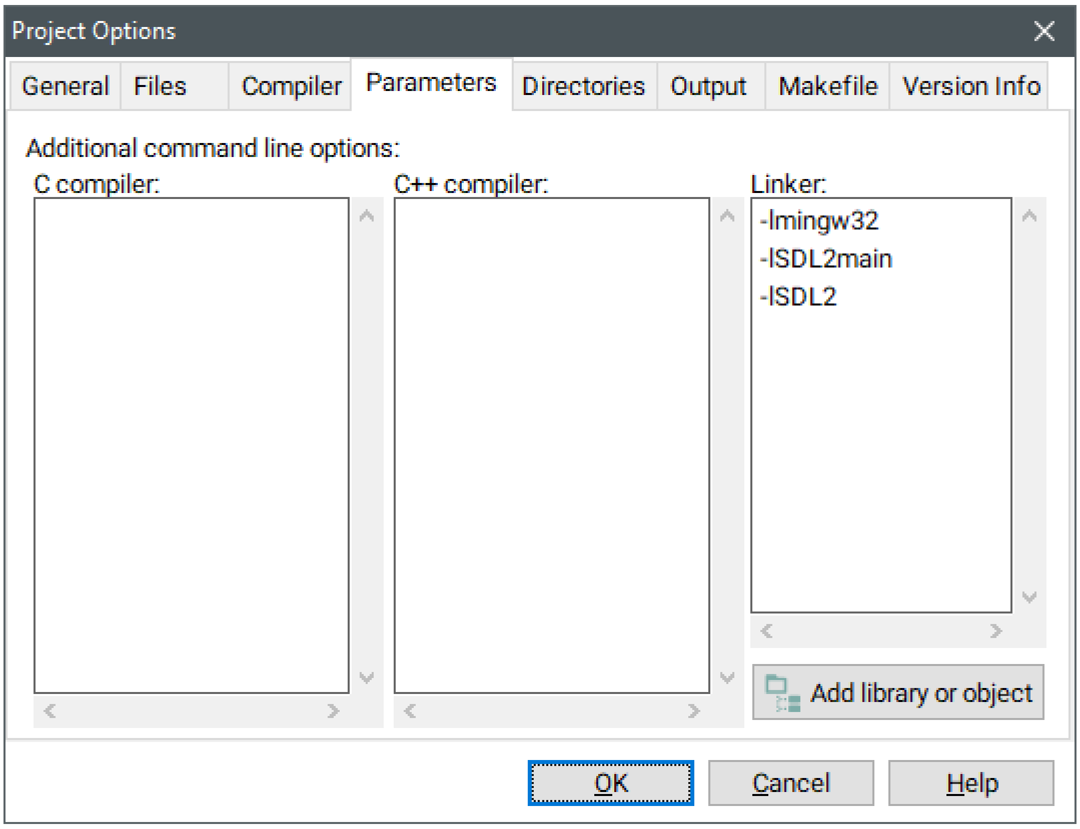
C:\Dev-Cpp-Embarcadero\include

Add the following to linker options as shown in the image below (we will also add SDL-BGI later).

Linker:

-lmingw32
-lSDL2main
-lSDL2

`sdl2-config --libs` is not used by the compiler on Windows.



```
-lmingw32
-lSDL2main
-lSDL2
```

Command line compile example:
// g++ $(OBJS) -IC:\mingw_dev_lib\include\SDL2 -LC:\mingw_dev_lib\lib -w -Wl,-subsystem,windows **-lmingw32 -lSDL2main -lSDL2** -o $(OBJ_NAME)

Copy "SDL2.dll" from the downloaded "SDL2-devel-2.26.2-mingw.zip" or the "SDL2-2.26.2-win32-x64.zip (Runtime library)" into the project directory where your final exe will exist.

Use the following #include for basic SDL applications:

For more complex application some additional headers may be required.

#include <SDL2/SDL.h>// lSDL2main -lSDL2

SDL.h is in a sub directory of "C:\Dev-Cpp-Embarcadero\include" + "SDL2/SDL.h"

Create a main.c file in your project and copy the following source to test SDL2.

Example 1: "SDL_Helloworld_1.c"

```
//Using SDL and standard IO

// -lmingw32 -lSDL2main -lSDL2

#include <SDL2/SDL.h>
#include <stdio.h>
#include <stdbool.h>

//Screen dimension constants
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;

int main( int argc, char* args[] )
    {
    //The window we'll be rendering to
    SDL_Window* window = NULL;

    //The surface contained by the window
    SDL_Surface* screenSurface = NULL;

    //Initialize SDL
    if( SDL_Init( SDL_INIT_VIDEO ) < 0 )
        {
        printf( "SDL could not initialize! SDL_Error: %s\n", SDL_GetError() );
        }
    else
        {
        //Create window
        window = SDL_CreateWindow( "SDL Tutorial", SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW_SHOWN );
        if( window == NULL )
            {
            printf( "Window could not be created! SDL_Error: %s\n", SDL_GetError()
);
            }
        else
            {
            //Get window surface
            screenSurface = SDL_GetWindowSurface( window );

            //Fill the surface white
            SDL_FillRect( screenSurface, NULL, SDL_MapRGB( screenSurface->format,
0xFF, 0xFF, 0xFF ) );

            //Update the surface
            SDL_UpdateWindowSurface( window );

            //Hack to get window to stay up
            SDL_Event e;
            bool quit = false;
            while( quit == false )
                {
                while( SDL_PollEvent( &e ) )
                    {
                    if( e.type == SDL_QUIT ) quit = true;
                    }
                }
            }
        }
    //Destroy window
    SDL_DestroyWindow( window );

    //Quit SDL subsystems
    SDL_Quit();

    return 0;
    }
```

Example 2 "SDL_Helloworld_2.c"

```
#include <SDL2/SDL.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>
//using namespace std;  // C++

// -lmingw32 -lSDL2main -lSDL2

int main(int argc, char** argv)
    {
    srand(time(NULL));
    SDL_Init(SDL_INIT_EVERYTHING);
    SDL_Window *window = SDL_CreateWindow("Hello World!", 100, 100, 640, 480,
SDL_WINDOW_SHOWN);
    SDL_Surface* screenSurface= SDL_GetWindowSurface(window);
//     SDL_Surface* img =SDL_LoadBMP("");
//     SDL_BlitSurface(img,NULL,screenSurface,NULL);
    bool exit=false;
    SDL_Event e;
    SDL_FillRect(screenSurface,NULL, SDL_MapRGB(screenSurface->format,128,0,120));
    SDL_UpdateWindowSurface(window);
    while(!exit)
        {
        while(SDL_PollEvent(&e)!=0)
            {
            if(e.type == SDL_QUIT)
                {
                exit=true;
                }
            if(e.type == SDL_KEYDOWN)
                {
                switch(e.key.keysym.sym)
                    {
                    case SDLK_UP:
                        exit=true;
                        break;
                    }
                }
            }


        }
    //SDL:FreeSurface(img);
    SDL_DestroyWindow(window);
    SDL_Quit();
    return 0;
    }
```

If the 2 example applications worked, then we have set up SDL2 files and project correctly.

SDL_BGI Setup

https://sdl-bgi.sourceforge.io/

https://sourceforge.net/projects/sdl-bgi/

Please Note! SDL_BGI Vers 3

Download “SDL_bgi-3.0.0-win.zip” from the following link.

https://sdl-bgi.sourceforge.io/

Unpack “SDL_bgi-3.0.0-win.zip”.

Note binaries for Windows (MSYS2 + mingw-w64, CodeBlocks, Dev-C++), I am using “mingw-w64\SDL_bgi.dll”.

Take careful note of the placement of the following files.

