Game Programming 101 Part I
by Bruno Sousa

Introduction

Here it is! My first tutorial to be officially published by GameDev.net (yes, I'm very proud). I wrote some other small tutorials for game programming beginners about several topics, but instead I decided to group them all and make a series of tutorials from the very beginning of a game to the publishing phase. I doubt with the games you'll learn to develop here you'll get rich, but with a little imagination and hard work maybe you can even get some of them in bundle packs (and I don't even ask for your money J).

For those who don't know me, my name is Bruno Sousa (or Akura on the Internet) and you can probably find me in #Gamedev chat.

I first thought of doing this series by starting with a simple game (Pong) and developing it with your help, but I decided to make this series infinite; that is, every time I get some free time I'll write the next tutorial. Since I'm still in school and trying to get into the industry my schedule isn't that good, but I'll try to do my best. As long as I have time (and GameDev.net doesn't mind by ramblings) I'll keep writing these tutorials.

The only thing required for this series is that you have a basic knowledge of C and C++. The reason why I just don't use one or the other is I like to merge them, since parts are easier using C and others are easier using C++. If you don't know one of these languages but know the other you'll have no problem grasping these tutorials, but even so, if you're stuck mail me at magick_pt@geocities.com and I'll do my best to help you out.

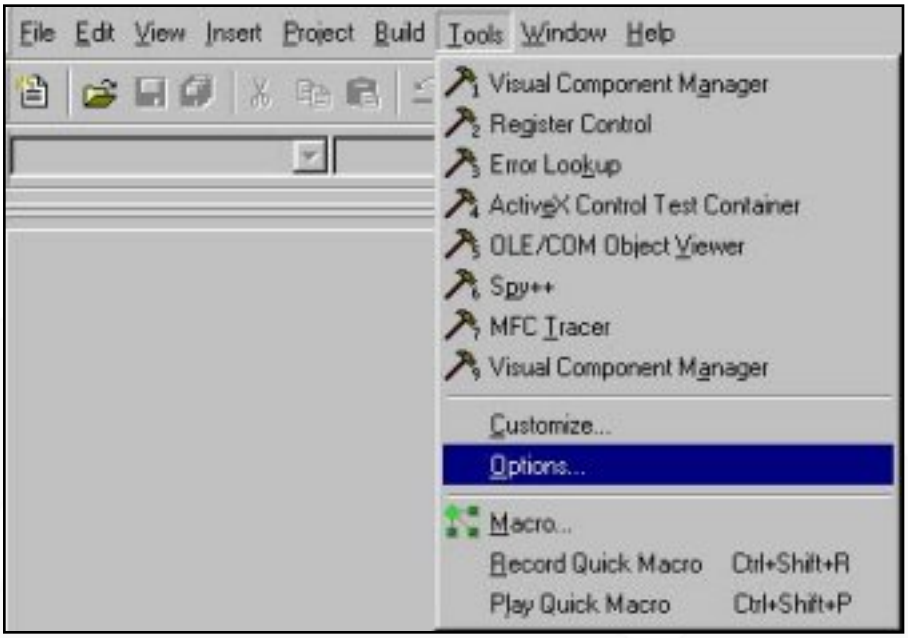
Also, I'm not the master of it all nor do I intend to be (for the time being). I'm as human as you so I can also make mistakes. If anyone notices an error or bug please tell me.

With all that said, let's start.

