11th International Research/Expert Conference
"Trends in the Development of Machinery and Associated Technology"
TMT 2007, Hammamet, Tunisia, 05-09 September, 2007.

VISUALIZATION WITH THE DIRECTX

Pavel Pokorny
Thomas Bata University in Zlin, Faculty of Applied Informatics
Mostni 5139, Zlin
Czech Republic

ABSTRACT
The modern control systems are composed from the more layer structure. For example, in the 2-layer structure can be the first layer programmable machine in the industry environment connected to the measured and controlled process. And the second layer can be personal computer with visualization and control software. Both phases are connected via the serial or wireless connection. This paper describes programming the visualization applications with the help of DirectX.
Keywords: Visualization, DirectX, programming.

1. INTRODUCTION
Technical visualization is an important aspect of product development. People from many disciplines and with varying requirements across an organization need to view and manipulate product data in different ways. CAD application is the traditional tool of the engineer who needs to create and edit the geometry, but others need to see this data or convey information in other ways to that of the engineer. For example, many users work in a visualized system environment and require a neutral viewing format within the organization. Therefore product visualization technology has been developed.
The modern control systems are composed from the more layer structure. The highest layer can be personal computer with visualization and control software. We can use varied applications for the controlling and visualization. Intouch or ControlWeb are most used. These applications have some advantages (for example variability and increase engineering productivity), but some disadvantages too (for example they are not good for small control systems and these applications are commercial - they are high priced).
The commercial applications offer outstanding ease of use and simple-to-configure graphics and significant enhancements that result in tremendous improvements in both operational and engineering productivity. Powerful wizards enable users to quickly create and deploy customized applications that connect and deliver real-time information.
In the next chapters one way is described, how to create simple visualization application in the Widows platform. It includes the creation of Widows program project with the help of Win32 API, implementing DirectX in this project (manipulating with the graphics in DirectDraw and Direct3D) and a short example to visualize simple control system.

2. WINDOWS PROGRAM PROJECT
To create a simple Windows project, it was selected C/C++ language and the compiler Microsoft Visual C++ 6.0. The program will be use the native Windows application programming interface (API), which is available for Windows95/98/ME/NT/XP. This programming interface is based on the features of Windows environment. The characteristic for these platforms are multitasking and multithreading. Windows possesses a graphical user interface (GUI), sometimes also called a "visual interface" or "graphical windowing environment." All GUIs make use of graphics on a bitmapped video display. Graphics provides better utilization of screen real estate, a visually rich environment for conveying information. From the programmer's perspective, the consistent user interface results

from using the routines built into Windows for constructing menus and dialog boxes. All menus have the same keyboard and mouse interface because Windows handles this job. [3]
The most obvious windows adorning our desktop are application windows. These windows contain a title bar that shows the program's name, a menu, and perhaps a toolbar and a scroll bar. Another type of window is the dialog box, which may or may not have a title bar. The user sees these windows as objects on the screen and interacts directly with them using the keyboard or the mouse. Interestingly enough, the programmer's perspective is analogous to the user's perspective. The window receives the user input in the form of "messages" to the window. A window also uses messages to communicate with other windows. Getting a good feel for messages is an important part of learning how to write programs for Windows. [3]
To create a Windows application in Visual C++ 6.0, click on the menu *File-New*. In the project window we must select the type of application. The standard Windows application has this type as "Win32 Application". After enter name of project and directory, where it will be saved we can click on the button "OK". On the following window we select type of application. We can switch between the empty, simple Win32 application and simple "Hello word" application. The simplest way is to select "Simple Win32 application". There will be created project with the source code of base Windows application. This code will be contains the creation of window class (here are defined features of this window), its registering and the creation of the window. After this the window is displayed and runs loop of Windows messages processing. We can respond to the required messages in the message procedure and the others messages are returned to Windows system. The example of these messages are mouse click, pressing keys or the switch between the activate/deactivate window.