Copy .\src **graphics.h** to C:\Dev-Cpp-Embarcadero\include

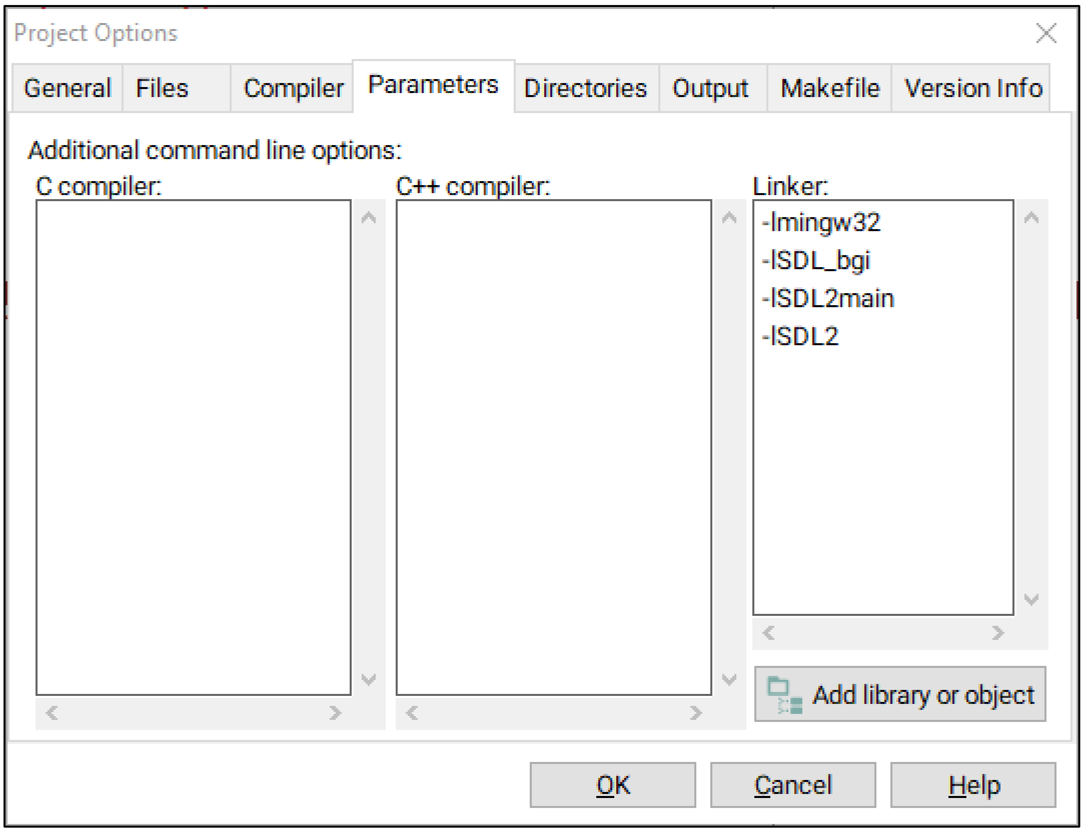
Copy .\src **SDL_bgi.h** to C:\Dev-Cpp-Embarcadero**\include\SDL2**

Copy . \bin**Mingw64** SDL_bgi.dll to **C:\Dev-Cpp-Embarcadero\lib**

Copy . \bin**Mingw64** SDL_bgi.dll to C:\Dev-Cpp-Embarcadero\bin

Copy . \bin**Mingw64** SDL_bgi.dll to you project directory (runtime).

Add -lSDL_bgi to your project linker settings along with the SDL2 linker settings.



Additional command line options:

C compiler:

C++ compiler:

Linker:

```
-lmingw32
-lSDL_bgi
-lSDL2main
-lSDL2
```

Use the following example to test SDL_BGI:

(There are many examples in the downloaded SDL-BGI archive. Some of the source may require minor adjustments to run correctly as the examples seem to be for earlier SDL-BGI versions)

SDL2.h is included by the graphics.h so only graphics.h is required.

Example: "SDL_bgi-3.0.0\test\arc.c"

```
/* arc example */
/* SDL_bgi-3.0.0\test\arc.c */

#include <graphics.h>

int main(int argc, char *argv[])
{
  /* request autodetection */
  int gdriver = DETECT, gmode;
  int midx, midy;
  int stangle = 45, endangle = 135;
  int radius = 100;

  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;
  setcolor(getmaxcolor());
  /* draw arc */
  arc(midx, midy, stangle, endangle, radius);

  /* clean up */
  getch();
  closegraph();
  return 0;
}
```

Example: “SDL_bgi-3.0.0\test\pieslice.c”

```
/* pieslice example */
/* SDL_bgi-3.0.0\test\ pieslice.c */

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h> // Comment this line out!

int main(int argc, char *argv[])
{
  /* request autodetection */
  int gdriver = DETECT, gmode;
  int midx, midy;
  int stangle = 45, endangle = 135, radius = 100;

  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  /* set fill style and draw a pie slice */
  setfillstyle(EMPTY_FILL, getmaxcolor());
  pieslice(midx, midy, stangle, endangle, radius);

  /* clean up */
  getch();
  closegraph();
  return 0;
}
```

NOTE! showinfobox() must be changed to showerrorbox()

Example: mandelbrot.c

```
/* mandelbrot.c  -*- C -*-
 * https://sdl-bgi.sourceforge.io/test/mandelbrot.c
 *
 * To compile:
 * gcc -o mandelbrot mandelbrot.c -lSDL_bgi -lSDL2
 *
 * By Guido Gonzato, May 2015-2022
 *
 * This is an unoptimised, simple but effective program for plotting
 * the Mandelbrot set. Left click to zoom in on a point, right click
 * to zoom out, middle click to restore the initial boundary,
 * ESC to quit.

 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>

int max_iter = 100; // max iteration
int maxx, maxy;

void mandelbrot (double, double, double, double);
void explain (void);
void amber_palette (void);
void blue_palette (void);
void purple_palette (void);

// -----

void mandelbrot (double x1, double y1, double x2, double y2)
{
  int xx, yy, counter;
  double dx, dy, x, y, a, b, tx, d;

  dy = (y2 - y1) / maxy;
  dx = dy;

  x = x1;
  for (xx = 0; xx < maxx; xx++) {

    y = y1;
    for (yy = 0; yy < maxy; yy++) {

      counter = 0;
      a = b = 0.0;

      // iteration: z(n+1) = z(n)^2 + c; z = ai + b; c = yi + x

      do {
       tx = a*a - b*b + x;
       b = 2*b*a + y;
       a = tx;
```

```
        d = a*a + b*b;
        counter++;
       } while (d <= 4.0 && counter < max_iter);

       setrgbcolor (counter);
       _putpixel (xx, yy);
       y += dy;

    } // y
    x += dx;

  } // x

} // mandelbrot ()

// -----

void amber_palette (void)
{
  int c;

  // initialize rgb palette entries 0 - (max_iter - 1)
  for (c = 0; c < max_iter; c++)
    setrgbpalette (c, max_iter - c, 50 + 2 * c, c);
  setrgbpalette (max_iter, 0x30, 0, 0x30); // the Mandelbrot set is purple

} // amber_palette ()

// -----

void purple_palette (void)
{
  int c;

  for (c = 0; c < max_iter; c++)
    setrgbpalette (c, 50 + 2 * c, c, max_iter - c);
  setrgbpalette (max_iter, 0, 0, 0); // the Mandelbrot set is black

} // purple_palette ()

// -----

void blue_palette (void)
{
  int c;

  for (c = 0; c < max_iter; c++)
    setrgbpalette (c, 0, c, 50 + 2 * c);
  setrgbpalette (max_iter, 0, 0, 0); // the Mandelbrot set is black

} // blue_palette ()

// -----

//showinfobox
void help (void)
{
  showerrorbox ("Press '1', '2', or '3' to change the palette\n"
                "left click to zoom in on a point\n"
```

```
              "right click to zoom out\n"
              "middle click or 'R' to restore the initial boundary\n"
              "'i' or '+', 'd' or '-' to increase/decrease max iterations\n"
              "arrow keys to move around\n"
              "'Q' to quit the program");
} // help ()

// -----

void explain (void)
{
  int
    i = 0,
    inc = 5,
    c;

  setbkcolor (COLOR (0, 0, 32)); // don't use a palette
  cleardevice ();
  setcolor (COLOR (255, 255, 0));

  settextstyle (TRIPLEX_FONT, HORIZ_DIR, 4);
  settextjustify (CENTER_TEXT, CENTER_TEXT);
  c = 2*textheight ("H");
  outtextxy (maxx / 2, maxy / 2 - 4*c,
          "Press '1', '2', or '3' to change the palette");
  outtextxy (maxx / 2, maxy / 2 - 3*c,
          "left click to zoom in on a point");
  outtextxy (maxx / 2, maxy / 2 - 2*c,
          "right click to zoom out");
  outtextxy (maxx / 2, maxy / 2 - c,
          "middle click or 'R' to restore the initial boundary");
  outtextxy (maxx / 2, maxy / 2,
          "'i' or '+', 'd' or '-' to increase/decrease max iterations");
  outtextxy (maxx / 2, maxy / 2 + c,
          "arrow keys to move around");
  outtextxy (maxx / 2, maxy / 2 + 2*c,
          "'H' to get help");

  outtextxy (maxx / 2, maxy / 2 + 3*c,
          "'Q' to quit the program");

  int
     ev;

  while (1) {

    setcolor (COLOR (i, 0, 0));
    outtextxy (maxx / 2, maxy / 2 + 4*c, "PRESS A KEY OR CLICK TO BEGIN");
    i += inc;
    if (255 == i)
      inc = -5;
    if (0 == i)
      inc = 5;
    delay(1);
    refresh ();

    event ();
    ev = eventtype ();
    if (ev == SDL_KEYDOWN || ev == SDL_MOUSEBUTTONDOWN)
```

```
      break;

  }
  cleardevice ();

  settextstyle (DEFAULT_FONT, HORIZ_DIR, 1);
  settextjustify (LEFT_TEXT, TOP_TEXT);

} // explain ()

// -----

int main (int argc, char *argv[])
{
  int palette, ch, init, redraw, flag;
  double xm, ym, xstep, ystep, xmin, ymin, xmax, ymax;
  char s[20];

  initwindow (0, 0); // fullscreen

  maxx = getmaxx ();
  maxy = getmaxy ();
  // initial coordinates of the middle point of the screen
  xm = -1.2;
  ym = 0.0;
  // initial range: xm +- xstep, ym +- ystep
  ystep = 1.2;
  xstep = ystep * maxx / (double) maxy;
  init = flag = redraw = 1;

  explain ();

  purple_palette ();
  palette = 1;

  ch = 'G';
  while (ch != 'q' && ch != 'Q') {

    xmin = xm - xstep;
    ymin = ym - ystep;
    xmax = xm + xstep;
    ymax = ym + ystep;

    if (redraw) {
      mandelbrot (xmin, ymin, xmax, ymax);
      refresh ();
      if (flag) {
       setcolor (WHITE);
       sprintf (s, "%d", max_iter);
       outtextxy (0, 0, s);
       flag = 0;
       refresh ();
      }
      redraw = 0;
    }

    ch = getevent (); // wait for a key, mouse click, or wheel motion

    switch (ch) {
```

```
    case 'h':
    case 'H':
      help ();
      break;

    case WM_LBUTTONDOWN:
    case WM_WHEELUP:
      xm = xmin + 2 * xstep * mousex () / maxx;
      ym = ymin + 2 * ystep * mousey () / maxy;
      xstep /= 2.0;
      ystep /= 2.0;
      init = 0;
      redraw = 1;
      break;

    case WM_RBUTTONDOWN:
    case WM_WHEELDOWN:
      xstep *= 2.0;
      ystep *= 2.0;
      init = 0;
      redraw = 1;
      break;

    case WM_MBUTTONDOWN:
    case 'r':
    case 'R':
      if (0 == init) {
        xm = -1.2;
        ym = 0.0;
        ystep = 1.2;
        xstep = ystep * (double) maxx / (double) maxy;
       redraw = 1;
      }
      break;

    case '1':
      purple_palette ();
      if (1 != palette) {
       redraw = 1;
       palette = 1;
      }
      break;

    case '2':
      blue_palette ();
      if (2 != palette) {
       redraw = 1;
       palette = 2;
      }
      break;

    case '3':
      amber_palette ();
      if (3 != palette) {
       redraw = 1;
       palette = 3;
      }
      break;
    case 'i':
    case '+':
      max_iter += 50;
      if (max_iter > PALETTE_SIZE)
       max_iter = PALETTE_SIZE;
      flag = redraw = 1;
      purple_palette ();
      blue_palette ();
      amber_palette ();
      break;

    case 'd':
    case '-':
      max_iter -= 50;
      if (max_iter < 50)
       max_iter = 50;
      flag = redraw = 1;
      purple_palette ();
      blue_palette ();
      amber_palette ();
      break;

    case KEY_LEFT:
      xm -= (xmax - xmin) / 20;
      redraw = 1;
      break;

    case KEY_RIGHT:
      xm += (xmax - xmin) / 20;
      redraw = 1;
      break;

    case KEY_UP:
      ym -= (ymax - ymin) / 20;
      redraw = 1;
      break;

    case KEY_DOWN:
      ym += (ymax - ymin) / 20;
      redraw = 1;
      break;

    default:
      redraw = 0;
    }

  } // while

  closegraph ();
  return 0;

} // main(void) ()

// ----- end of file mandelbrot.c
```