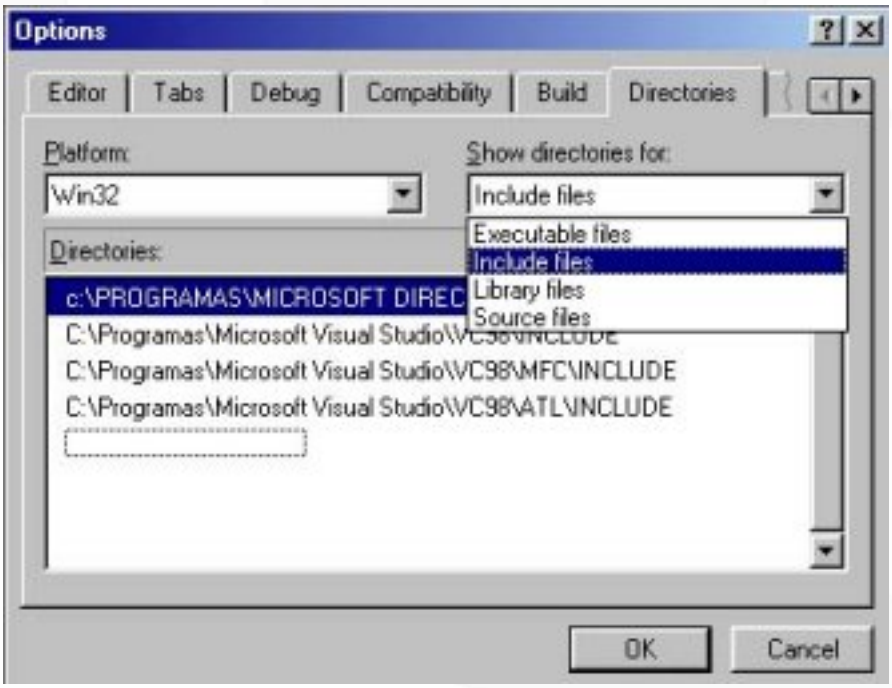
The very beginning

As you well know by now, any game developer that wants to have a future should know at least the basics of DirectX. Sure there are other options like OpenGL for graphics or the new OpenAL for audio, but at the present moment the industry standard is DirectX and so, that is what we are going to use. You'll need a compiler that is capable of generating Win32 executables (this series will feature Visual C++ 6.0, but you can use

any other compiler) and you will also need the DirectX SDK (we'll use version 7.0 since it is the most current version of it). You can download or order the DirectX SDK at the Microsoft DirectX SDK homepage.



Also I presume that you already know how to compile a normal Win32 program with your compiler and use external libraries. Since DirectX need a little bit of preparation before use let me describe the process if you're using Visual C++. First go to the Tool menu, then Options, and choose the tab Directories. In the *Show directories for* combo box choose Include files and add a new directory. Specify the directory where you installed the DirectX SDK and the include directory (if you have installed the SDK to `C:\DXSDK` the path should be `C:\DXSDK\include`). After adding the directory move it to the top of the list so when the compiler searches for the DirectX files it will use those in the directory you installed to and not the one that came with the compiler (which contains older DirectX headers). Then select Library files in the *Show directories for* combo box and do the same thing as above but replacing `\include` with `\lib`. You are now set to use the DirectX SDK. You will still need to manually add the libraries to your project but we'll take care of that in the next section.



The first game we'll develop will be a Pong clone. You can take many approaches to

develop a game, the one that will be presented here is the one I like, but if you prefer working in another manner (like first code the wrappers and then design, etc) it's your call. I stick with this one because I got used to it during my apprentice time.

The approach I take to program any game is given below:

- Design the complete game
- Develop all the wrappers, engine, tools needed for the game
- Start the development process with test beds
- Merge of all test beds to create a complete game
- Test and correct bugs
- Polish off
- Make lots of publicity about it

These are the steps I prefer to use to develop my games; if you have your own, fine stick with it.

Also a note: the term test bed usually refers to using a pencil and paper, toys, and/or programs to test an idea. I use the term test bed as part of a feature or engine, like text output, scrolling functions, and the save/load mechanism. I prefer to develop these features in a of separate program and then integrate it with the game. This system is also known as the tier system, I call it test bed - the important thing is that even though names differ the concepts are the same.

The first design

When I started to develop games I never designed. It was boring and I didn't like it (and still don't like to do it) and since I knew the game in my head I would have no problem programming it. Well, I was completely wrong. Designing a game will make us think of all the irrelevant aspects of the game that we only remember after doing those 500 lines of code to realise you need to change the structures and with that, the last weekend's work is going to the Recycle Bin. Even though Pong is a relatively a simple game, we will still make a small design document just to have the basic game worked out on paper, not just in our heads. Again, if you have your own design templates you don't have to use mine. I like designing like this because it is more natural for me to think of a game like this. This model works well for a team of 1-2 programmers for a relatively simple project; for more a complicated game another type of design document is recommended since it will have to be divided in parts and all. check GameDev.net's design section for some tutorials how to make your own design documents.

Introduction

This is a small introduction to the game. It's a short description to give someone an idea of what the game is

Ping is a Pong clone for the Windows platform using DirectX for graphics and input. The player has control of a space ship where he enters a playfield to play against another player (human or computer controlled) and tries to make the ball pass the goal area.

