Many developers became interested in programming because they saw a video game and thought, “How did they do that?” This book helps demystify what is required to make video games. Being able to write games on a next-generation console such as the Xbox 360 has never been an option for the masses before. Now with the XNA Framework, games can be written for the console.

By the end of this book, you will have created four complete games and many demos along the way. This book takes a serious look at performance-related issues when writing games using XNA for Windows and the Xbox 360. Two chapters are devoted to the High Level Shader Language (HLSL), which is a necessity for writing great games. The book covers physics and artificial intelligence (AI). It also covers special effects, including explosions, transitions, and how to create a 3D particle system. It demonstrates how to create a sound project using the Microsoft Cross-Platform Audio Creation Tool (XACT) and how to directly access sound files in a game. Two chapters are devoted to programming games for the Zune. Saving and loading a high score list and creating a full menu system are also taught in this book. Five chapters are devoted to creating multiplayer games. Writing network games can be challenging, and this book covers networking in detail. The final two chapters are on best practices and provide tips on how to sell games on the Xbox LIVE Marketplace. In general, this book contains a great foundation for many topics that need to be learned to create and sell a full-featured single-player or multiplayer game.

Who Should Read This Book?

This book was written for developers. You should have a good understanding of programming in general. The book uses C#, but if you know any modern language, such as C++, Java, or VB.NET, you will have no problem understanding the code in this book. The book assumes some understanding of the Microsoft .NET Framework, which is what the XNA Framework runs on. Without prior experience writing code using the .NET Framework, you might have to do a little research now and then, but should not have trouble getting through this book.

This book was written with a few different audiences in mind. Business application developers who want to use their programming skill set to write computer games are one audience. Graphics and game developers who have been around the OpenGL and DirectX block should also find useful information in this book—especially in seeing how things are done “the XNA way.” The book also targets readers who have some programming experience but have not done anything formal. The book teaches by example. It is written in such a way that if you are not in front of your computer, you can still get valuable information from the book because the code is presented as it is being discussed.
Hardware and Software Requirements

The code in this book is compiled with XNA Game Studio 3.0. In order to complete the games and demos in this book, the requirements that follow must be met.

Supported Operating Systems

The following operating systems are supported:

- Windows XP Home Edition
- Windows XP Professional Edition
- Windows XP Media Center Edition
- Windows XP Tablet Edition
- Windows Vista Home Basic Edition
- Windows Vista Home Premium Edition
- Windows Vista Business Edition
- Windows Vista Enterprise Edition
- Windows Vista Ultimate Edition

Windows XP requires Service Pack 2 or later.

Hardware Requirements

When you run XNA Framework games on Windows, a graphics card that supports Shader Model 1.1 is required. This book has samples that use Shader Model 2.0 and a couple that use Shader Model 3.0. To get the most from this book, you need a graphics card that supports Shader Model 3.0. The graphics card should have the most up-to-date drivers. Updated drivers can be found on the graphics card’s hardware vendor website.

When you run XNA Framework games on the Xbox 360 console, a hard drive must be connected to the console.

Software Requirements

All the software required to utilize the XNA Framework on Windows is free:

- Microsoft Visual C# 2005 Express Edition
- Microsoft XNA Game Studio Express
- DirectX 9.0c

Instructions on installing the software can be found in Chapter 1, “Introducing the XNA Framework and XNA Game Studio.”
Code Examples

The source code for the examples in this book can be found on the accompanying CD. Any updates to the code can be downloaded via www.samspublishing.com or www.xnaessentials.com.

How This Book Is Organized

This book is organized into 11 main parts, representing the information you need to understand to use XNA Game Studio effectively. Writing a book is an interesting challenge. There are basically two routes an author can go. One route is to create small bite-sized pieces that can be used as a reference. The other route is to take the reader on a journey from start to finish, covering important topics along the way but doing it in such a manner that the reader is gradually learning concepts. Then, once the entire book has been enjoyed, the reader can go back and reread certain sections for mastery.

I have tried to take the second approach in writing this book. The book is best read in order. The Internet has a wealth of information. Learning about a particular topic is not difficult. You can easily find information from many different sources on a particular topic. The problem is there is usually no place to see how a lot of different topics work together. With a book that is designed to be read from front to back, the main drawback is a larger time commitment. However, there is usually deeper understanding by the time the task is complete versus the same amount of time spent looking at particular topics on the subject from online tutorials and blog posts. Both are very important, but because a wealth of reference information is available online already, there was no need to make this a reference book.

There was some criticism concerning the order of the first book. This book is not organized in a manner similar to many other books. However, a lot of thought was put into the order of this book. I do believe this book’s order is important, and I did not change it from the first edition. I start with a very basic chapter explaining the history of XNA and very detailed instructions on how to install XNA Game Studio. Most people will not need this, but it is there for those who do. The next chapter jumps right in to talking about the Xbox 360. Even though there are people who do not have an Xbox 360, it is important to put this chapter up front so you can be aware of certain things when creating games using XNA. It is always important to know what you are up against before you start. It is for this same reason that the very next chapter is on performance. Most books simply give a nod to performance in a later chapter or maybe an appendix, if at all. I personally believe that thinking about performance early on is crucial to making a good game. This does not mean we need to do micro-optimizations early in the process; instead, it is all about measurement. This is why performance is discussed so early in the book.
The first real game code that is presented in this book is written for 3D. Many people are shocked that 2D is not discussed until Chapter 9, “2D Basics.” The reason for putting 3D before 2D in this book is because picking up 3D is not any harder than learning 2D. The early chapters are there to introduce you to the XNA Framework as well as the concepts behind a camera. It is my hope to tear down the mental block many people have that 3D is much harder than 2D. Granted, there are some complex topics surrounding 3D, and those are covered later in the book. However, by getting started by drawing models and responding to input, you’ll see there is not a huge difference in the knowledge needed to write 3D games versus 2D games.

After discussing 3D and the Content Pipeline, the book discusses 2D and then moves into two chapters devoted to programming for the Zune. The next part of the book discusses the High Level Shader Language. Physics and artificial intelligence are discussed next. The code for those chapters uses the basic 3D information you will learn in earlier parts the book.

This is followed up by talking about more advanced 3D topics. A single-player 3D game is then built, thus allowing us to put into practice all you will learn in this book.

The next part of the book provides an intensive look at developing multiplayer games. Then the final part of the book discusses some best practices, most of which are done while creating the demos and games throughout the book. The last chapter explains the review process and getting your game into a condition to be sold on the Xbox LIVE Marketplace.
CHAPTER 1

Introducing the XNA Framework and XNA Game Studio

Most developers I know decided to enter the computer field and specifically programming because of computer games. Game development can be one of the most challenging disciplines of software engineering—it can also be the most rewarding!

Never before has it been possible for the masses to create games for a game console, much less a next-generation game console. As a relatively new technology, XNA is going to experience tremendous growth. The sooner we get to know this technology, the better we will be able to understand the changes that will come in the future.