SDL2_image SDL2_Mixer

This is optional and for users that only want to make use of SDL2 directly without graphics.h (SDL-BGI)

Ensure that you are using a compatible version of SDL-Mixer and SDL-Image and that it is the development versions (-devel-).

https://github.com/libsdl-org/SDL_image

https://github.com/libsdl-org/SDL_mixer

SDL2_image-devel-2.6.2-mingw.zip

SDL2_image-devel-2.6.2-mingw.zip

SDL2_mixer-devel-2.6.2-mingw.zip

Runtime Libraries, Redistributable DLLs. To be exported with your project executable.

SDL2_image-2.6.2-win32-x64.zip

SDL2_mixer-2.6.2-win32-x64.zip

Copy files from ".\SDL2_image-2.6.2\x86_64-w64-mingw32\Include, lib, bin"

to

Include, lib, bin folders in your Dev-C++ directory.

Do the same for .\SDL2_mixer-2.6.2\x86_64-w64-mingw32\

Also place SDL2_image.dll and/or SDL2_mixer.dll in the project directory.

Add -lSDL2_image and -lSDL2_image to the project linker options as well as the previous SDL2 linker options..

[-lSDL2_image]

[-lSDL2_mixer]

Use the following #include:

NOTE! SDL_Image and SDL_Mixer do not work with #include <graphics.h>.

#include <SDL2/SDL.h>// lSDL2main -lSDL2

#include <SDL2/SDL_image.h>// -lSDL2_image

#include <SDL2/SDL_mixer.h>// -lSDL2_mixer

I don't currently have a test example for SDL_Image or SDL_Mixer.

Examples:
This is currently outside of the scope of this tutorial as it is SDL-BGI focused. You can find some examples to get started here: https://github.com/aminosbh/sdl2-mixer-sample https://gist.github.com/fschr/a8476f8993e5b9a60aa1c7ec4af3704b

SDL2 Linux Setup

NOTE! SDL-BGI apps will not run well under VirtualBox clients.

https://wiki.libsdl.org/SDL2/Installation

Please note the most recent version of SDL2 available will depend upon the Ubuntu version you are using. The following list the latest version of SDL2 available on Focal 20.04.

To install the SDL runtime library only use "**sudo apt-get install libsdl2-2.0-0"** as "sudo apt-get install libsdl2" does not find the library.

The following is the GCC build environment required to compile C/C++ source code. Running the following if already installed will check for a package update.

`sudo apt-get install build-essential` (If not already installed)

Install SDL2 development (-dev) library.

`sudo apt-get install libsdl2-dev` (This will install the below libsdl2-2.0-0 runtime as well)

Many additional dependencies will also be installed.

libsdl2-2.0-0 (SDL2 Runtime library)

The following is for the runtime library only. `sudo apt-get install libsdl2` (sudo apt-get install libsdl2-2.0-0)

Linking: -lSDL2 (Code::Block does not use the -l linker prefix)

Or see the setup guide for Code::Blocks setup section.

Link libraries: ~~-lSDL_bgi~~ -lSDL2main -lSDL2 Other linker options: `sdl2-config --libs`

Optional, not required. libsdl2-image-dev libsdl2-mixer-dev [-lSDL2_image] [-lSDL2_mixer]
#install sdl2 sudo apt install libsdl2-dev #install sdl image - if you want to display images `sudo apt install libjpeg-dev libwebp-dev libtiff5-dev libsdl2-image-dev`

#install sdl mixer - if you want sound

```
sudo apt install libmikmod-dev libfishsound1-dev libsmpeg-dev liboggz2-dev
libflac-dev libfluidsynth-dev libsdl2-mixer-dev
```

#install sdl true type fonts - if you want to use text

```
sudo apt install libfreetype6-dev libsdl2-ttf-dev
```

Use

`sdl2-config --cflags --libs` -lSDL2 -lSDL2_mixer -lSDL2_image -lSDL2_ttf

to link them, for example:

g++ myProgram.cpp -o myProgram `sdl2-config --cflags --libs` -lSDL2 -lSDL2_mixer -lSDL2_image -lSDL2_ttf

Examples:

This is currently outside of the scope of this tutorial as it is SDL-BGI focused.
You can find some examples to get started here:
https://github.com/aminosbh/sdl2-mixer-sample
https://gist.github.com/fschr/a8476f8993e5b9a60aa1c7ec4af3704b

Include file path:

/usr/include/SDL2/SDL.h

#include < SDL2/SDL.h>

You can try some of the test examples provided with the SDL2 source code.

https://github.com/libsdl-org/SDL

Or search SDL2 example github to find many repositories containing examples for a variety of tasks.