Minimum specifications

You can probably neglect this part of the design document, but I put it here because it is an important point in a more complete design document

Pentium 166
16 Megabytes RAM
3 megabytes free of disk space
DirectX compatible video card
Windows 95, 98 or 2000

Rules of the game

This section generally is featured in a Gameplay section of the design document. For such a small game it can be a separate section of its own

The rules for Ping are very easy and straightforward. The objective of the game is to score a goal, and to do this, the player needs to make the ball pass the goal line of the adversary while trying not to let the ball pass its own. Each spaceship can only move in the vertical direction within the bounds of the playfield.

Menus

This section should be divided into subsections for the sake of comprehension. Again, for such a simple game, it can all be described in one or two paragraphs

The main menu will have 4 options: New game, High score, Credits, and Quit. An image of a Ping game is shown in the background.

- The New game option takes the game to another menu where the player chooses the difficulty level and a single player or two player game of Ping. After the selection the player is taken to the actual game.
- The High score option will show the high score table.
- The Credits option will show a scrolling credits text with all the credits for the game.
- The Quit option will ask the player if he or she really wants to quit and based on the response it will quit or not.

The Ingame menu will feature only 3 options: Continue game, Restart game, and Forfeit game, and will also serve to pause the game. The Ingame menu will appear in a box on top of the game screen.

- The Continue game option will return the player to the game.
- The Restart game option will restart the game with the present options.
- The Forfeit game option will end the game and take the player to the main menu.

Graphics

All the graphics of Ping will be from SpriteLib. All the backgrounds will be done in a paint program or will feature some kind of demo effect like a space field or plasma.

Graphical engine

Usually this is a separate section of the design document

Ping will feature a really basic graphical engine. It has to be able to load images from disk, blit from surfaces to other surfaces and handle smooth movement.

The text functions will be done using GDI (Graphics Device Interface). Even though this is not the fastest option it will suit the basic text output needs of Ping.

Controls

Usually the player has the choice to set his own controls, and here the actions supported by the controls are described

Controlling the spaceships in Ping is pretty easy. The first player has two keys, Up and Down, to move the spaceship up and down. When present, the second player will use A and D for the same purpose.

Artificial Intelligence

For larger projects I like to also make a separate design document for artificial intelligence, but Ping doesn't need one

Ping will feature 4 difficulty settings, Easy, Normal, Hard, and You're nuts.

The computer controlled ship will calculate the collision point where the ball should hit the ship and move there, the accuracy of the calculation is determined by the difficulty settings, which add a randomness in the tracking algorithm according to the difficulty.

Schedule

This is where usually the most problems occur. Almost no one can exactly make an exact schedule for a game, but it should feature the time for each part that must be developed

Even though Ping needs a schedule as any other game should, I'm not making one because I'm only devoting time to it when possible. Since developing the game and writing these tutorials is more time consuming than just programming the game I chose not to have a schedule.

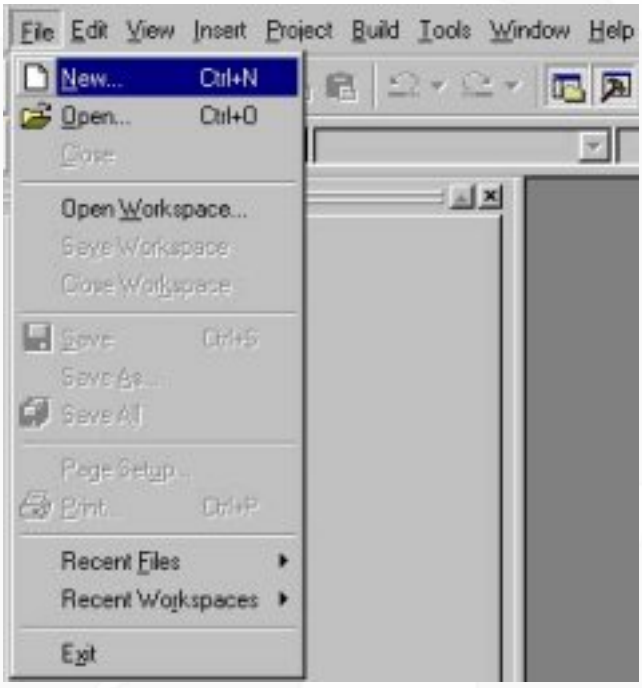
And this is the end of our design document. It's not the best design document but it will help us out a little when developing Ping.