Microsoft is leading the way in how content will be created for game consoles. Soon other game console manufacturers will be jumping at a way to allow the public to create content for their machines. The great news for the Xbox 360 is that Microsoft has spent a lot time over the years creating productive and stable development environments for developers. We will be installing one of Microsoft’s latest integrated development environments (IDEs) in this chapter. Before we get to that, though, let’s take a look at the technology we discuss in this book—XNA.

What Is the XNA Framework?

You have probably heard the statement, “To know where you are going, you need to know where you have been.” I am uncertain if that is entirely true, but I do believe it applies here. Before we dig into exactly what XNA is and what it can do for us, let’s take a moment to look at DirectX because that is what the XNA Framework is built on.
The Foundation of the XNA Framework

Let’s take a journey back to the days of DOS on the PC. When programming games, graphic demos, and the like in DOS, programmers typically had to write low-level code to talk directly to the sound card, graphics cards, and input devices. This was tedious, and the resulting code was error prone because different manufacturers would handle different BIOS interrupts, I/O ports, and memory banks differently. Therefore, the code would work on one system and not another.

Later, Microsoft released the Windows 95 operating system. Many game programmers were skeptical at writing games for Windows—and rightly so—because there was no way to get down to the hardware level to do things that required a lot of speed. Windows 95 had a protected memory model that kept developers from directly accessing the low-level interrupts of the hardware.

To solve this problem, Microsoft created a technology called DirectX. It was actually called Windows Game SDK to begin with, but the name was quickly switched after a reporter poked fun at the API names DirectDraw, DirectSound, and DirectPlay, calling the SDK “Direct ‘X.’” Microsoft ran with the name, and DirectX 1.0 was born a few months after Windows 95 was released. I remember working with DirectDraw for a couple of demos back when this technology first came out.

Because of DirectX, developers had a way to write games with one source that would work on all PCs, regardless of their hardware. Hardware vendors were eager to work with Microsoft on standardizing an interface to access their hardware. They created device drivers to which DirectX would map its API, so all of the work that previously had to be done by game programmers was taken care of, and programmers could then spend their time doing what they wanted to—write games! Vendors called this a hardware abstraction layer (HAL). They also developed a hardware emulation layer (HEL), which emulates hardware through software in case hardware isn’t present. Of course, this was slower but it allowed certain games to be run on machines with no special hardware.

After a couple of years, Microsoft released DirectX 3.0, which ran on Windows NT 4 as well as Windows 95. As part of those upgrades, Microsoft introduced Direct3D. This allowed developers to create 3D objects inside of 3D worlds. DirectX 4 was never released, but DirectX 5 was released in 1997 and later had some upgrades to work under Windows 98.

When DirectX 8 came on the scene in 2000, some of the newly available graphics hardware had vertex and pixel shaders. As a result, Microsoft added in a way to pass custom program code to the hardware. Through assembly code, the game developer could manipulate the data the main game passed to the graphics card. This assembly code was consumed directly by the graphics hardware.

When there was no graphics hardware, games were slow, but they were very flexible. Later, as hardware rendering became prominent, the games were faster, but they were not very flexible in that all of the games really started to look the same. Now with shaders, the speed of the hardware is combined with the flexibility for each game to render and light its 3D content differently.
This brings us to present-day DirectX: We are up to DirectX 9 and 10. Before I talk about DirectX 9, I’ll spend some time talking about DirectX 10. DirectX 10 was released at the same time as Microsoft Windows Vista. In fact, DirectX 10 only works on Vista. This is largely due to the fact that Microsoft has made major changes in the driver model for this operating system. DirectX 10 also requires a Shader Model 4.0 graphics card.

The Xbox 360 runs on DirectX 9 plus some additional partial support for Shader Model 3.0 functionality. DirectX 9 is the foundation for Managed DirectX, an API that exposed the core DirectX functionality to .NET Framework developers. There was a lot of concern about whether this “wrapper” could be as fast as the C++ counterparts. Fortunately, it was almost as fast—about 98% was the benchmark touted. I experienced these benchmark speeds first-hand while on the beta team for this technology. I fell in love with Managed DirectX.

The XNA Framework took the lessons learned from Managed DirectX and used that foundation as a launching pad. To be clear, XNA was built from the ground up and was not built on top of Managed DirectX. It doesn’t use the same namespaces as Managed DirectX and is not simply pointing to the Managed DirectX methods in the background. Although XNA utilizes DirectX 9 in the background, there are no references to DirectX’s API like there were in Managed DirectX.

**XNA Today**

XNA is actually a generic term, much like the term .NET. XNA really refers to anything that Microsoft produces that relates to game developers. The XNA Framework is the API we are discussing. The final piece to XNA is the XNA Game Studio application, which we discuss in detail later. This is the IDE we use to develop our XNA games.

**TIP**

In this book, whenever I use the term *XNA*, I am really referring to the XNA Framework, unless otherwise noted.

XNA allows us to do a lot of things. We have easy access to the input devices (keyboard, game pad or controller, mouse). XNA gives us easy access to the graphics hardware. We are able to easily control audio through XNA. XNA provides the ability for us to store information such as high scores and even saved games. XNA also has networking capabilities built in. This was introduced in version 2.0 of the product. Microsoft uses the Xbox LIVE technology for network support.

To get started using XNA, you have to install some software. You need to install the latest version of DirectX 9 as well as have a graphics card that supports DirectX 9.0c and Shader Model 1.1. (You should get a card that supports Shader Model 3.0 because some of the examples, including the starter kit we use in this chapter and the next one, will not run without it.) You also need to install Visual C# Express or one of the other Visual Studio SKUs, the DirectX 9 runtime, and finally XNA Game Studio. Fortunately, all of the software is free! If you don’t have graphics hardware that can support Shader Model 2.0, you
can pick up a card relatively inexpensively for about US$35. If possible, you should purchase a graphics card that can support Shader Model 3.0 because a couple of examples at the end of the book require it. Windows Vista machines have graphics cards that support Shader Model 4.0 and definitely meet the needs of our XNA games.

In the past, only subscribers to the XNA Creators Club could play the games made by other developers. Xbox LIVE Community Games, introduced in version 3.0 of XNA Game Studio, has changed that. Through a peer review process, games can be approved and put on Xbox LIVE for the world to download. Never before has there been such an easy way for a game to be seen by so many people.

Not only is XNA Game Studio great for the professional, it is great for the game hobbyist, the student, as well as someone just getting started because you do not have to shell out a lot of money to get up and running. One exception to this is if you actually want to deploy your games on your Xbox 360. To do that, you need to subscribe to the XNA Creators Club for US$99 a year (or US$49 for four months). Writing games for the PC using XNA is totally free! As an added benefit of paying for the Creators Club subscription, you are able to review other creators’ games and are able to submit your own games to sell on Xbox LIVE Marketplace. This is discussed in Part XI, “Xbox LIVE Community Games.”

