Introduction to Game Programming

Session 1 – Introduction to Tao SDL

Nacho Cabanes
Nacho Iborra

IES San Vicente

Index of contents

1. Introduction

In this session we are going to introduce the library that we are going to use in next sessions to implement a graphical video game: Tao SDL.

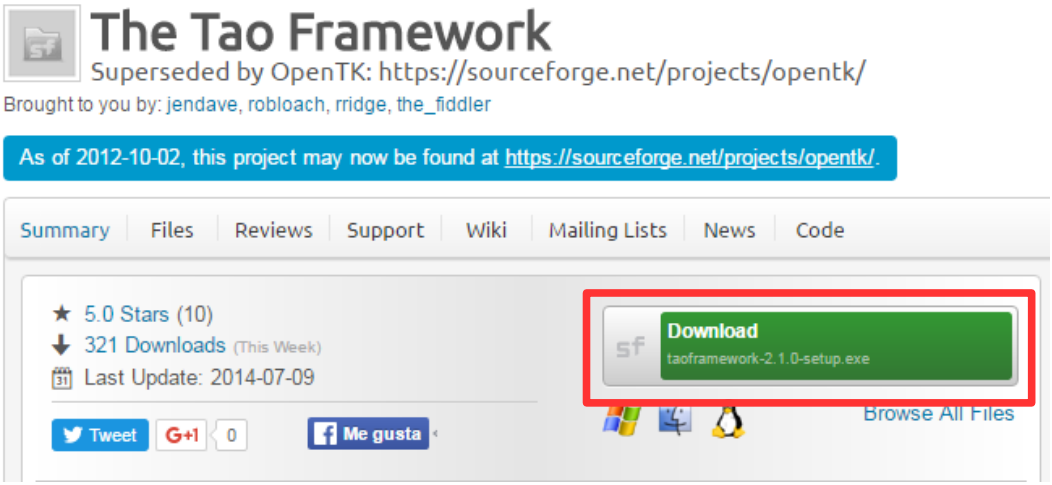
Tao SDL is an adaptation of SDL, a graphical library to create video games in several platforms. It allows drawing images on the screen, getting input from different peripherals (keyboard, mouse, gamepad), and some other features such as sounds or fonts. It works in C++, but there are some extensions to write games in other languages, such as Python or C#. In these sessions we will see the basic steps to use it in this last language.

At the beginning, Tao SDL was part of the Mono project (the free compiler for C#), but it was abandoned, and in last years it has joint a SourceForge project called *Tao framework*. Its official web site is:

http://sourceforge.net/projects/taoframework/

1.1. Installing Tao SDL

To install Tao SDL in our system, we just need to download and launch the installer from the official web site. Although you will find distributions for Windows, Linux and Mac, the easiest way to make it work is under Windows.