3. DIRECTX

DirectX is the Microsoft collection of APIs that are designed to give game developers a low-level interface to the PC hardware that is running Windows. On version 9.0 (the last version for Windows XP), each DirectX API component provides access to different aspects of the hardware, including graphics, sound, and networking, all through a standard interface. This interface allows developers to write their games using one set of functions, regardless of the hardware they’re being run on. [4]
The DirectX API contains multiple components, each one representing a different aspect of the system. Each API can be used independently, thereby adding only the functionality, which our application requires. The DirectX components are DirectX Graphics (it handles all graphics output), DirectInput (all user input keyboard, mouse, joystick, gamepad), DirectPlay (network support for our applications - communication with other computers allows more than one person to play), DirectSound (sound effects or background music), DirectMusic (dynamic soundtrack, which can be played back on a timed schedule or adapted to the gameplay using pitch, tempo, or volume changes) and DirectShow (cut scenes and streaming audio – for example playing AVI, MP3, MPEG, and ASF files).
In this paper there we describe only work with DirectX graphics. In includes DirectDraw and Direct3D. Next paragraphs contain informations about these APIs. Now we show, how to implement the support DirectX API into our Windows applications.
This implementation is not difficult. We must download and install the DirectX SDK (software development kit) in our computer. It is free [1] and contains DirectX required header files, libraries, examples, documentation. Then we need to set the path into header and library directories into our compiler. It is set on the text menu "Tools-Options" and card "Project and solutions". The final step is to include header files and libraries into our project. We include the required files only. For the example, when we want to create DirectDraw application, we include the header file "ddraw.h" and libraries "dxguid.lib" (in contains the ground of DirectX algorithms) and "ddraw.lib".

3.1. DirectDraw

DirectDraw is a 2D API. That is, it contains commands for 2D rendering and does not support 3D hardware acceleration. A programmer could use DirectDraw to draw 3D graphics, but the rendering would be slow compared to an API such as Direct3D which does support 3D hardware acceleration. As of DirectX version 8.0, DirectDraw was merged into a new package called DirectX Graphics, which is really just Direct3D with a few DirectDraw API additions. DirectDraw can still be used by programmers, but they must use older DirectX interfaces (DirectX 7 and below). [2]

DirectDraw has the limiting for 2D graphics. But in has some benefits. The fist benefit is low hardware requirements. The applications based on DirectDraw need small memory and CPU/GPU calculation time. The second benefit is the simplest source code of these applications. Such source code is better managed and there is easier error removing. And the last benefit is the enough graphic visualization.

3.2. Direct3D

Direct3D is a 3D API. That is, it contains many commands for 3D rendering, but contains few commands for rendering 2D graphics. Microsoft strives to continually update Direct3D to support the latest technology available on 3D graphics cards. Direct3D offers full vertex software emulation but no pixel software emulation for features not available in hardware. For example, if a program programmed using Direct3D requires pixel shaders and the video card on the user's computer does not support that feature, Direct3D will not emulate it. Direct3D is formed by two big APIs. Retained Mode and Immediate Mode. Immediate mode provides an interface to every video card 3D functions (lighting, materials, clipping, transformations, textures, depth buffering...). Retained mode is built over the previous one and provides higher level graphics techniques such as hierarchy and animation. Immediate mode is preferred by video game developers because it gives them more freedom to use graphics techniques. [5]

In Direct3D is the computer graphics future. It supports many visualization techniques to show more real scenes. But when the displaying is more realistic, then the hardware requirements are increased. Therefore, this API is used on most modern computers. Source code of these applications is more extended and less transparent. And it must not to be more cleared in the visualization of the technological process.

4. SHORT EXAMPLE WITH DIRECTDRAW

In this chapter we describe a simple visualization example with the help of Win32 API and DirectDraw. It will be control of speed of mixer. The mixer speed will be numerical displayed (the better way is the possibility to set it). But more illustration is to graphics displaying this mixer speed. The simple scheme of mixer is on the figure 1 left. When we want to display rotation of this mixer we need some phases of positions.

The starting position is attained, when the mixer is rotated to 360 degree. Because this displaying needs to be continuous, there we must have some phases of rotation. Acceptable is 24 phases (each mixer position is rotated to 15 degree). Then we have 24 pictures in some raster graphics format (for example *.bmp, *.png or *.jpg) and these pictures will be stepwise changed (the speed of change these pictures on the screen will be depend on the speed of mixer) and this change displaying give us the animation effect the rotated mixer. The first four phases of this rotation (to 15 degree) are shown on the figure 1.