Oh, in case you are wondering what XNA stands for, XNA’s Not Acronymed (or so Microsoft says in the XNA FAQ).

**Installing Visual C# 2008 Express**

To get started, you must have the software installed. Let’s start by installing Visual C# 2008 Express.

---

**TIP**

Any Visual Studio 2008 SKU works with XNA Game Studio 3.0.

---

XNA requires C# due to how the Content Pipeline is used. Some people have successfully created demos using other languages, such as VB.NET and even F#. However, this is not currently supported by Microsoft and won’t be discussed in this book. This book assumes you have a good understanding of C#. If you know C++, Java, or VB.NET, you should be able to pick up C# pretty quickly.

I provide detailed steps to make sure anyone who has not worked with Visual C# Express will be able to get it installed with no issues. Feel free to skip this section if you already have a Visual Studio 2008 SKU installed.
To install Visual C# 2008 Express, follow these steps:

1. You will need to be connected to the Internet to install the application. The application can be downloaded by browsing to http://www.microsoft.com/express/download/ and clicking the Visual C# 2008 Express Edition Download link to download and run the vcssetup.exe setup program.

2. Optional. On the Welcome to Setup screen, select the check box to send data about your setup experience to Microsoft. This way, if something goes awry, Microsoft can get the data and try to make the experience better the next time around. This screen is shown in Figure 1.1.

3. Click Next to continue.

4. The next screen is the End-User License Agreement. If you accept the terms, select the check box and click Next.

5. The following screen, shown in Figure 1.2, has two installation options you can check. Neither of these options is required to utilize XNA Game Studio.
CHAPTER 1  Introducing the XNA Framework and XNA Game Studio

6. Click Next to continue.

7. The next screen, shown in Figure 1.3, asks where we would like to install Visual C# Express. Note that other required applications, including Microsoft .NET Framework 3.5, will be installed. This is required because C# runs on the .NET Framework. You will also notice it requires more than 300MB of space.

8. Click Next to continue.

9. Now you are looking at the Installation Progress screen, where you can monitor the progress of the installation.
10. On the Setup Complete screen, you can see the Microsoft Update link. Click it to get any of the latest service packs for Visual C# Express.

11. Click Exit to complete the installation.

**TIP**

After you install Visual C# 2008 Express, a reboot may be required.

You have now successfully installed the first piece of the pie to start creating excellent games with XNA! Before we continue to the next piece of software, you need to open up Visual C# Express. It might take a couple of minutes to launch the first time the application is loaded. Once the Visual C# Express is loaded, you should see the Start Page, shown in Figure 1.4.

![Start Page inside of Visual C# Express](image)

FIGURE 1.4 This is the Start Page inside of Visual C# Express.

The following procedure is optional, but it does ensure that everything is working correctly on your machine:

1. In the Recent Projects section, find Create Project and click the link. You can also create a new project under the File menu.

2. Visual C# Express installed several default templates that you can choose from. Select the Windows Application template, as displayed in Figure 1.5.
3. You can leave the name set to WindowsFormsApplication1 because you will just be discarding this project when we are done.

4. Click OK to create the application.

5. At this point a new project should have been created, and you should be looking at a blank Windows Form called Form1.

6. Press Ctrl+F5 or click Start Without Debugging on the Debug menu.

If everything compiled correctly, the form you just saw in design mode should actually be running. Granted, it doesn’t do anything, but it does prove that you can compile and run C# through Visual C# Express. The end result can be seen in Figure 1.6. Close down the application you just created as well as Visual C# Express. Feel free to discard the application.
Installing the DirectX Runtime

You also need the DirectX 9 runtime if it isn't already on your machine. To get started, follow these steps:

1. Run the dxwebsetup.exe file from Microsoft’s website. This can be found by clicking the DirectX Runtime Web Installer link at the bottom of the XNA Creators Club Online – Downloads web page (http://creators.xna.com/en-US/downloads). This file contains the redistribution package of the February 2007 version of DirectX 9. You will need to be connected to the Internet so it can completely install the application.
2. You are greeted with the End-User License Agreement. Handle with care.
3. The next screen is a dialog box asking where you would like the installation files to be stored. You can pick any directory you want as long as you remember it so you can actually install the runtime—you are simply extracting the files needed to install the runtime.
4. Click OK to continue.
5. You will be prompted to create that directory if the directory entered doesn’t exist. Click Yes to continue.
6. Wait for the dialog box with the progress bar to finish unpacking the files.

Now you can actually install the runtime by following these steps:

1. Browse to the folder where you installed the files and run the dxsetup.exe file to actually install DirectX 9 onto your machine.
2. The welcome screen you see includes the End-User License Agreement. Select the appropriate radio button to continue.
3. Following the agreement is a screen stating that it will install DirectX. Click Next.
4. Once it finishes installing (a progress bar will be visible while the files are being installed), you will be presented with the Installation Complete screen.
5. Simply click Finish to exit the setup.

Now we can move on to installing XNA Game Studio.

Installing XNA Game Studio

To use XNA Game Studio, you can use any of the Visual Studio SKUs, including Visual C# Express.

**WARNING**

You must run the Visual C# Express IDE at least one time before installing XNA Game Studio. If this is not done, not all the functionality will be installed. If XNA Game Studio was installed prematurely, you will need to uninstall XNA Game Studio, run Visual C# Express, and then exit the IDE. Then you will be able to reinstall XNA Game Studio. This is true for any of the Visual Studio SKUs.
To get started, complete the following steps:

1. Run the XNAGS30_setup.msi file from Microsoft’s website. The file can be downloaded by clicking the top link on the XNA Creators Club Online – Downloads website (http://creators.xna.com/en-US/downloads).

2. Click Next to get past the setup welcome screen.

3. The next screen is the End-User License Agreement. If you accept the terms, select the check box and click Next.

4. A notification dialog box opens that allows the Windows Firewall to have rules added to it. These rules allow communication between the computer and the Xbox 360, as well as allow for communication between network games. This can be seen in Figure 1.7.

5. Click Install to continue. The next screen shows the progress of the installation.

6. Once all of the required files are installed, you are presented with a completion dialog box. Simply click Finish to exit the setup.

After you have installed XNA Game Studio, you can go to the Start menu and see that it added a few more items than those contained in the IDE. Make sure to take the time and read through some of the XNA Game Studio documentation. There is also a Tools folder that contains a couple of tools we will be looking at later. We will discuss the XACT tool in Chapter 7, “Sound and Music,” and the XNA Framework Remote Performance Monitor for Xbox 360 application in Chapter 3, “Performance Considerations.” Go ahead and open the Visual C# Express or Visual Studio IDE.

FIGURE 1.7 XNA Game Studio modifies the Windows Firewall so an Xbox 360 and the PC can talk to each other. It also allows network games created with XNA to communicate.
Creating the Platformer Projects

TIP

Everything in this book works with all the Visual Studio 2008 SKUs as well as Visual C# 2008 Express. From this point on I will simply use the term Visual Studio, regardless of which SKU (including C# Express) is being used.

