



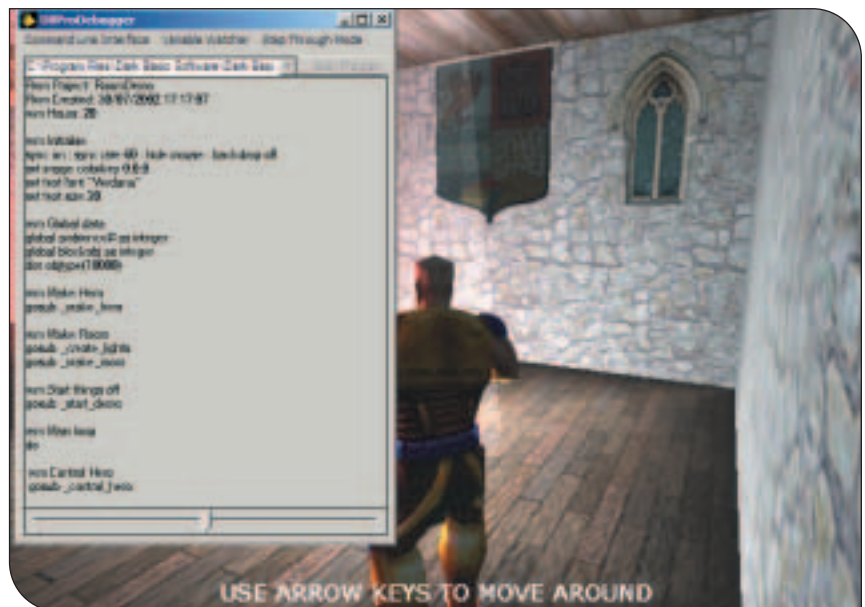
## DarkBASIC Professional

by Justin Lloyd

**R**emember when every microcomputer shipped with a version of BASIC? When you didn't have to resort to pages of C++ code calling obscure API functions just to put a color rectangle on-screen? DarkBASIC Professional is a hidden gem often overlooked by the commercial game development community — a modern, structured BASIC dialect with a lot of commands specifically for handling games using DirectX. It is capable of turning out a 2D platformers and 2D shooters as easily it handles 3D first-person shooters, both indoor and outdoor.

So what kind of games can you create with DarkBASIC Professional? Pretty much anything you can create in C++/Win32/DirectX, though the learning curve is shallower. It is an ideal “scripting language” for game designers and game programmers in a rush to try out an idea. DarkBASIC allows you to get the idea up on-screen rapidly with minimum fuss, and from there take it all the way through to conclusion, including burning a gold master, if necessary.

The current release, unlike previous versions, is a native Windows IDE that provides an editor and integrated compiler. The editor is reasonably well featured but is still primitive in places. The IDE is just about capable of the task for which it's designed but little more. It handles multiple source files, assets, version num-



**FIGURE 1.** Typically, character and game state status flow from the game engine to the AI middleware, and then character control requests flow from the AI middleware to the game engine, and are acted out by the characters.

bering, and a TO DO list.

The BASIC compiler turns out real machine code, and a source-level debugger is included providing basic features such as breakpoints, plus single-step and variable watches, both auto-scope and user-defined. Disassembling the compiled output indicates the compiler is not optimizing any of the code. It's just a straight conversion from BASIC to assembly with the ability to call in to other DLLs, mak-

ing it extensible by a programmer capable of writing dynamic link libraries in other tools such as Visual BASIC, C++, or Delphi. The compiler turns out stand-alone executables with the ability to bundle any additional DLLs and assets into a single executable file, and it also removes any unused portions of the DarkBASIC libraries. Once the executable is generated (with or without bundled assets), it can be automatically compressed and encrypted to create smaller and more secure distributables.

**Graphics Capabilities.** DarkBASIC is specifically created for manipulating the DirectX API in a simple and easy way. It

---

**ERIC DYBSAND** | *Eric has been involved with computer game AI since 1987, doing game design, programming, and testing for a variety of genres of computer games. He has been a speaker on computer game AI at the GDC for the last seven years and is a contributing author on AI to the Game Programming Gems and AI Wisdom series of books.*

handles all of the DirectX 8.1 features, providing ample facilities for 2D — using the DirectX sprite interface — and 3D graphics, plus input and audio. The package makes initialization of a full-screen or windowed DirectX application as simple as can be, by either setting the startup options in the project explorer or issuing a BASIC command that dictates screen resolution along with windowed or exclusive mode. The project settings also control the type of executable that is created—whether it is standalone or a Windows installer.