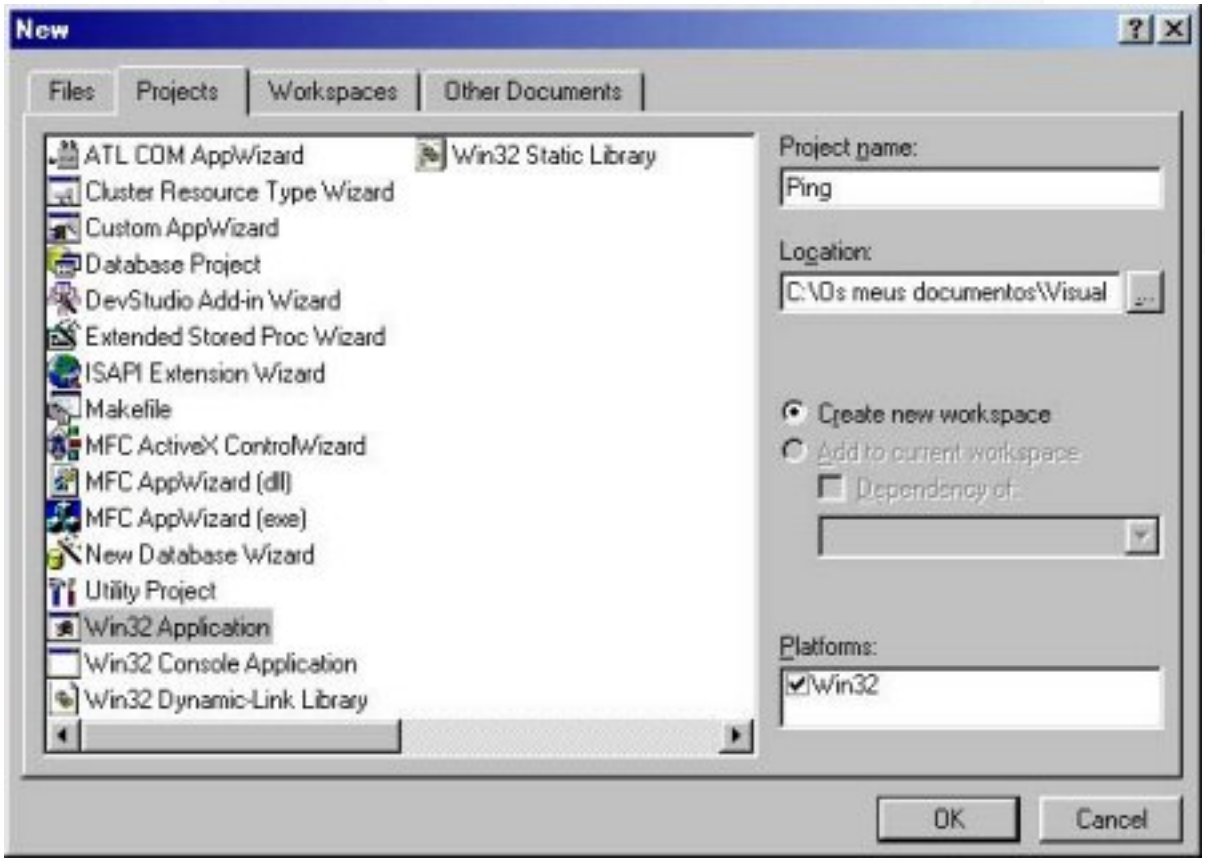
Setting things up

We are not actually going to start programming anything now. It is best to start by setting all things up so that in the next tutorial we don't have to worry about annoying little details.

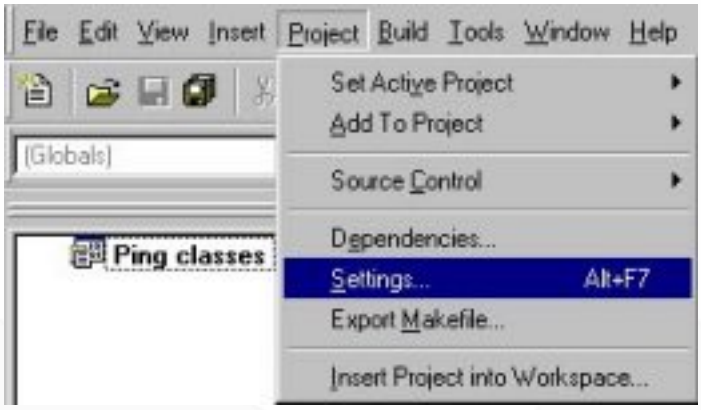
Again, all steps covered here are using Visual C++. If you have another compiler please look in your documentation on how to set it up to compile DirectX programs.