When you installed XNA Game Studio, it added properties to Visual Studio to allow it to behave differently under certain circumstances. Mainly it added some templates (which we will look at shortly) as well as the ability for Visual Studio to handle content via the XNA Content Pipeline. It also added a way for you to send data to your Xbox 360, as you will see in the next chapter.

Creating the Platformer Projects

With XNA Game Studio opened, once you create a new project, you should see a screen similar to Figure 1.8. Select the Platformer Starter Kit template and feel free to change the name of the project. Click OK to create the project.

FIGURE 1.8 You can see that installing XNA Game Studio added eight more templates to Visual Studio.
Compiling and Running Platformer

At this point you have your software installed and have even created a starter template (created by Microsoft) that you can take for a spin. You need to make sure you can compile the code. To just compile without running, either press Ctrl+Shift+B, press F6, or click Build Solution on the Build menu. The code should have compiled without any issues. You can now press Ctrl+F5 to actually run the game. Have some fun playing the game. Feel free to look around the code and tweak it. Fortunately, you can always re-create the template if something gets really messed up!

**TIP**

When working with one solution file and multiple project files in Visual Studio, you can easily change which devices you are currently building and deploying to by changing the Solutions Platform dropdown box in the toolbar. If you select Mixed Platforms, you will compile for each platform every time. For the project you set as your startup project, XNA Game Studio will try to deploy the game to that device.

**Summary**

In this chapter, I laid the groundwork in getting all the software required installed so you can actually create games on your PC. We even compiled a game and played it. After getting a game session fix, join me in the next chapter, where we will get this project up and running on the Xbox 360!
Index

Numerics

2D, 171
  coordinate systems, 172
text, drawing, 189
2D games
  collision detection, 240-243
enemies, drawing, 235-240
explosions, adding, 246-251
heroes, drawing, 230-235
parallax scrolling, creating, 220-229
setting up game skeletons, 217-220
sounds, adding, 251
switching between states, 229-230
transitions, adding, 243-246
winning and losing, 243
3D games, tunnel vision game, 471
3D lighting, 391
  ambient, 397-398
  bump mapping, 403
  creating custom vertex formats, 391-394
demos, 394-397
directional, 399-403
  normal mapping, 403-407
parallax mapping, 407-410
  relief mapping, 410-414
texture animation, 414-416
3D models
  loading, 114-119
  normal mapping, 406
  relief mapping, 413
texturing, 119-120
3D Objects
  cameras
    creating, 57
    projection, 57-58
view, 58-59
world, 59
checking performance of, 72-76
effects, 63-64
GameComponent objects, creating, 69-72
index buffers, 66-69
mactrices, definition of, 56
textures, 65-66
transformations
definition of, 56
multiple transformations, 57
rotation, 56, 79-80
scaling, 56, 76-78
translation, 56
vectors, overview, 55-56
vertex buffers, 59-63
vertices, overview, 55
80-20 Rule, 34-35
A
acceleration, 326-330
AccelerationDemo program, 327-330
  Draw method, 328
  DrawModel method, 328
  InitializeValues method, 328
  LoadContent method, 328
  PhysicalObject class, 328
  Update method, 329-330
adding
  cameras, 495-499
  crosshairs, 494-495
  explosions (2D games), 246-251
  scoring, 511-512
  skyboxes to tunnel vision game, 472
  sounds, 542-544
  sounds (2D games), 251
  states, 709
  transitions (2D games), 243-246
  turn-based multiplayer games, 652-665
Additive, 213
AddMissile method, tunnel vision game, 477
AddService method, 85
AddValue method, 713
AddVerticesInformation, 157
advanced textures, 3D lighting, 391
  ambient, 397-398
  bump mapping, 403
  creating custom vertex formats, 391-394
demos, 394-397
directional, 399-403
  normal mapping, 403-407
  parallax mapping, 407-410
  relief mapping, 410-414
texture animation, 414-416
AdvancedTexturingDemo, InitializeVertices method, 395
AI algorithms
advanced chase algorithm (TrackPlayer method), 381-382
AIDemo program, 375-380
   Draw method, 377
   DrawModel method, 376
   Initialization method, 377
   KeepWithinBounds method, 379
private member fields, 376
   skybox, 375
   sphere member field, 375
   UpdatePlayer method, 379
evading algorithm (EvadePlayer method), 382
FSMs (finite state machines), 343
   creating, 384-387
random movement algorithm (MoveRandomly method), 382-384
simple chase algorithm (TrackPlayerStraightLine method), 380-381
AIDemo program, 375-380
   Draw method, 377
   DrawModel method, 376
   Initialization method, 377
   KeepWithinBounds method, 379
   Player class, 376
   private member fields, 376
   skybox, 375
   sphere member field, 375
   UpdatePlayer method, 379
evading algorithm (EvadePlayer method), 382
FSMs (finite state machines), creating, 384-387
random movement algorithm (MoveRandomly method), 382-384
simple chase algorithm (TrackPlayerStraightLine method), 380-381
AIState enumerated type, 384
algorithms (AI)
   advanced chase algorithm (TrackPlayer method), 381-382
AIDemo program, 375-380
   Draw method, 377
   DrawModel method, 376
   Initialization method, 377
   KeepWithinBounds method, 379
   Player class, 376
   private member fields, 376
   skybox, 375
   sphere member field, 375
   UpdatePlayer method, 379
   evading algorithm (EvadePlayer method), 382
   FSMS (finite state machines), creating, 384-387
   random movement algorithm (MoveRandomly method), 382-384
   simple chase algorithm (TrackPlayerStraightLine method), 380-381
all function (HLSL), 299
alpha blending, sprites, 173
AlwaysDisplay property, 536
ambient lighting, 397-398
AmbientTexture.fx file, 307-310, 399
animation
   cel animation, 191
   textures, 414-416
animationKey, 641
ANTS Profiler, 35
any function (HLSL), 299
application data, passing to GPU, 306-307
ApplyFriction method, CollisionDemo program, 340
ApplySmoothing method, 714
architecture, networks, 549
   clients/servers, 549
   hybrid, 550-551
   peer-to-peer, 550
   System Link versus LIVE, 552
artwork, Xbox LIVE Community Games (XBLCG), 720
asin function (HLSL), 299
assembly language, 292
AssemblyDescription attribute (games), 26
AssemblyInfo.cs file, 25
AssemblyTitle attribute (games), 25
associating Xbox 360 with computer, 21-23
atan function (HLSL), 299
atan2 function (HLSL), 299
attributes
  AssemblyDescription, 26
  AssemblyTitle, 25
  Guid, 26
audio, Xbox LIVE Community Games (XBLCG), 722