The 2D sprite system is very capable, handling animations, movement, and collision between sprites. The 3D side is an even more comprehensive command set for graphics and math, covering vectors, matrices, Catmull-Rom and Hermite splines, plus all the usual trigonometry commands found in other modern BASIC dialects. The 3D graphics handle the usual rendering of polygons, loading of 3D objects from disk, and creating primitives at run time, plus commands to control the individual joints on a boned object and real-time mesh deformation with direct support for a lot of popular model formats, including Quake 2 and 3, Half-Life, DirectX, and 3DS Max.

Advanced texturing and rendering techniques using bump mapping, environment mapping, and multitexturing give the shine to games, along with support for hardware shadows, vertex and pixel shaders, and even a built-in cartoon shader. Rounding out the graphics capabilities is a general-purpose particle system with specific support for the unique properties of snow and fire particles.

Camera control allows for multiple simultaneous cameras for multi-viewport rendering to either the back buffer or a bitmap. A camera is moved around dynamically via code or the tracking feature that makes it follow another object. Pointing a camera at a specific object or point in space is trivial, especially for non-programmers unfamiliar with look-at matrices. One of the nice features I explored was the intelligent navigation. By placing invisible static collision boxes in a scene, the camera will attempt to avoid entering the boxes. Scene lights are fully

controllable, generated either in a level editor or dynamically at run time, with commands for spot, point, and ambient, along with control of fogging distance.

Object collision detection is handled automatically, with bounding spheres for 3D, and rectangles for 2D sprites. Once a collision has taken place, DarkBASIC can either automatically move the colliding objects apart, providing for a primitive collision resolution, or leave it up to the programmer to determine the exact outcome. The collision command set provides raycasting tests for user-determined collisions and probing.

DarkBASIC renders indoor and outdoor 3D scenes equally well with PVS and BSP support and a capable terrain engine that loads BSP worlds, handling the culling, texturing, and collision automatically. There's a bundled BSP compiler that's installed with the IDE, and a programmer can generate dynamic terrain from height map files at run time.

**Other Features.** Beyond the graphics features, provision is made for input from keyboard, joystick/joypad and mouse, along with force-feedback capability. There are also extensions available for free from the DarkBASIC support web site that interface with a VR glove and USB light gun. The language provides strong audio support; streaming CD audio, MIDI and MP3 playback for music, and WAV and other formats for the sound effects, all of which can be processed as 3D positional audio. It is also possible to capture directly from the microphone, which proves useful for networked games. Video, either from an AVI/MPEG file or DVD can be mapped on to a texture in real time and then wrapped around an object.

DarkBASIC features two command sets that deal with network communication. The first is for accessing FTP sites where you can retrieve or store any data you want—this feature can be used for sending high scores to a remote server, downloading the latest patch or level pack for an established game, or even automatically upgrading the game from unregistered to registered. The other command set handles multiplayer games via LAN or

TCP/IP using a peer-to-peer or client/server architecture. With this system, the hardest part of creating a client/server multiplayer game is presenting a meaningful user interface to allow the player to connect. I wasn't expecting very much from this area of the package. Usually network communication in many game SDKs is bolted on an as afterthought and allows you to send only the most rudimentary of messages. DarkBASIC handles the basics and more, with provision for transmission and reception of arbitrary blocks of memory, bitmaps, audio, and complete object meshes.

**Documentation and Help.** The manual, spiral bound so that it will lie flat on the desktop next to your keyboard, covers

## DARKBASIC PRO ★★★★★

### STATS

Dark Basic Software  
Wigan, Lancashire, U.K.  
Fax: +44 (0)8451 275 338  
[www.darkbasic.com](http://www.darkbasic.com)

### PRICE

\$99.00

### SYSTEM REQUIREMENTS

Recommended Specification: Pentium 3 733Mhz or above, 128MB RAM, DirectX compatible Graphics card with at least 32MB RAM and hardware 3D acceleration. DirectX compatible Sound card, 16x CD-Rom, 400Meg hard drive space

### PROS

1. Simplifies access to all aspects of DirectX.
2. Allows you to quickly get an idea up and running.
3. Once the framework is in place, designers can treat it just like scripting.