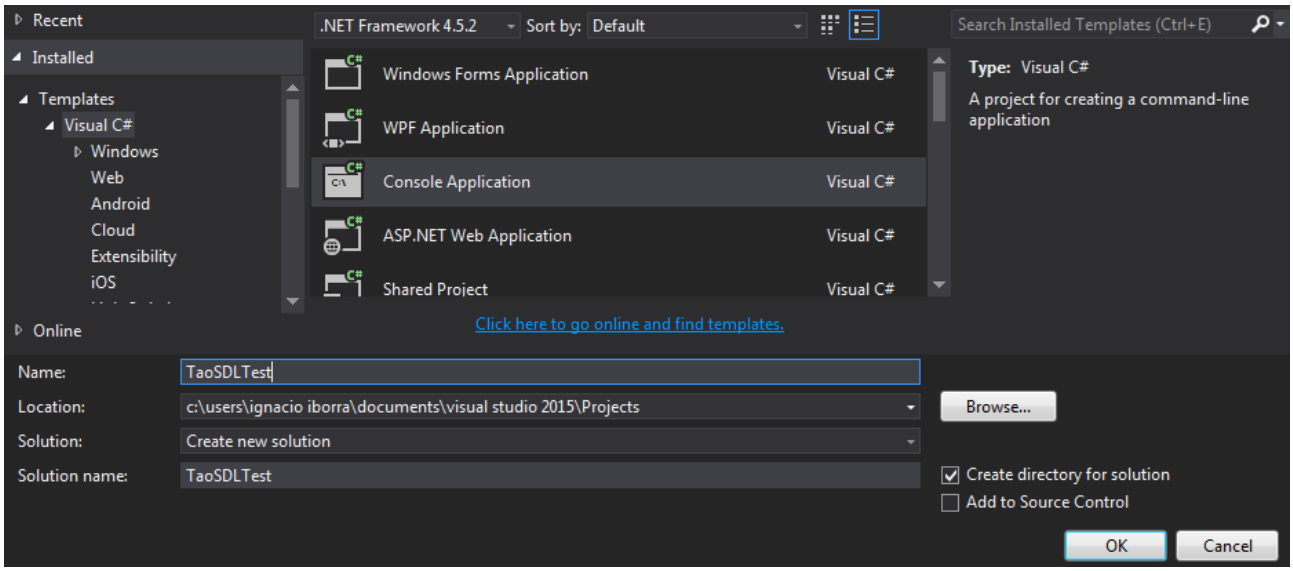


When the wizard launches, we just leave the default options and click on *Next* if needed. A folder called "*C:\Program Files (x86)\TaoFramework*" will be created. Remember the path of this folder, because we will need to get some files from it.

2. Configuring a project

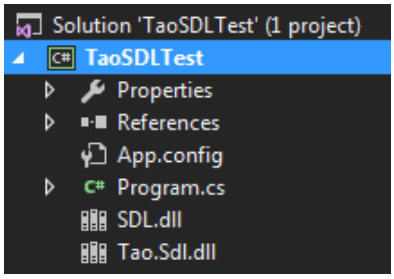
Once we have installed Tao framework in our system, we can create a project with our IDE (Visual Studio, SharpDevelop or Xamarin), and load the libraries to create the video game.

First of all, create a *Console Application* project with the desired name:



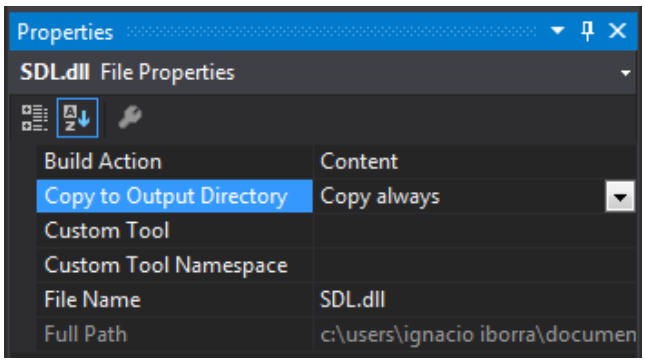
Then, we need to go to the Tao framework installation folder (see subsection 1.1 above) and copy these files into the source folder of our project (without any subfolder):

- *lib/SDL.dll*
- *bin/Tao.Sdl.dll*



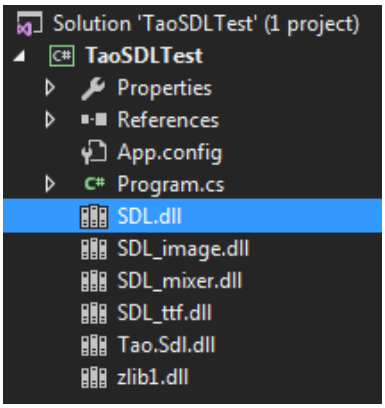
IMPORTANT: do NOT just copy and paste the DLL files inside the project folder from Windows explorer. Do it in Visual Studio, by pasting the copied files in the solution panel on the right. Otherwise, Visual Studio will not consider these files as part of the project.