B
Back button, 715
bandwidth constraints, 556-561
BaseGameState class, 349, 356-357
BaseMenuState class, 595
BaseMenuState.cs, 591
BasicEffect class, 63
BeginDraw method, 569
benchmarks
  creating, 35-37
  micro-benchmark testing, 41-42, 46, 50-51
  CheckPerformance class, 42
  Program.cs file, 46
  time measurement relationships, 50-51
bitmap fonts, creating, 188
BitShifter, 558
blend modes
  Additive, 213
  sprite batch demos, 179-183
  sprites, 173
blending mode, RotateAndScaleDemo, 203
blocks, try/catch, 616
blurring, 319
bubbles, creating with particle effects, 461-462
buffers
  index buffers, 66-69
  vertex buffers, 59-63
bump mapping, 403
C
calculations, 331
Camera class, InitializeCamera method, 87
Camera object, 498
cameraReference field, 89
cameras
  creating, 57
  first person cameras, 102-104
  game-specific cameras, creating for tunnel vision, 495-499
  input devices
    game pads, 93-97
    keyboards, 89-93
    mouse, 97-98
  projection, 57-58
  render targets, configuring, 502
  spin rate, 91-93
  split screens, 104-108
  stationary cameras, 98-100
  cull mode, 99
    Update method, 99-100
  view, 58-59
  viewports, 709
  world, 59
CancelMenu method, 595, 620
capacity, particle systems, 440
Card.cs, 634
ceil function (HLSL), 299
cel animation, 191-201
  AddAnimation method, 197
  Draw method, 198
CelAnimationManager, 641
CeRange values, 641
Cg language, 292
chalk effect, 321
ChangeState method, 354
chase algorithms
  TrackPlayer method, 381-382
  TrackPlayerStraightLine method, 380-381
CheckCards method, 646-648, 655
CheckCollisions method, 487, 544
CheckPerformance class, 42
clamp function (HLSL), 299
classes
  BaseGameState, 356
  BaseMenuState, 595
  BasicEffect, 63
  Camera, InitializeCamera method, 87
  CheckPerformance, 42
  DrawableGameComponent, 347
  FadingState, 368-369
  FirstPersonCamera, Update method, 102-104
  GameState, 348-351
  GameStateManager, 352-354
  InputHandler, 85, 598
  LostGameState, 367
  OptionsMenuState, 360-361
particles, creating, 436-441
PhysicalObject, 328, 334
Player (AIDemo), 376
PlayingState, 365-366, 543, 715
StartLevelState, 361-363
StartMenuState, 358-360
System.Diagnostics, 50

How can we make this index more useful? Email us at indexes@samspublishing.com
private member fields, 334
Update method, 336
conservation of momentum, 332
final velocities, 333
impulse, 332
kinetic energy, 332-333
momentum, 331-332
overview, 331
color
fading to, 204-206
switching RGB values, 318
colorful effects, creating with particle effects, 467-468
compiling
Spacewar Windows Start Kit project, 18
tunnel vision game, 472
components
GameStateManager, 589
InputHandler, 589
compression, 558-561
ConcentrationCards.png texture, 639
conditions (HLSL), 304
configuring
game states, 471-472, 605-630
games
adding cameras, 495-499
adding crosshairs, 494-495
logic, 493-494
HUD, 510-511
level timers, 508-509
menus, 590-591
refactoring states, 591-605
network demos, 565-577
Zunes, 578-582
options menus, updating, 536-540
Particle classes, 436-441
particle system demos, 455-458
particle system effects, 458-468
particle system engines, 443-452
particle systems, updating, 541-542
point sprite effect files, 452-455
radar, 501-504
real-time multiplayer games
creating, 671
functionality, 672-688
networks, 689-709
prediction/smoothing, 709-716
templates, 672
scoring, 511-512
creating high score screens, 532-537
tracking, 512-523
sound, 542-544
start menus, updating, 528-533
templates, 583-590
title screens, updating, 525-529
tunnels, 504-508
turn-based multiplayer games, 631-632
adding, 652-665
creating, 669-670
functionality, 633-651
players joining/leaving, 665-668
templates, 632
VertexPointSprite, 442-443
Xbox LIVE Community Games (XBLCG), 719
artwork, 720
audio, 722
controls, 720-721
displays, 719
marketing, 725-726
menus, 721
networks, 723
profiles, 723
rich presence data, 724
storage, 724
Trial Mode, 721-722
configuring states, updating, 539-541
connecting Xbox 360 to computer, 21-23
connections, network architecture, 549
  clients/servers, 549
  hybrid, 550-551
  peer-to-peer, 550
  System Link versus LIVE, 552
conservation of kinetic energy, 333
conservation of momentum, 332
constraints, bandwidth, 556-561
contact force, 330
ContainsState method, 354
content, multiple projects, 578
Content Compiler, 113
Content DOM, overview, 113
Content Importer, overview, 113
Content Manager, overview, 114
Content Pipeline
  Content Compiler, 113
  Content DOM, 113
  Content Importer, 113
  Content Manager, 114
  custom importers, creating, 164
  loading 3D models, 114-119
  texturing 3D models, 119-120
Content Pipeline extensions, debugging, 167
defaults, Xbox LIVE Community Games (XBLCG), 720-721
coordinate systems
  2D, 172
    right-handed coordinate systems, 60
cos function (HLSL), 299
cosh function (HLSL), 299
Covey, Stephen, 33
Create method, 615
CreateEnemies method, 683
CreateLookAt method, 58
CreateParticle method, 448
CreatePerspectiveFieldOfView method, 57
CreateScale method, 78-80
CreateTexture method, 430, 527
CreateTranslation method, 77, 313
CreateVisualizationTexture method, 268
Creators Club, subscribing to, 20-21
Creators Club website, 728
cross function (HLSL), 299
cross-fade (dissolve), 422-423
crosshairs, creating for tunnel vision game, 494-495
cull mode, stationary cameras, 99
currentSmoothing value, 714
custom Content Pipeline importer, creating, 164
customizing
  menus, 590-591
    refactoring states, 591-605
  vertext formats, 391-394

d
D
D3DCOLORtoUBYTE4 function (HLSL), 299
ddx function (HLSL), 299
ddy function (HLSL), 299
Debug menu commands, Deploy Solution command, 23
debugging
  Content Pipeline extensions, 167
  Xbox 360, 25
degrees function (HLSL), 299
deleting players, 668
demo program (Xbox 360), 25-28

How can we make this index more useful? Email us at indexes@samspublishing.com
demos