### CONS

1. Requires CD or USB dongle to be inserted in the machine at all times.
2. Can be clumsy to express some ideas due to lack of object-oriented features.
3. Lack of professional production features could make it difficult to integrate in to a professional development environment.



the BASIC command set. Each section is logically divided so that it deals with a specific area: 2D sprites, 3D objects, vectors and matrices, memory manipulation, and so on. Each BASIC command is given a brief paragraph of information and the syntax that it requires. There are a few weaknesses in the manual: For one thing, it doesn't supply brief examples of how to use commands. Often many commands will work in conjunction, and the user must infer how they bolt together. Many, but not all, of the commands are covered in the online examples replete with source code, and the online help provides more comprehensive information with small code snippets. I don't see the need for the physical manual — either bring it up to the level of the online documentation or abandon it all together. The paper manual is little more than filler. It would be nice to see a nice, thick user guide that covers the particular BASIC dialect in more depth. One complaint I have with the online help is that it uses a non-“Windows standard” help browser. Is there any reason to deviate from the capable and usable Windows help format these days?

**Wrap Up.** DarkBASIC is a great tool for developing both complete games and quick mock-ups, yet has been overlooked by many professional game developers who couldn't imagine that BASIC could be up to the challenge of creating a modern game with sophisticated graphics. Just as we were going to press, the company told us it was about to post a public beta of DarkBASIC 5, which will have many improvements, including support for the .FX shader format, and pixel-accurate selection of objects during game play.

The compiler & IDE package are actively supported by the publisher/developer with regular updates and a monthly news letter. Cost is \$99.99, with no royalty fees, and access to the DarkBASIC Developer Network costs an additional \$69 per year. Consult the website, which frequently runs special offers. The publisher offers educational support to schools and universities with free licenses available.

## Right Hemisphere's Deep Paint 2 by sean wagstaff

One of the chief challenges in creating visually exciting game art is painting interesting, realistic, and visually rich scenery and textures quickly and efficiently. While Adobe's Photoshop is indisputably the gold standard for just about every kind of digital image editing, Right Hemisphere's Deep Paint 2 offers a number of unusual creative features that complement Photoshop's powerful toolset.

Rather than trying to compete with Photoshop head-on, Right Hemisphere has wisely set up Deep Paint as a companion to Photoshop and has included import and export functionality, along with a Photoshop plug-in that lets the two work together. As a result, you can paint with Deep Paint's unusual depth-mapped brushes, while retaining the fine control and extensive tools of Photoshop.

Version 2 of Deep Paint introduced a number of useful new features, including greatly enhanced cloning tools, for painting based on the underlying pixels in imported layers, and spline tools, which can be used for accurately creating masks or for creating stroke paths for various brushes.

Chief among Deep Paint's unusual features is the capability to paint with textures, including bump mapping and specular, in addition to using basic colors. For example, you can easily create a panel of riveted steel by painting on rivets that are bump- and specular-mapped differently from the surrounding metal plate, or you can paint with stones and other objects that have their own transparency masks and bump values.

Game artists involved in creating environments will appreciate the capability to add textures as texture brushes. You can create your own texture brushes by creating your own sets of color, bump, specular, and alpha maps, and then use these brushes to paint the textures into an image quickly.

In addition to texture brushes, Deep Paint does an impressive job of simulating wet and dry paints that mimic real-world materials, including thick, gooey

oil paints and other materials where depth and specular shine are an important element.

Deep Paint has always let you set lighting that is automatically rendered on the bump-mapped surfaces of your painting, and you can choose to retain this rendered lighting for use in your textures, or to omit the lighting if your rendering engine is going use the bump maps you've created.

Typically, the texture-mapping workflow involves exporting UV maps to an image file and then painting over these maps to achieve the desired texture. Deep Paint's texture brushes make this easy, although unlike Deep Paint 3D, the 2D version has no 3D object painting features of its own.

While Deep Paint excels for painting textures and images that have bump, specular, and lighting effects prerendered, it's not so ideal if you are creating textures for games that render bump and specular effects in-engine. In this case, you'll need to generate separate maps for the color, bump, and specular values. While it's great to be able to see depth and highlights enabled as you paint,

### VENDOR SHEET

- **AI Implant**  
Biographic Technologies  
Montreal, Quebec, Canada  
(514) 844-5255 x250  
[www.ai-implant.com/games](http://www.ai-implant.com/games)
- **DirectIA**  
MASA Group  
Paris, France  
U.S.: (212) 343-8838  
[www.directia.com](http://www.directia.com)
- **Renderware AI**  
Criterion Software  
Austin, Tex.  
(512) 478-5605  
[www.renderware.com/renderwareai.html](http://www.renderware.com/renderwareai.html)
- **Simbionic**  
Stottler Henke  
San Mateo, Calif.  
(650) 655-7242  
[www.simbionic.com](http://www.simbionic.com)

Deep Paint offers no provision for exporting these effects into separate map files, so its usefulness is severely reduced in this environment.