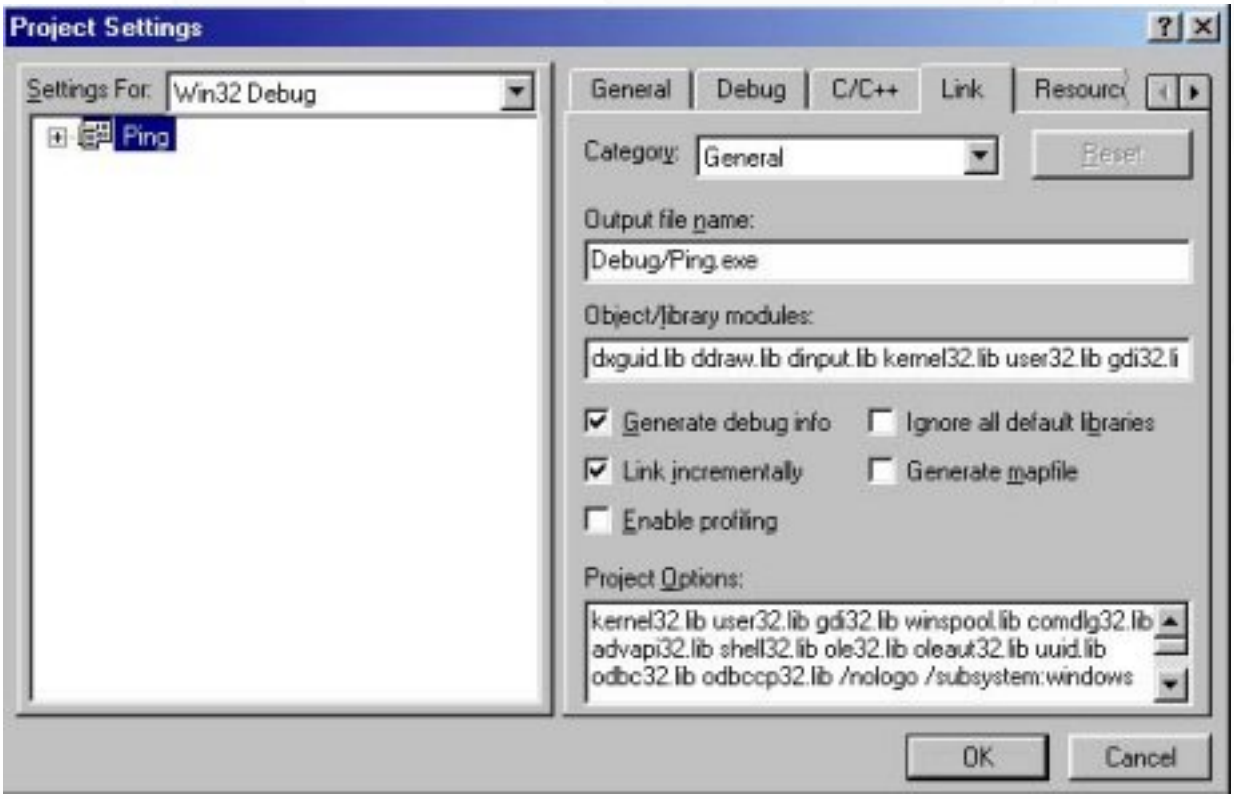
First we need to create a Project. Go to the File menu, then select New and choose Win32 Application. Enter the name Ping in the *Project name* text box. Click Ok.





Now we need to add the DirectX libraries to the project. Go to the Project menu and select Settings. Choose the Link tab and in *Object/library modules* text box add `dxguid.lib ddraw.lib dinput.lib`.





Conclusion

In this first part of this series you learned mainly how to design a simple game and how to set up Visual C++ for your future DirectX projects. In the next tutorial we'll get into DirectDraw (graphics component of DirectX) and if we have time DirectInput (input component of DirectX). We will also have a basic introduction to Win32 programming.

Any thoughts, ideas, suggestions, error correction, or if you just want to talk about your dog, mail me at magick_pt@geocities.com.

Until then, stay well and don't forget to watch Scooby Doo.

Discuss this article in the forums

© 1999-2001 Gamedev.net. All rights reserved. Terms of Use Privacy Policy
Comments? Questions? Feedback? Send us an e-mail!