3D lighting, 394-397
GameStateManagementSample, 532
network
creating, 565-577
Zunes, 578-582
particle systems, 455-458
Deploy Solution command (Debug menu), 23
deploying
Xbox 360, 23-24
Zune, 258
depth maps, 407
design
object-oriented design, 344-345
real-time multiplayer games
creating, 671
functionality, 672-688
networks, 689-709
prediction/smoothing, 709-716
templates, 672
turn-based multiplayer games, 631-632
adding, 652-665
creating, 669-670
functionality, 633-651
players joining/leaving, 665-668
templates, 632
determining function (HLSL), 299
directional lighting, 399-403
directional wipes, 423-428
DirectionalTexture.fx file, 401-402
DirectX, 7-8
FFFF (Fixed Function Pipeline), 291
installing, 15
DirectX runtime, installing, 8-15
DisplayAlbumArtDemo, Zune, 255-258
displays, Xbox LIVE Community Games (XBLCG), 719
dissolve, 422-423
distance function (HLSL), 299
distributing games, 24-25
DOS, 8
dot function (HLSL), 299
Draw method, 27, 77, 328, 406, 420, 709
 AIDemo program, 377
cel animation, 198
GameStateDemo project, 373
particle system demo, 457-458
particle system engines, 449-450
radar, creating, 502
split screens, 108
tunnel vision game, 479-480, 488, 493
DrawableGameComponent class, 347
DrawIndexedPrimitives,
 DrawUserIndexedPrimitives versus, 74-76
drawing
2D text, 189
enemies (2D games), 235-240
heroes (2D games), 230-235
multiple sprites from one texture demo, 177-179
DrawMessage method, 568
DrawModel method, 117-119, 328, 421
AIDemo program, 376
DrawRectangle method, 76
tunnels, creating, 507
DrawScene method, split screens, 108
DrawUserIndexedPrimitives,
 DrawIndexedPrimitives versus, 74-76
dual platform programming, 28-31
dxwebsetup.exe file, 15
E

E ratings, 724
effects, 63-64
  blurred images, 319
cel animation, 199
chalk, 321
embossed images, 319
explosions, creating, 214-215
fading to color, 204-206
fire, creating, 206-214
grayscale, 320
negative images, 318
NightTime, 316-317
particle system demos, 455-458
particle systems, creating, 458-468
postprocessing, 316-317
rotation, 203
scaling, 203
sharpened images, 318-319
switching RGB values, 318
wavy, 321
elastic collisions, 332
embossed images, 319
enemies, drawing (2D games), 235-240
Enemy.cs, 673
EnemyManager drawable game component, 503
Entries collection, 595
enumerations
  AIState, 384
  managing game state with, 346
  PrimitiveType, 64
EvadePlayer method, 382, 682, 706
evading algorithm (EvadePlayer method), 382
exiting Spacewar project, 24
exp function (HLSL), 299
exp2 function (HLSL), 299
explosions
  adding in 2D games, 246-251
  creating, 214-215
extensions, debugging Content Pipeline extensions, 167
extern modifier (HLSL), 295

F

faceforward function (HLSL), 300
fades, cross-fades (dissolve), 422-423
fading to color, 204-206
FadingState class, 368-369
FFP (Fixed Function Pipeline), 291
field force, 330
fields, storedTime, 509
files
  AmbientTexture.fx, 307-310, 399
  AssemblyInfo.cs, 25
  DirectionalTexture.fx, 401-402
  GameStateInterfaces.cs, 355-356
  interfaces.cs, 349
  NormalMapping.fx, 403
  ParallaxMapping.fx, 407-409
  point sprite, formatting, 452-455
  point sprites, 452-455
  ReliefMapping.fx, 411-413
  .xnb file extension, 114
final velocities, 333
  CollisionDemo program, 338
FindColorIndex method, 210
FindWinner method, 661-662
FinishSavingHighScore method, 518

How can we make this index more useful? Email us at indexes@samspublishing.com
finite state machines (FSMs), 343
  AlState enumerated type, 384
  conditions, 386-387
  constants, 385-386
  creating, 384-387
  properties, 384
  switch statement, 385
fire, 428-434
  creating, 206-214
FireDemo, Zune, 260-262
first person cameras, 102-104
FirstPersonCamera class, Update method, 102-104
Fixed Function Pipeline (FFP), 291
flags, Game.IsActive, 722
floor function (HLSL), 300
fmod function (HLSL), 300
fonts
  bitmap fonts, creating, 188
  TrueType fonts, 187
  importing, 188
force, 330-331
formatting
  game states, 471-472, 605-630
  games
    adding cameras, 495-499
    adding crosshairs, 494-495
    logic, 491-494
  HUD, 510-511
  level timers, 508-509
  menus, 590-591
    refactoring states, 591-605
  network demos, 565-577
  Zunes, 578-582
  options menus, updating, 536-540
Particle classes, 436-441
particle system demos, 455-458
particle system effects, 458-468
particle system engines, 443-452
particle systems, updating, 541-542
point sprite effect files, 452-455
radar, 501-504
real-time multiplayer games
  creating, 671
  functionality, 672-688
  networks, 689-709
  prediction/smoothing, 709-716
  templates, 672
scoring, 511-512
  creating high score screens, 532-537
  tracking, 512-523
sound, 542-544
start menus, updating, 528-533
states, updating, 539-541
templates, 583-590
title screens, updating, 525-529
tunnels, 504-508
turn-based multiplayer games, 631-632
  adding, 652-665
  creating, 669-670
  functionality, 633-651
  players joining/leaving, 665-668
  templates, 632
VertexPointSprite, 442-443
Xbox LIVE Community Games (XBLCG), 719
  artwork, 720
  audio, 722
  controls, 720-721
  displays, 719
  marketing, 725-726
  menus, 721
  networks, 723
  profiles, 723
  rich presence data, 724
storage, 724
Trial Mode, 721-722

**formulas**
- acceleration, 326
- conservation of kinetic energy, 333
- conservation of momentum, 332
- final velocities, 333
- force, 331
- impulse, 332
- kinetic energy, 332
- momentum, 331
- velocity, 326

**frac function (HLSL), 300**
**framesBetweenPackets variable, 713**
**frexp function (HLSL), 300**
**FSMs (finite state machines), 343-344**
  - creating, 384-387
    - AIState enumerated type, 384
    - conditions, 386-387
    - constants, 385-386
    - properties, 384
    - switch statement, 385

**functionality**
- real-time multiplayer games, 672-688
- turn-based multiplayer games, 633-651

**functions**
- HLSL, 298-304
- pixelShader, 454

**fwidth function (HLSL), 300**

**G**

**game components**, number of, 84
**game components**, creating, 69-72
**Game Information page**, 730

**game logic, tunnel vision game, 473-486, 490, 494**
- AddEnemy method, 483
- AddMissile method, 477
- CheckCollisions method, 487
- CollisionDetection method, 478
- Draw method, 479-480, 488, 493
- InitializeMissile method, 478
- Load method, 476
- LoadContent method, 476-482
- PhysicalObject class, 474
- StartLevelState, 491
- StateChanged method, 488-491
- UnloadContent method, 476-482
- Update method, 493
- UpdateEnemies method, 487
- VertexDisplacement.fx, 489

**game loops, definition of, 34**
**Game object, 84**
**game pads, 93-97**
  - determining state of, 93-95
  - SetVibration method, 96-97