Besides, we need to change the properties of these files and set that they must be copied to the output folder always, so that they are available when running the game.

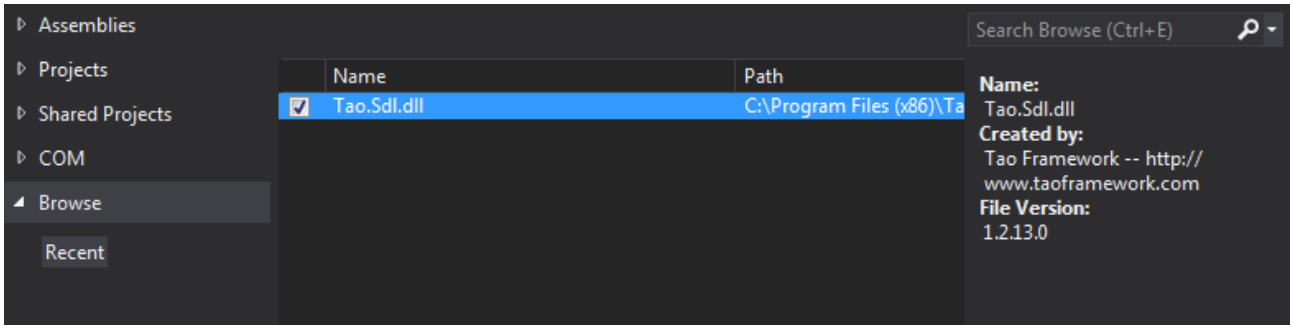


We may also need some other libraries for our video game, so we can do the same with the corresponding files (copy them to the source folder of our project, and set their properties to be always copied to the output folder). These libraries (available in *lib* subfolder) are:

- *SDL_image.dll* to deal with images (for instance, backgrounds or sprites)
- *SDL_ttf.dll* and *libfreetype-6.dll* to show texts with TTF fonts
- *SDL_mixer.dll* to add audios and sound effects to our game
- *zlib1.dll*, an auxiliary library that is often used by other libraries



Finally, in the project references, we need to add *Tao.Sdl.dll* library to use its classes in our application. To do this, we right click on the *References* section (in the right panel in Visual Studio), and choose *Add reference*. Then, we look for the DLL with the *Browse* button, and check it.



Exercise 1

Create a project called **TaoSDLTest** and follow all the previous steps to add the Tao libraries on it, and leave it ready to use. We will use this project in next sessions to test some code before adapting it to our real video game.

3. The Hardware class

To finish with this introductory session, we are going to create a class called **Hardware** inside our project. This class is going to manage everything about hardware communication: screen configuration, key inputs and so on. Within this class, we are going to define our own methods to deal with hardware issues, and these methods are going to use Tao SDL methods, that are (or may be) more difficult to remember or understand.

As we are going to use Tao SDL classes and elements, we need to use the *Tao.Sdl* library, so our class structure will be like this:

```
using System;
using Tao.Sdl;

namespace TaoSDLTest
{
	class Hardware
	{
	}
}
```

The attributes of this class are:

- The screen width and height
- The color depth (in bits)
- An attribute of type *IntPtr* (from *System* library) that will manage the screen. We will see more properties of this type of object in next sessions.

```
using System;
using Tao.Sdl;

namespace TaoSDLTest
{
	class Hardware
	{
		short screenWidth;
		short screenHeight;
		short colorDepth;
		IntPtr screen;
	}
}
```

Note that screen width and height are *short* variables, although we could consider defining them as *ushort*, or *int*. But these types would produce a compilation error, since the Tao SDL methods that we are going to use need these attributes to be *short*.