Example 1 “Basic SDL Hello world”

```
//Using SDL and standard IO

// -lmingw32 -lSDL2main -lSDL2

#include <SDL2/SDL.h>
#include <stdio.h>
#include <stdbool.h>

//Screen dimension constants
const int SCREEN_WIDTH = 640;
const int SCREEN_HEIGHT = 480;

int main( int argc, char* args[] )
    {
    //The window we'll be rendering to
    SDL_Window* window = NULL;
    //The surface contained by the window
    SDL_Surface* screenSurface = NULL;

    //Initialize SDL
    if( SDL_Init( SDL_INIT_VIDEO ) < 0 )
        {
        printf( "SDL could not initialize! SDL_Error: %s\n", SDL_GetError() );
        }
    else
        {
        //Create window
        window = SDL_CreateWindow( "SDL Tutorial", SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEIGHT, SDL_WINDOW_SHOWN );
        if( window == NULL )
            {
            printf( "Window could not be created! SDL_Error: %s\n", SDL_GetError()
);
            }
        else
            {
            //Get window surface
            screenSurface = SDL_GetWindowSurface( window );

            //Fill the surface white
            SDL_FillRect( screenSurface, NULL, SDL_MapRGB( screenSurface->format,
0xFF, 0xFF, 0xFF ) );

            //Update the surface
            SDL_UpdateWindowSurface( window );

            //Hack to get window to stay up
            SDL_Event e;
            bool quit = false;
            while( quit == false )
                {
                while( SDL_PollEvent( &e ) )
                    {
                    if( e.type == SDL_QUIT ) quit = true;
                    }
                }
            }
        }
    //Destroy window
    SDL_DestroyWindow( window );

    //Quit SDL subsystems
    SDL_Quit();

    return 0;
    }
```

SDL-BGI setup

https://sdl-bgi.sourceforge.io/

https://sourceforge.net/projects/sdl-bgi/

Use your web browser and download the following package.

https://sourceforge.net/projects/sdl-bgi/files/sdl_bgi_3.0.0-1_amd64.deb

Use the package install manager.

Double click in Lubuntu will use the Discover package manager by default.

Or right click on the file and choose a package manager for the install. QApt, Discover.

Install using the terminal:

Navigate to the download location (or place the full file path in front of the package /home/user1/Downloads/sdl_bgi_3.0.0-1_amd64.deb). Run the following dpkg command to install the .deb package.

```
sudo dpkg -i sdl_bgi_3.0.0-1_amd64.deb
```

`sudo apt-get install -f` (Attempt to fix any missing dependencies)

Installed path: /usr/include/SDL2/SDL_bgi.h

#include < SDL2/SDL_bgi.h>

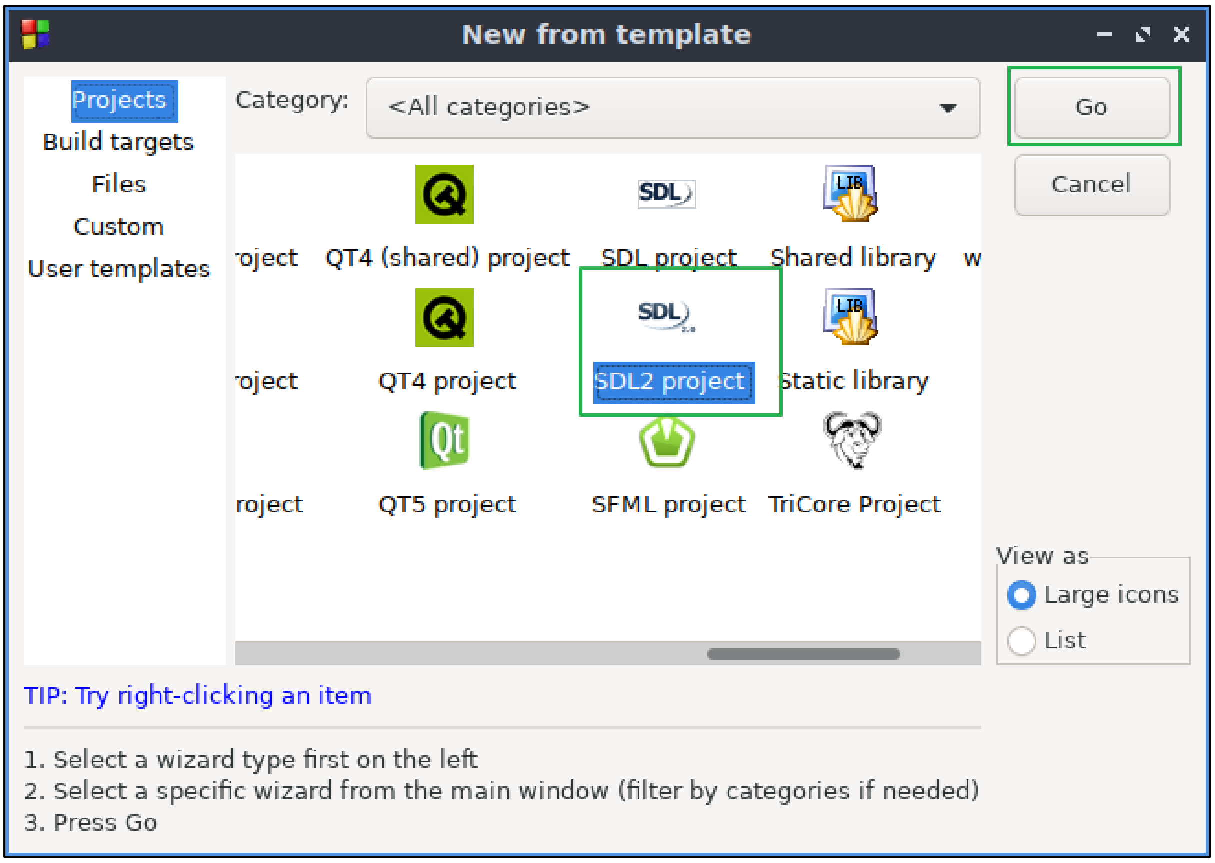
See the Code::Block setup section for how to set up an SDL2 and SDL-BGI project.

Code::Blocks setup

Start CodeBlocks and then click on Create a new project.

Select SDL2 project and Go.

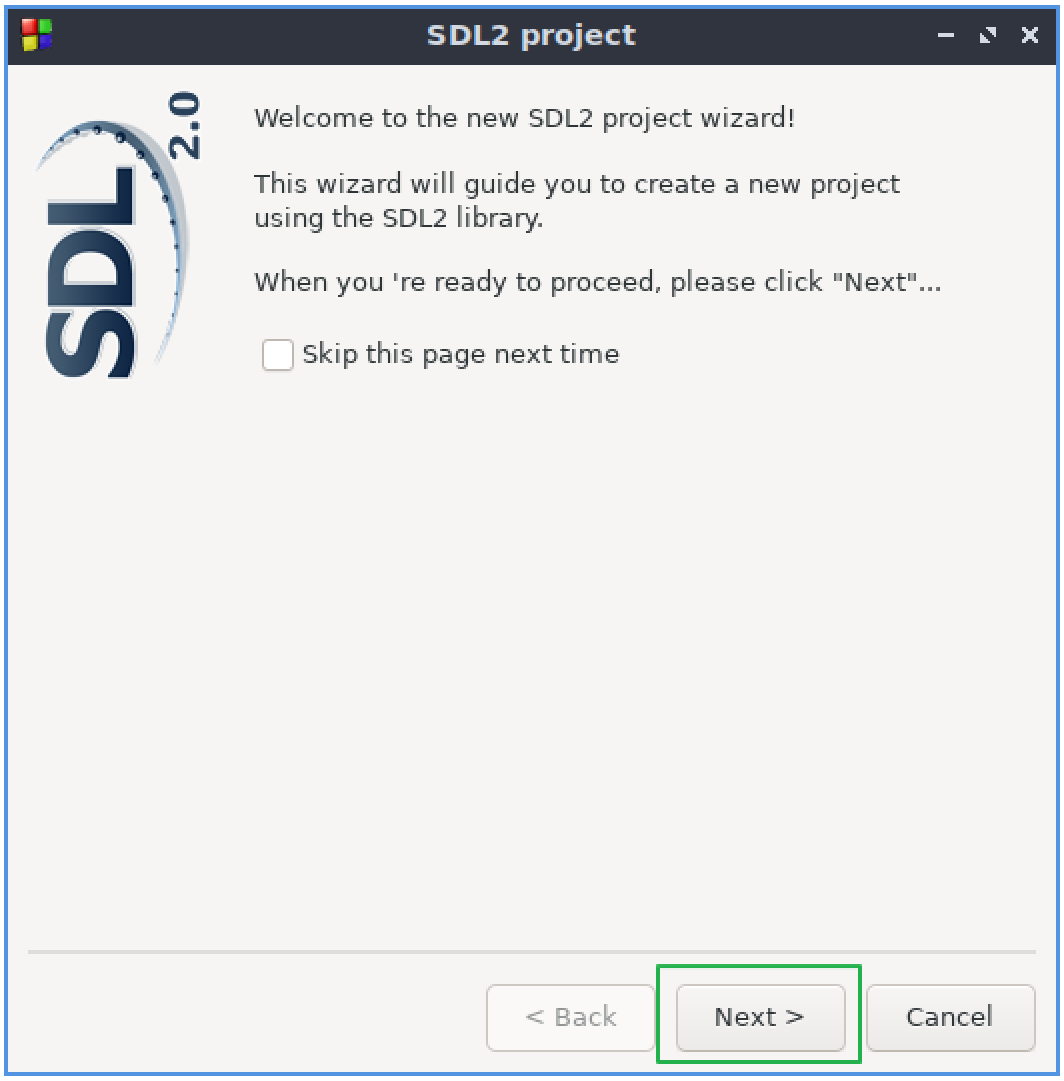
Note! It is possible to start a blank project and manually fill in the SDL2 and SDL-BGI options.