I was impressed with Deep Paint's ability to render realistic looking alpha-, depth-, and specular-mapped textures in 2D paintings, but for game artists, Right Hemisphere could add a number of features to make it more useful. Primarily, the software needs the capability to view, edit, and export the underlying effects channels that control these effects. Even for 2D artists who simply want deeper-level control over their images, more control over the underlying channels would be a good thing. I'd also like to see a wrap-around feature in the brushes that would let us paint textures that automatically tile seamlessly.

Deep Paint 2 is available for Windows 98/NT 4/2000/XP for \$249 (\$49 for an upgrade). As an add-on to Photoshop, it is a valuable addition to the creative toolbox.

4 stars | Deep Paint 2 | Right Hemisphere | [www.righthemisphere.com](http://www.righthemisphere.com)

## Advanced 3D Game Programming with DirectX 9.0

written by Peter Walsh and Adrian Perez  
Reviewed by jeremy jessup

**A**dvanced 3D Game Programming with DirectX 9.0 by Peter Walsh covers a broad range of subjects critical to making games: graphics, artificial intelligence, networking, and mathematics. Priced at \$60, the book contains 11 chapters that span approximately 520 pages.

The first chapter, "Windows," describes how to create a window and respond to some of the common Windows messaging events. The chapter defines several custom classes that loosely resemble code created by Visual Studio's workspace wizard but which is cleaner and in a Win32 flavor. These classes form the framework for a generic Windows game.

The next three chapters ("Getting

Started with DirectX," "DirectInput," and "DirectSound") show how to compile and link DirectX with your application and initialize two of the subsystems found in DirectX, DirectSound, and DirectInput. The subsystems are briefly highlighted, and wrapper classes are given to simplify their usage. The DirectInput and DirectSound chapters focus on initialization of each system rather than exploring the more sophisticated uses of each system such as force feedback or dynamic audio mixing.

Chapters on 3D math, artificial intelligence, and networking follow. The math chapter provides basic math definitions including the dot and cross products as well as container classes for vectors and matrices. The AI chapter is brief. Readers seeking to gain a deeper understanding should read the chapter in conjunction with a decent college text that describes fundamental search routines such as A\* or Dijkstra's algorithm. Finally, the networking chapter relies on WinSock without mentioning DirectPlay. Classes are provided to encapsulate the network layer of a game. While all three chapters are essential to game programming, none adequately covered the complexity and nuances of each subject given the space provided.

The remaining chapters presented in the last fifth of the book discuss rendering and are easily the high point of the text. Beginning with lighting and fog parameters, Walsh explores several sophisticated graphics techniques including the mathematics of animation, subdivision of surfaces, radiosity, and progressive meshes. Walsh attempts to detail advanced topics like multi-texture and multi-pass rendering using the fixed function pipeline. Then, examples of multi-pass texture mapping (light maps, environment maps, and glow maps) are provided to illustrate various DirectX render states. Last, Walsh discusses scene management to assist in reducing the number of objects drawn per frame by using portals and occludes to test for polygon visibility. Yet despite featuring DirectX 9.0, many of the new SDK features were missing from the text, such as vertex and pixel shaders, displacement maps, or the

two-sided stencil mode.

For the price of the book, a companion CD containing the source code would have been beneficial, although the sample code is available online. When I downloaded the sample code to try out the examples, three of the four programs crashed because the DirectX device on my GeForce 4 wasn't successfully initialized when rendering began. Since the examples executed in full-screen mode, the computer needed to be rebooted. Additionally, the sample code contained a couple C/C++ techniques that are not commonly found in game development, such as exceptions and nameless unions.

Overall, this book's title is at odds with the subject matter. It provides an overview of the basic theory, API calls, and usage. When more details would help to clarify a point, the reader was often referred to the DirectX SDK help. As an experienced game developer, I found very little of value in this book. None of the topics is explored adequately, leaving the seasoned reader with nothing but an unsatisfying overview and possibly a reference to the DirectX SDK help file.

Ultimately, it feels as if they've taken the first edition of this book for DirectX 7 and just updated API calls here and there, as it definitely doesn't capture "state of the art" in any sense.

one star | Advanced 3D Game Programming with DirectX 9.0 | Wordware | [www.wordware.com](http://www.wordware.com)