**game services**
  - creating, 83-86
  - GraphicsDevice, 84

**game skeletons, setting up, 217-220**
**game state, 345**
**game-specific cameras, creating for tunnel vision game, 495-499**
**Game.IsActive flag, 722**
**Game1 method, 372-373**
**Game1.cs code files, 612**
**GameComponent objects, creating, 69-72**
**GameManager.PushState method, 602**
**Gamer.SignedInGamers, 723**
**GamerJoinedEventHandler method, 569**
**GamerLeftEventHandler method, 749**

How can we make this index more useful? Email us at indexes@samspublishing.com
GamerServicesComponent, 589

**games, 24**
- 3D games, 471
- game components, number of, 84
- game services
  - creating, 83-86
  - GraphicsDevice, 84
- game state, 345
- GameComponent objects, creating, 69-72
- Grand Theft Auto III, 345
- improving, 544-545
- network requirements, 552
  - hardware, 554
  - membership, 553-554
- optimizing to run on Zune, 283-286
- players, inviting to, 563-564
- porting to run on Zune, 273-280
- postprocessing
  - blurring images, 319
  - chalk effect, 321
  - embossed images, 319
  - game code, 313-316
  - grayscale images, 320
  - negative images, 318
  - NightTime effect, 316-317
  - overview, 313
  - sharpening images, 318-319
  - switching RGB values, 318
  - wavy effect, 321
- real-time multiplayer
  - creating, 671
  - functionality, 672-688
  - networks, 689-709
  - prediction/smoothing, 709-716
  - templates, 672
- reviewing, 727-730

running in landscape mode (Zune), 280-283
selling, 727-737
Spacewar, 24
split screens, 104-108
states, formatting, 605-630
submitting, 732-735
turn-based multiplayers
  - adding, 652-665
  - creating, 669-670
  - design, 631-632
  - functionality, 633-651
  - players joining/leaving, 665-668
  - templates, 632

Games Catalog, 728

games states, creating in tunnel vision game, 471

GameState class, 348-351

GameState.PlayerIndexInControl property, 601

GameStateDemo project, 348

- BaseGameState class, 349, 356-357
- ChangeState method, 354
- ContainsState method, 354
- Draw method, 373
- FadingState class, 368-369
- Game1 constructor, 373
- GameState class, 348-351
- GameStateInterfaces.cs file, 355-356
- GameStateManager class, 351-352
- IGameState interface, 348
- interfaces.cs file, 349
- LostGameState class, 367
- OptionsMenuState class, 360-361
- PlayingState class, 365-366
- PushState method, 353-354
- RemoveState method, 353
- StartLevelState class, 361-363
- StartMenuState class, 358-360
TitleIntroState class, 357-358
WonGameState class, 367
YesNoDialogState class, 363
GameStateInterfaces.cs file, 355-356
GameStateManagementSample demo, 532
GameManager, 600
GameManager class, 351-352
ChangeState method, 354
ContainsState method, 354
PushState method, 353
GameManager component, 589
garbage collection, 39-41
GetIndexFromColor method, 699
GetNextPlayer method, 648, 668
GetService method, 85
GetState method
game pads, 93-94
keyboards, 89
GetTexture method, 503
GPU, passing application data to, 306-307
Grand Theft Auto III, 345
graphics cards, 200
GraphicsDevice game service, 84
gravity, 330
gray scale images, 320
Guid attribute (games), 26
Guide.IsTrialMode property, 721
Guide.ShowMarketplace method, 722

H
HandleInput method, 646, 680
hardware, requirements, 554
hardware abstraction layer, 8
hardware emulation layer, 8
headers, packets, 557
height maps, 407
HEL (hardware emulation layer), 8
HelpState.cs, 603
heroes, drawing (2D games), 230-235
High Level Shader Language, 292
high score screens, creating, 532-537
HLSL (High Level Shader Language), 292
AmbientTexture.fx demo, 307-310
application data, passing to GPU, 306-307
intrinsic functions, 298-304
loops, 304
modifiers, 295
overview, 291-292
passes, 306
pixel shaders, 305
postprocessing
blurring images, 319
cchalk effect, 321
effect code, 316-317
eembossed images, 319
ggame code, 313-316
grayscale images, 320
negative images, 318
NightTime effect, 316-317
overview, 313
sharpening images, 318-319
switching RGB values, 318
wavy effect, 321
semantics, 296-298
shader process flow, 293-294
structs, 296-298
techniques, 305-306
variables, types, 294-295
vertex displacement, 311-313

How can we make this index more useful? Email us at indexes@samspublishing.com
vertex shaders, 304-305
   compared to pixel shaders, 291
HLSLFireDemo project, 428-434
Hoare, Sir Tony, 34
HookSessionEvents method, 626
HUD, creating, 510-511
hybrid networks, 550-551

IGameState interface, 348
IGameStateManager interface, 600
images
   blurring, 319
   chalk effect, 321
   color, switching RGB values, 318
   embossed effect, 319
   grayscale effect, 320
   negative images, creating, 318
   sharpening, 318-319
   wavy effect, 321
implementing particle system engines, 443-452
importing TrueType fonts, 188
improving games, 544-545
impulse, 332
index buffers, 66-69
indexes
   index buffers, 66-69
   players, 720
inelastic collisions, 332
INetworkMenuState interface, 612
Initialization method, AlDemo program, 377
Initialize method, 543, 639
   CollisionDemo program, 334
   Particle class, 438-439
InitializeCamera method, 57, 87
InitializeDefaultHighScore method, 521
InitializeMissile method, tunnel vision game, 478
InitializeValues method, 328
   CollisionDemo program, 335
InitializeVertices method,
   AdvancedTexturingDemo, 395
input devices
   game pads, 93-97
   determining state of, 93-95
   SetVibration method, 96-97
   keyboards, 89-93
   camera reference direction, 89
   determining state of, 89-91
   spin rate, 91-93
   mouse, 97-98
   determining state of, 97-98
input handlers, updating, 134-139
Input namespace, 85
InputHandler class, 85, 598
InputHandler component, 589
InputHandler interface, 85
installing
   DirectX 9 runtime, 15
   DirectX runtime, 8-15
   XNA Game Studio Express, 15-17
interfaces
   IGameState, 348
   IGameStateManager, 600
   IInputHandler, 85
   INetworkMenuState, 612
   ISessionListState, 616
   ISessionLobbyState, 620
interfaces.cs file, 349
interpolating, 715
intrinsic functions (HLSL), 298-304
inviting players to games, 563-564
IsActive condition, 579
IISessionListState interface, 616
IISessionLobbyState interface, 620
isfinite function (HLSL), 300
IsGameFinished property, 649
isinf function (HLSL), 300
isnan function (HLSL), 300

J
joining
games, 665-668
inviting players to, 563-564
XNA Creators Club, 20-21
JoinInProgressGame method, 628