In the constructor of this class, we assign a value to every attribute, and configure the *screen* attribute according to the desired width and height. Besides, we are going to use an additional parameter to set if we want to play the game in full screen mode or not:

```
using System;
using Tao.Sdl;

namespace TaoSDLTest
{
      class Hardware
      {
            short screenWidth;
            short screenHeight;
            short colorDepth;
            IntPtr screen;

            public Hardware(short width, short height, short depth,
                        bool fullScreen)
            {
                  screenWidth = width;
                  screenHeight = height;
                  colorDepth = depth;

                  int flags = Sdl.SDL_HWSURFACE | Sdl.SDL_DOUBLEBUF |
                              Sdl.SDL_ANYFORMAT;
                  if (fullScreen)
                        flags = flags | Sdl.SDL_FULLSCREEN;

                  Sdl.SDL_Init(Sdl.SDL_INIT_EVERYTHING);
                  screen = Sdl.SDL_SetVideoMode(screenWidth, screenHeight,
                                                colorDepth, flags);
                  Sdl.SDL_Rect rect = new Sdl.SDL_Rect(0, 0, screenWidth,
                                                       screenHeight);
                  Sdl.SDL_SetClipRect(screen, ref rect);
            }
      }
}
```

The first lines just assign a value to each attribute (*screenWidth, screenHeight* and *colorDepth*). Then, we set some boolean flags to configure the screen, and then we init the *screen* attribute according to these flags. Finally, we will define a rectangle to represent the screen.

We can also define a destructor to remove everything once the game finishes.

```
using System;
```

```
using Tao.Sdl;

namespace TaoSDLTest
{
      class Hardware
      {
            ...
            ~Hardware()
            {
                  Sdl.SDL_Quit();
            }
      }
}
```

Exercise 2

Add this class with its attributes, constructor and destructor to projects *TaoSDLTest* from previous exercise.

4. Some last changes

4.1. Our first test

To test this class and see how the screen is shown, we can add these lines to our *Main* method:

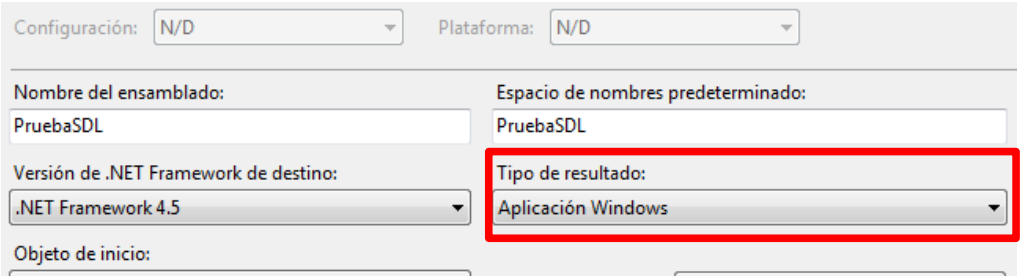
```
static void Main(string[] args)
{
      Hardware h = new Hardware(800, 600, 24, false);
      System.Threading.Thread.Sleep(5000);
}
```

The last line (*Thread.Sleep* method) will not be necessary in the future, but now we need it to show the screen for some seconds. Otherwise, it will be closed too quickly. We can also try to change the last parameter of the constructor to *true*, and see if the game is shown in full screen mode.

4.2. Changing the application type

Now that we know that our game is shown, we can change the application type. Remember that we defined the project as a *Console Application* project. With this type of project, a console will always be shown along with the game screen, and we will need to close both windows when we stop playing.

To avoid this, we can change the application type to a *Windows application*. If we had defined the project as a *Windows application* project at the beginning, we would have been forced to use a default window form component in the project. But now, we can change the project type without having to use this component. To do this, we right click on the project's name, and go to the *Properties* option. Then, change the *Result type* or *Output type* option (depending on your version of Visual Studio) to *Windows application*:



Exercise 3

Add the code above to the *Main* method of *TaoSDLTest* project, and test it. Once you make sure that it works properly, change the application type of the project to a *Window application*.

Exercise 4

Repeat all these steps in a new project called **Gauntlet.** Remember to copy all the required DLL files properly, add the *Hardware class* and test it in the *Main* method.