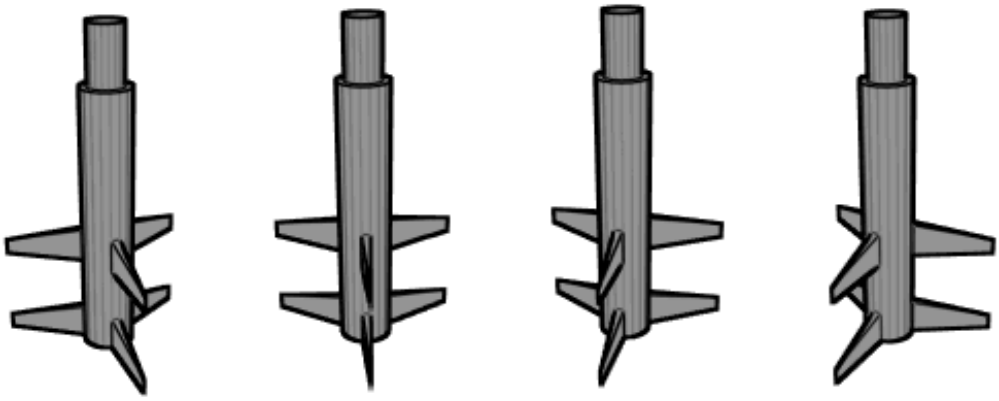

Figure 1. Four phases (positions) of mixer

This implementation to the Win32 program with DirectDraw is not difficult. The DirectDraw is based on surfaces. Surfaces are blocks in computer memory, where are saved displaying informations. The base (and the most important) surface is the front buffer and it contains, what we see on the screen.

The second base surface is the back buffer. Here are saved displaying informations, which will be displayed in the next step. When it is the right time to display it, these two surfaces are flipped or the back buffer is transferred to front buffer and then we can save new graphics informations to the back buffer. This process helps to prevent unpleasant flicker effect.
In DirectDraw we can have many surfaces. Except the one front buffer and one or more back buffers, here exist so-called off-screen surfaces. These surfaces contain graphics informations and as necessary they are transferred to the back buffer. Here are saved pictures often and here we can save our pictures (phases) of rotated mixer. They can be saved in the 24 separated surfaces or in one surface only. When they are saved to the separated surfaces, we need to write more source code to serve, which from the picture needs to be displayed, but this code is transparent. Only one surface has the benefit of short and efficient source code, but it is not very transparent. Additional, the pictures must to be periodically placed in this one surface.
These described algorithms are easy to create them and because DirectDraw has not exacting system and hardware requirements, to run this application we need not the modern computer.

5. CONCLUSION

The using of the visualized control systems is very popular at presents. The modern control applications, which use the visualize give us better illustration of the controlled process. To this visualization, we can work with the commercial visualization programs (like Intouch or ControlWeb) or we can program own application. The benefit of commercial programs is the universality and the relative easy possibility of large applications creation. Against this, they are very costly and they have frequently needless computer requirements.
The benefit of the application, which we develop, is the fact, that it is "made to measure" – there are not needless system and hardware requirements. But in dependency of difficulty of control system, this application can be complicated to program it.
The example on this paper presents simple control application based on Win32API and DirectDraw to visualize the controlling of mixer speed. This is only simulation of this process, but, it should not be very difficult to implement it into the real environment.

6. ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education of the Czech Republic in the range of research projects No. MSM 7088352102.

7. REFERENCES

[1] Commercial electronic documentation: Microsoft DirectX SDK [on line]. Place of edition: Microsoft Corporations, 1290 Avenue of the Americas, Sixth Floor, New York, 2007. [cit. 2.5.2007]. Available on: < http://www.microsoft.com/directx>.
[2] Dunlop, R., Shepherd, D. & Martin, M.: Sams Teach Yourself DirectX 7 in 24 Hours. 1th ed. Sams Publisher, 1999, 450 p. (ISBN 067231634X.)
[3] Petzold, Ch.: Programming Windows. 5th ed. Microsoft Press, 1998, 1478 p. (ISBN 157231995X).
[4] Sanchez, J., Canton, M.P.: DirectX 3D graphics programming bible. 1. ed., Hungry MindsBk&CD-Rom edition, 780 p. (ISBN 0764546333)
[5] Walnum, C.: Programujeme grafiku v Microsoft Direct3D (in Czech). 1. ed., Computer Press, Praha 2004. 360 p. (ISBN 8025101363).