TIP: Try right-clicking an item

1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Click next on the SDL2 project wizard screen.

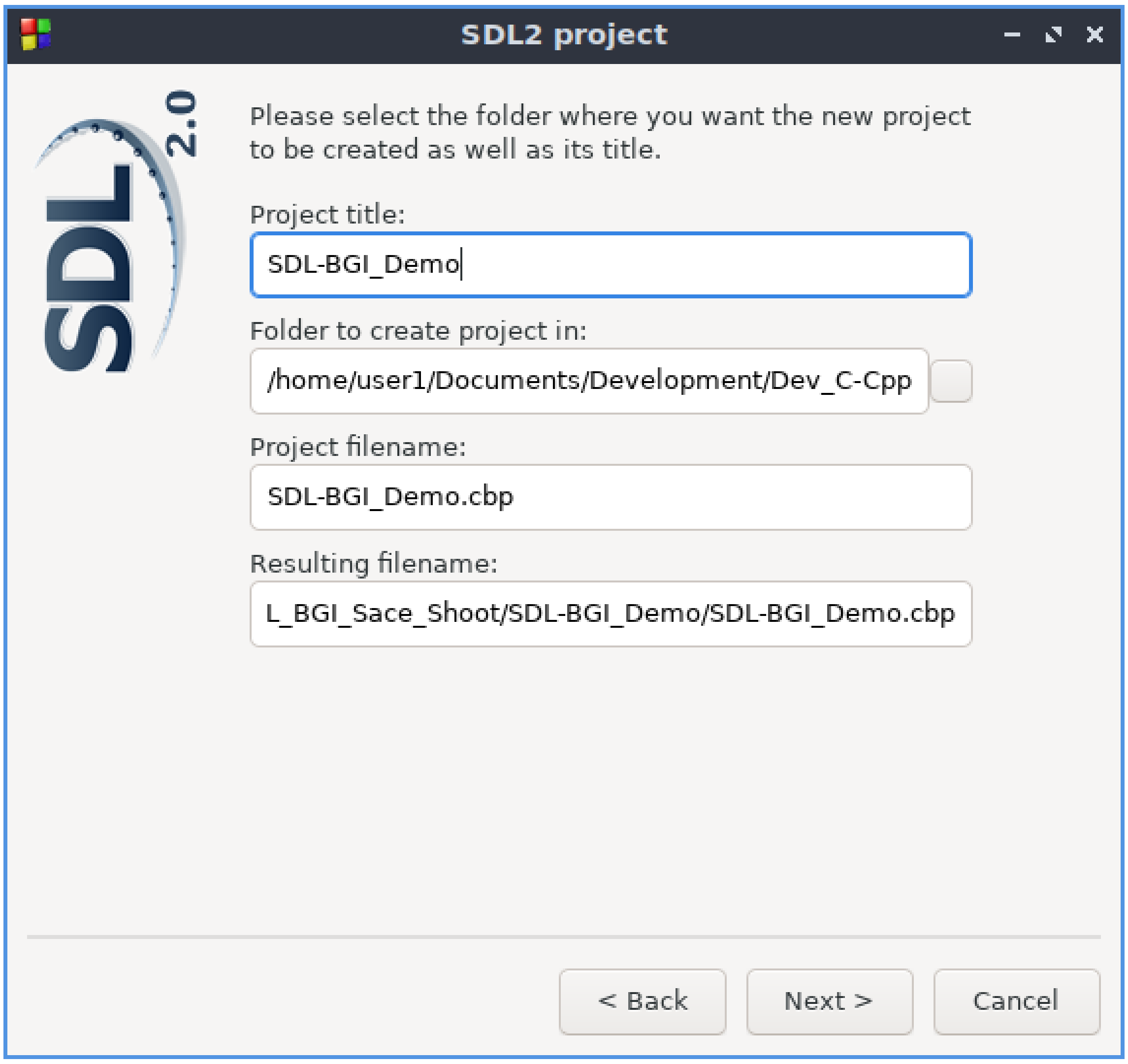


Welcome to the new SDL2 project wizard!

This wizard will guide you to create a new project using the SDL2 library.

When you 're ready to proceed, please click "Next"...

Fill out the project name and path details.


SDL2 project

Please select the folder where you want the new project to be created as well as its title.

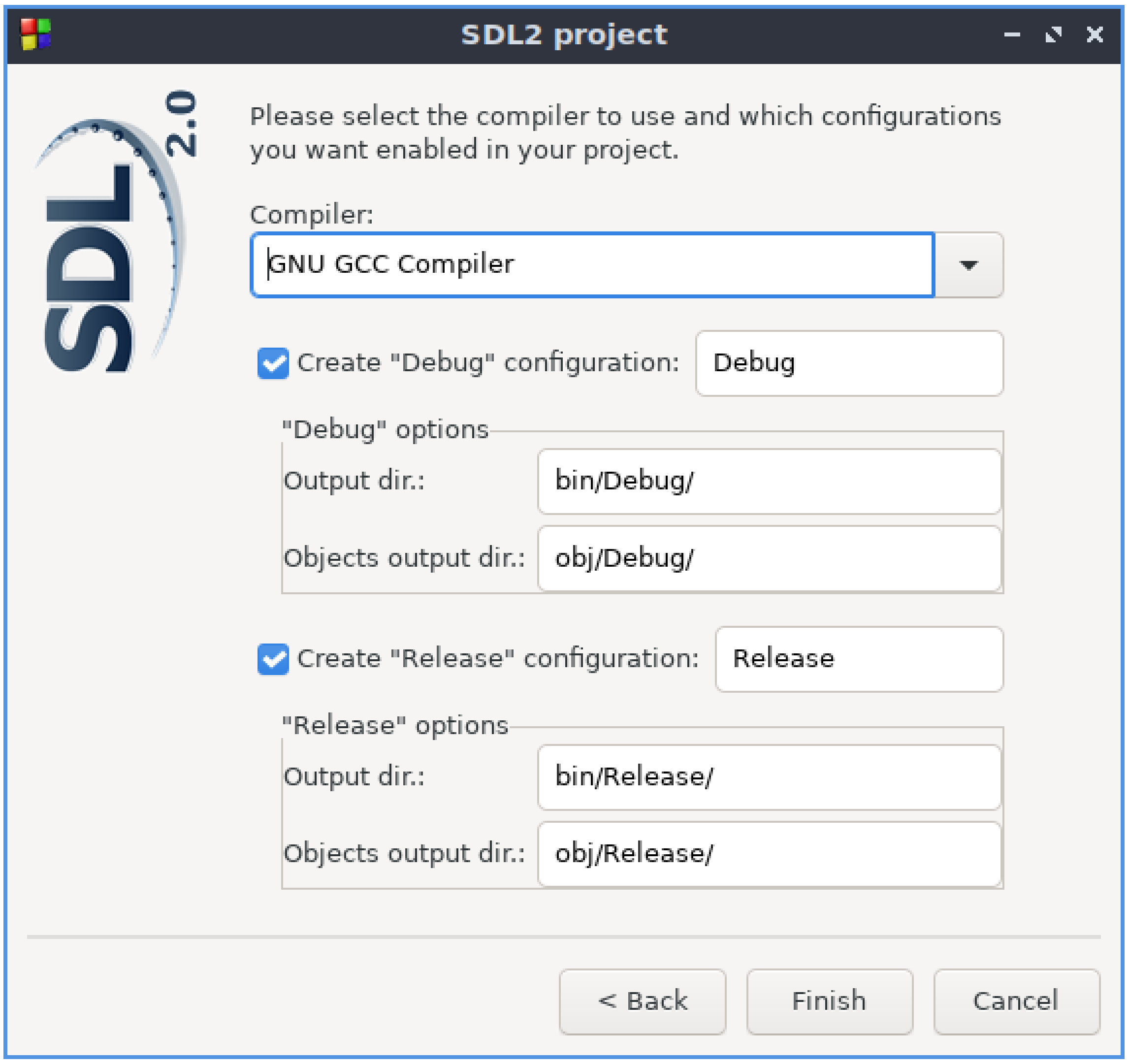
Project title:
SDL-BGI_Demo

Folder to create project in:
/home/user1/Documents/Development/Dev_C-Cpp

Project filename:
SDL-BGI_Demo.cbp

Resulting filename:
L_BGI_Sace_Shoot/SDL-BGI_Demo/SDL-BGI_Demo.cbp

Continue with the default compiler output options.


SDL2 project

Please select the compiler to use and which configurations you want enabled in your project.

Compiler: GNU GCC Compiler

[✓] Create "Debug" configuration: Debug

"Debug" options

Output dir.: bin/Debug/

Objects output dir.: obj/Debug/

[✓] Create "Release" configuration: Release

"Release" options

Output dir.: bin/Release/

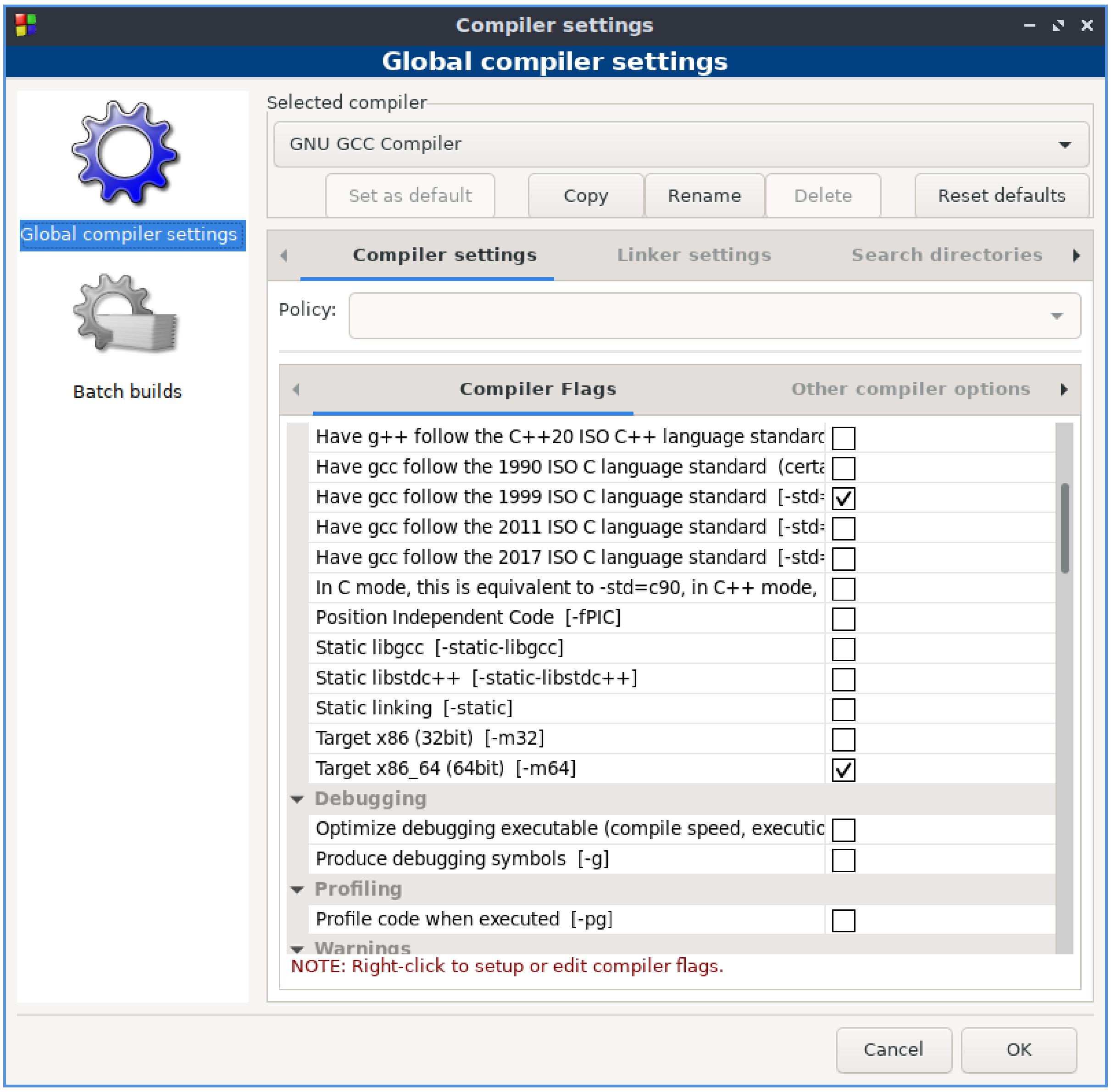
Objects output dir.: obj/Release/

Select Build and Run to test the default SDL2 install. If the colour screen comes up then all is good :)

Manual settings check plus SDL-BGI.

From the menu select “Settings” -> “Compiler…”

Check the following **Global settings** for C99 and x64 bit compile.



Compiler settings

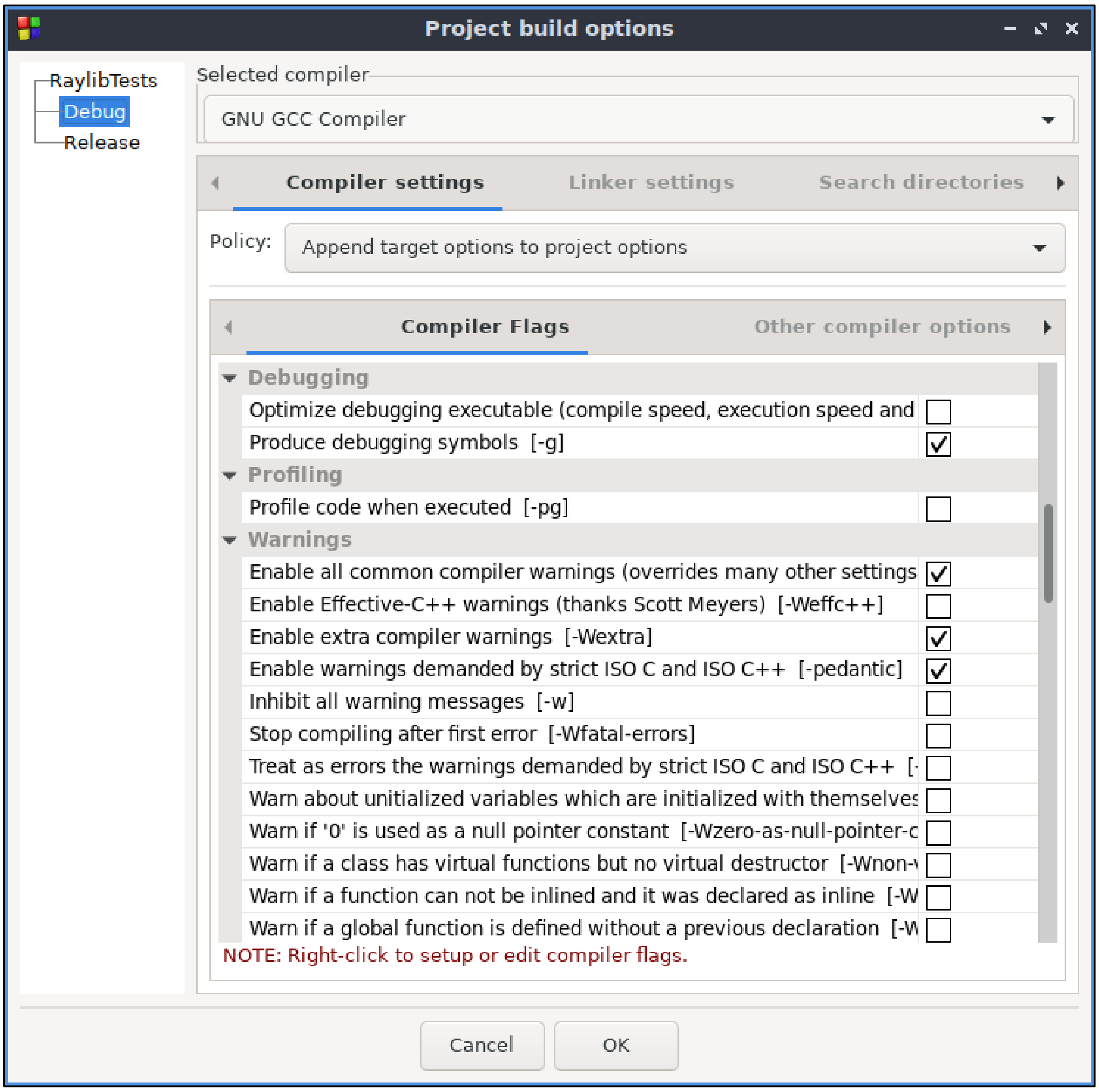
Global compiler settings

Selected compiler: GNU GCC Compiler

Policy:

Have g++ follow the C++20 ISO C++ language standarc	☐
Have gcc follow the 1990 ISO C language standard (certa	☐
Have gcc follow the 1999 ISO C language standard [-std=	✓
Have gcc follow the 2011 ISO C language standard [-std=	☐
Have gcc follow the 2017 ISO C language standard [-std=	☐
In C mode, this is equivalent to -std=c90, in C++ mode,	☐
Position Independent Code [-fPIC]	☐
Static libgcc [-static-libgcc]	☐
Static libstdc++ [-static-libstdc++]	☐
Static linking [-static]	☐
Target x86 (32bit) [-m32]	☐
Target x86_64 (64bit) [-m64]	✓
Debugging	
Optimize debugging executable (compile speed, executic	☐
Produce debugging symbols [-g]	☐
Profiling	
Profile code when executed [-pg]	☐
Warnings	

NOTE: Right-click to setup or edit compiler flags.



Debugging
- [ ] Optimize debugging executable (compile speed, execution speed and
- [✓] Produce debugging symbols [-g]

Profiling
- [ ] Profile code when executed [-pg]

Warnings
- [✓] Enable all common compiler warnings (overrides many other settings
- [ ] Enable Effective-C++ warnings (thanks Scott Meyers) [-Weffc++]
- [✓] Enable extra compiler warnings [-Wextra]
- [✓] Enable warnings demanded by strict ISO C and ISO C++ [-pedantic]
- [ ] Inhibit all warning messages [-w]
- [ ] Stop compiling after first error [-Wfatal-errors]
- [ ] Treat as errors the warnings demanded by strict ISO C and ISO C++ [-
- [ ] Warn about unitialized variables which are initialized with themselves
- [ ] Warn if '0' is used as a null pointer constant [-Wzero-as-null-pointer-c
- [ ] Warn if a class has virtual functions but no virtual destructor [-Wnon-
- [ ] Warn if a function can not be inlined and it was declared as inline [-W
- [ ] Warn if a global function is defined without a previous declaration [-W

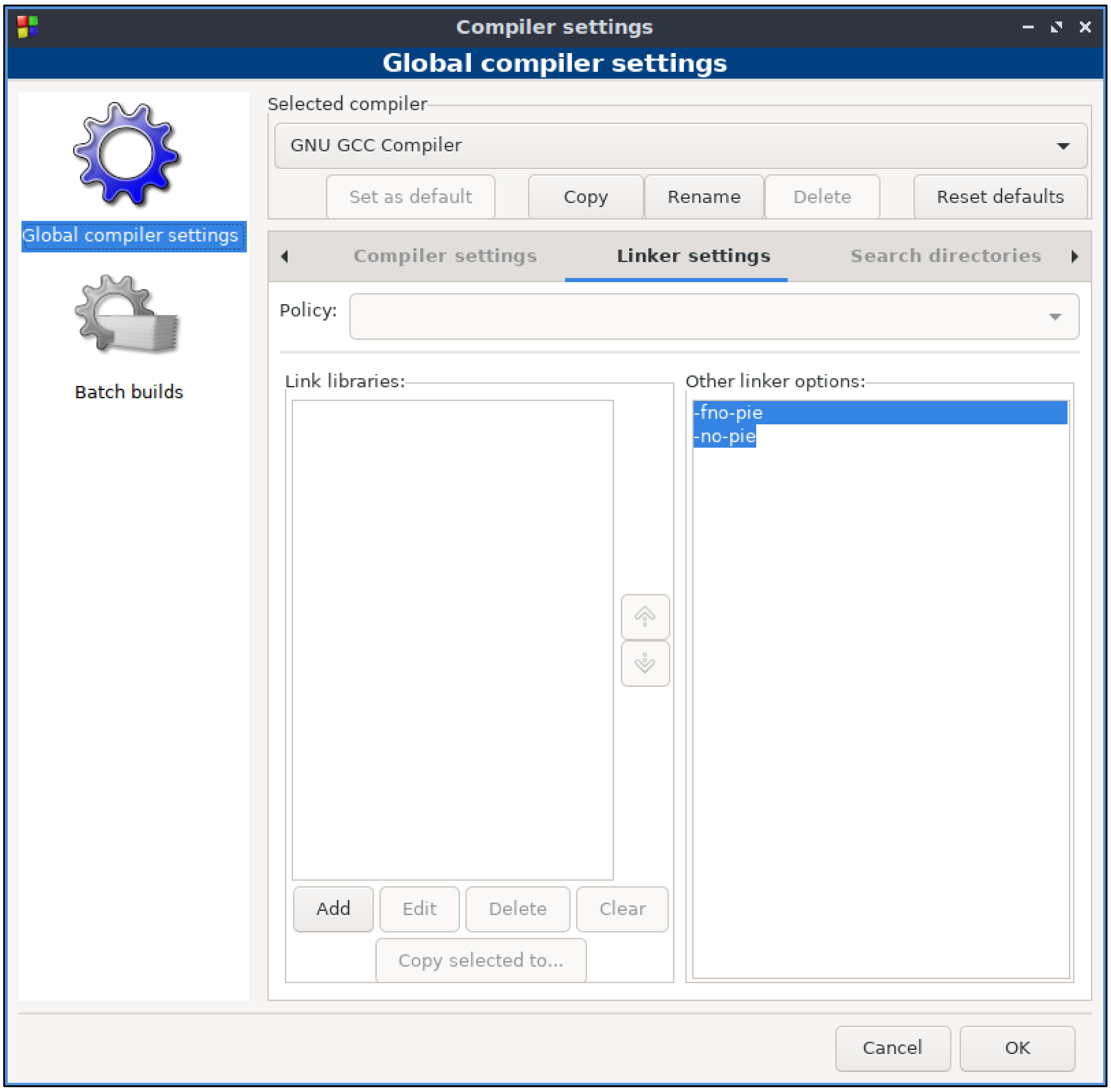
NOTE: Right-click to setup or edit compiler flags.

Select the “Linker setting” Tab and check that “Other linker options” contains the following 2 lines.

-fno-pie

-no-pie

Write each setting in line by line. This will create an executable ELF file.



```
-fno-pie
-no-pie
```

CORECT THESE SETTINGS INFO!!!

Next open “Project” -> “Build Options...”

Make sure to select the Global project option on the top left.

Select the “Linker setting” Tab and add the following to “Link libraries:”

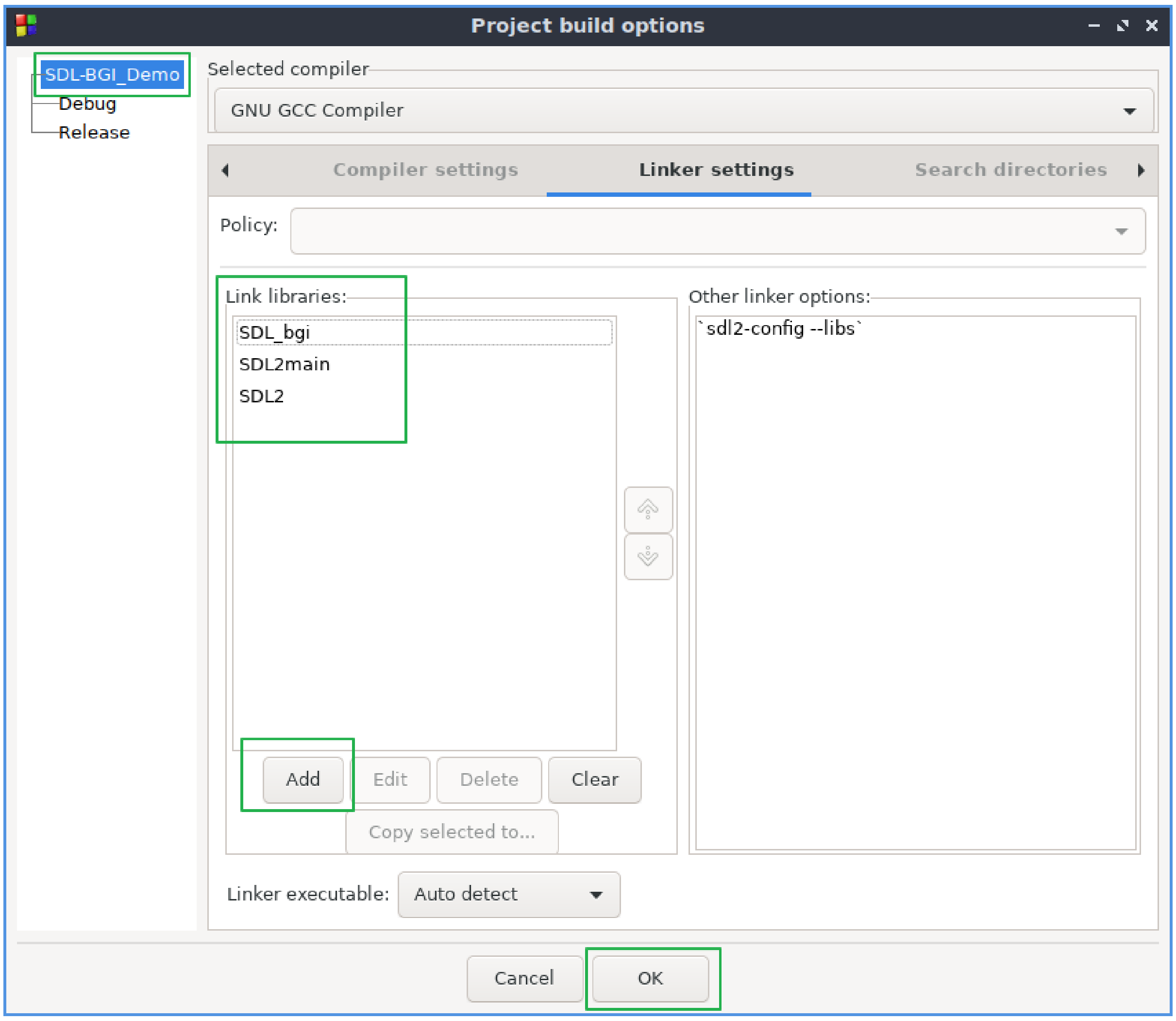
SDL_bgi

SDL2main

SDL2

Under “Other linker options” add the following line.

`` `sdl2-config --libs` ``



Link libraries:

SDL_bgi
SDL2main
SDL2

Other linker options:

`sdl2-config --libs`

Linker executable: Auto detect

You can click [OK] to save and close the panel.

At this point the Code::Blocks project should be set up and you can compile a basic example.

Remove the default main.cpp (SDL2 example) file from the project, and replace it with new empty file and save it as main.c

Add a test example to the new page to test SDL-BGI with.

Example: “SDL_bgi-3.0.0\test\arc.c”

```
/* arc example */
/* SDL_bgi-3.0.0\test\arc.c */

#include <graphics.h>

int main(int argc, char *argv[])
{
  /* request autodetection */
  int gdriver = DETECT, gmode;
  int midx, midy;
  int stangle = 45, endangle = 135;
  int radius = 100;

  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;
  setcolor(getmaxcolor());

  /* draw arc */
  arc(midx, midy, stangle, endangle, radius);

  /* clean up */
  getch();
  closegraph();
  return 0;
}
```

Demo examples can be found in the source code directory under:

SDL_bgi-3.0.0-win\SDL_bgi-3.0.0\demo

SDL_bgi-3.0.0-win\SDL_bgi-3.0.0\test

Some of the demo files may not work as expected and will need slight modification. Many of the examples are targeted toward windows or DOS under the Borland turbo C. You will need to check what function is being called from the missing header and use a header and function that is equivalent under Linux. Sometimes it is only required to remove a header from the example to get the correct function. For example in the following the #include <conio.h> is not required for the getch(); function as it is already declared in graphics.h.

Example: “SDL_bgi-3.0.0\test\pieslice.c”

```
/* pieslice example */
/* SDL_bgi-3.0.0\test\pieslice.c */

#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h> // Comment this line out!

int main(int argc, char *argv[])
{
  /* request autodetection */
  int gdriver = DETECT, gmode;
  int midx, midy;
  int stangle = 45, endangle = 135, radius = 100;

  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "C:\\TC\\BGI");

  midx = getmaxx() / 2;
  midy = getmaxy() / 2;

  /* set fill style and draw a pie slice */
  setfillstyle(EMPTY_FILL, getmaxcolor());
  pieslice(midx, midy, stangle, endangle, radius);
  /* clean up */
  getch();
  closegraph();
  return 0;
}
```

In the above example the 2 libraries:

```
#include <stdlib.h>
#include <stdio.h>
```

Have not been used in the example and can also be commented out, although in most cases the 2 STD library headers will be used in any substantial code.

NOTE!

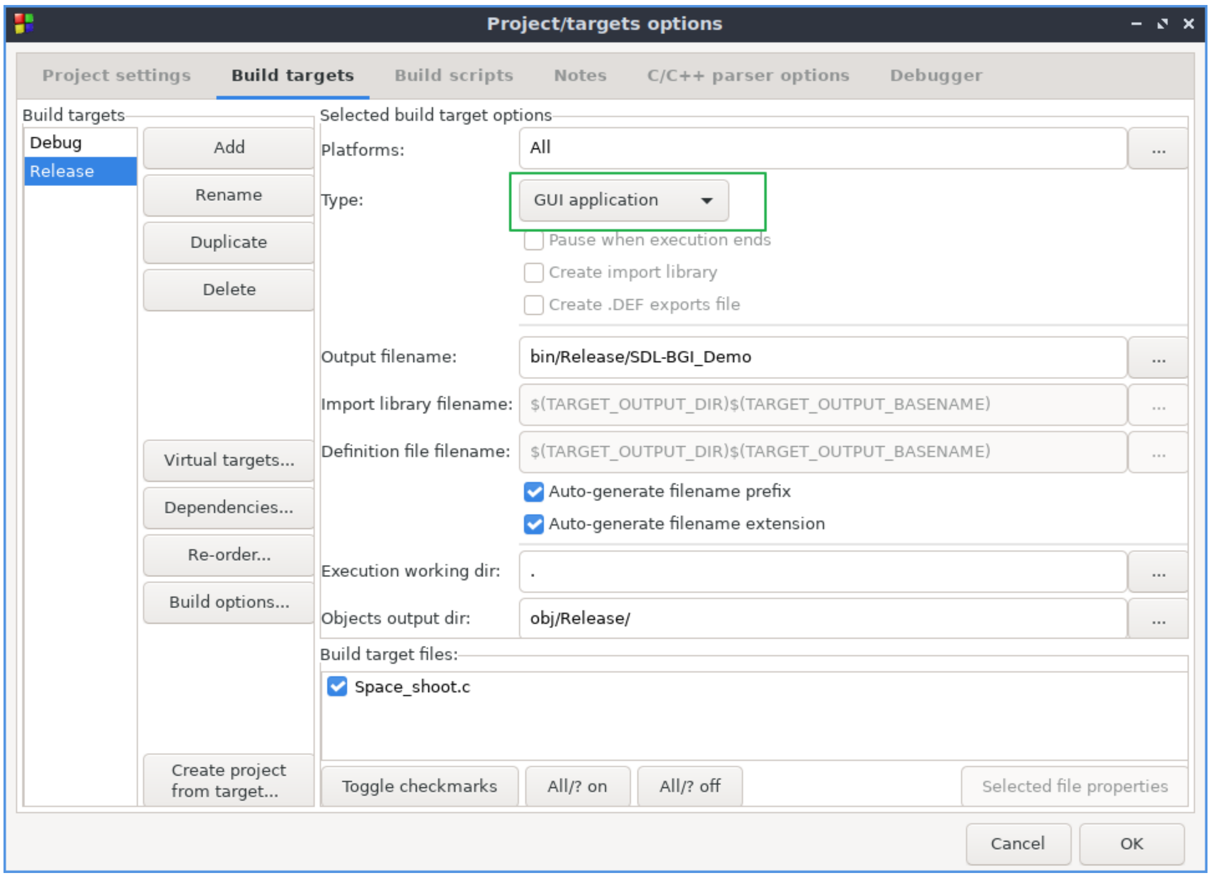
By default this is set to be run from the command console for debugging purposes.

When wanting a standalone graphical window ensure all calls to the console are removed.

Open “Project” -> Properties...”

Switch to “Build targets” tab and change “Console application” to “GUI application”.

Can also use “Native application”.



Platforms: All

Type: GUI application

- [ ] Pause when execution ends
- [ ] Create import library
- [ ] Create .DEF exports file

Output filename: bin/Release/SDL-BGI_Demo

Import library filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

Definition file filename: $(TARGET_OUTPUT_DIR)$(TARGET_OUTPUT_BASENAME)

- [x] Auto-generate filename prefix
- [x] Auto-generate filename extension

Execution working dir: .

Objects output dir: obj/Release/

Build target files:

- [x] Space_shoot.c

Also note that as a standalone application the “assets” folder will need to be in the same directory as the compiled executable.

/bin/Debug/SDL-BGI_Demo

/bin/Debug/assets/*.*

When exporting compiled applications to another Linux computer you will need to offer instructions to install the required runtime libraries to use your application (SDL2 and SDL-BGI). Alternatively you can create an installer that adds the libraries as a dependency when instating.

I am working on instructions for how to create basic application installers for Windows and Linux in another tutorial.