K
KeepWithinBounds method, 677
AIDemo program, 379
keyboards, 89-93
camera reference direction, 89
determining state of, 89-91
spin rate, 91-93
keywords
matrix, 295
ref, 39
vector, 294
kinematics
acceleration, 326-330
force, 330-331
overview, 325-326
velocity, 326
kinetic energy, 332-333

L
landscape mode, running games in Zune, 280-283
LANs (local area networks), 552
laser scanners, creating with particle effects, 464-465
laser shields, creating with particle effects, 462-464
latency, 554-555, 711
layouts, menus
formatting, 590-591
refactoring states, 591-605
ldexp function (HLSL), 300
leaving games, 665-668
length function (HLSL), 300
lerp function (HLSL), 300
level timers, creating, 508-509
libraries, 86-88
lighting
3D, 391
ambient, 397-398
bump mapping, 403
creating custom vertex formats, 391-394
demos, 394-397
directional, 399-403
normal mapping, 403-407
parallax mapping, 407-410
relief mapping, 410-414
texture animation, 414-416
AdvancedTexturingDemo, InitializeVertices method, 395
limitations of Zune, 286-287
LineList member (PrimitiveType enumeration), 64
LineStrip member (PrimitiveType enumeration), 64
lit function (HLSL), 300
LIVE, 552

How can we make this index more useful? Email us at indexes@samspublishing.com
Load method, tunnel vision game, 476
Load3DObject project, 113-119
LoadContent method, 418, 476-482, 506, 532, 640
  particle system demo, 456
tunnel vision game, 476-482
loading
  3D models, 113-119
  screen demos, sprite batches, 176-177
local ad-hoc Wi-Fi gaming, 578-582
local area networks, 552
log function (HLSL), 300
log10 function (HLSL), 300
log2 function (HLSL), 300
logic, games
  adding cameras, 495-499
  adding crosshairs, 494-495
  creating, 473-491, 493-494
loops
  definition of, 34
    HLSL, 304
losing 2D games, 243
loss, packet, 555-556
LostGameState class, 367
low function (HLSL), 301
managing game state
  enumerated types, 346
GameStateDemo project, 348
  BaseGameState class, 349, 356-357
ChangeState method, 354
ContainsState method, 354
Draw method, 373
FadingState class, 368-369
Game1 constructor, 373
GameState class, 348-351
GameStateInterfaces.cs file, 355-356
GameStateManager class, 351-352
IGameState interface, 348
interfaces.cs file, 349
LostGameState class, 367
OptionsMenuState class, 360-361
PlayingState class, 365-366
PushState method, 353-354
RemoveState method, 353
StartLevelState class, 361-363
StartMenuState class, 358-360
TitleIntroState class, 357-358
WonGameState class, 367
YesNoDialogState class, 363
goals of, 345-346
stacks, 346-348
mapping
  bump, 403
  normal, 403-407
  normal mapping, 406
  parallax, 407-410
  relief, 410-414
  relief mapping, 413
marketing Xbox LIVE Community Games (XBLCG), 725-726
mass, 331
matrices
  defining, 295
definition of, 56
matrix keyword, 295
max function (HLSL), 300
measuring performance
  80-20 Rule, 34-35
  benchmarks, creating, 35-37
overview, 33-34
profiler tools, 35
on Xbox 360, 37-39

**membership in XNA Creators Club,** purchasing, 20

**membership requirements,** 553-554

**memory,** 39
   - buffers
      - index buffers, 66-69
      - vertex buffers, 59-63
   - managing, 39

**menus**
   - configuring, 590-591
     - refactoring states, 591-605
   - options, updating, 536-540
   - start, updating, 528-533
   - Xbox LIVE Community Games (XBLCG), 721

**MenuSelected method,** 595, 620

**MessageDialogState,** 611

**MessageDialogState.cs,** 609

**MessageType enums,** 695

**methods**
   - AddService, 85
   - AddValue, 713
   - ApplyFriction, CollisionDemo program, 340
   - ApplySmoothing, 714
   - BeginDraw, 569
   - CancelMenu, 595, 620
   - ChangeState, 354
   - CheckCards, 646-648, 655
   - CheckCollisions, 544
   - ContainsState, 354
   - Create, 615
   - CreateEnemies, 683
   - CreateLookAt, 58
   - CreatePerspectiveFieldOfView, 57
   - CreateScale, 78-80
   - CreateTexture, 430, 527
   - CreateTranslation, 77, 313
   - Draw, 27, 77, 328, 406, 420, 709
     - AIManager program, 377
     - GameStateDemo project, 373
     - split screens, 108
   - DrawMessage, 568
   - DrawModel, 117-119, 328, 421
     - AIManager program, 376
   - DrawRectangle, 76
   - DrawScene, split screens, 108
   - EvadePlayer, 382, 682, 706
   - FindWinner, 661-662
   - FinishSavingHighScore, 518
   - Game1, 372-373
   - GameManager.PushState, 602
   - GamerJoinedEventHandler, 569
   - GamerLeftEventHandler, 627
   - GetIndexFromColor, 699
   - GetNextPlayer, 648, 668
   - GetService, 85
   - GameState
      - game pads, 93-94
      - keyboards, 89
   - GetTexture, 503
   - Guide.ShowMarketplace, 722
   - HandleInput, 646, 680
   - HookSessionEvents, 626
   - Initialization, AIManager program, 377
   - Initialize, 543, 639
     - CollisionDemo program, 334
   - InitializeCamera, 57, 87
   - InitializeDefaultHighScore, 521
   - InitializeValues, 328
     - CollisionDemo program, 335
   - InitializeVertices,
     AdvancedTexturingDemo, 395

*How can we make this index more useful? Email us at indexes@samspublishing.com*
JoinInProgressGame, 628
KeepWithinBounds, 677
AI Demo program, 379
LoadContent, 418, 476-482, 506, 532, 640
MenuSelected, 595, 620
MoveRandomly, 382-384, 680
NetworkSession.Create, 612
NetworkSession.FindGamerByld, 667
NetworkSession.Update, 653
OnInviteAccepted, 563
PausedState, 601
PlayerLeft, 628, 665
PowerCurve, 498
PushState, 353-354
ReadIncomingPackets, 653, 699
ReadInputs, 573
ReadNetworkPacket, 710
RemoveState, 353
SessionEndedEventHandler, 570, 626
SetCameraProperties, 686
SetPlayerColor, 698
SetPresenceInformation, 642, 684
SetupGame, 509, 642, 684
SetValue, 312
SetVibration, 96-97
ShuffleCards, 642
StartGame, 588
StartLevel, 509
StartLocalMultiplayerGame, 612
StateChanged, 347, 619, 626
TrackPlayer, 381-382, 682
TrackPlayerStraightLine, 380-381
UnloadContent, 28, 418, 431, 476-482, 502, 533
Update, 406, 419, 509
AccelerationDemo program, 329-330
CollisionDemo program, 336
first person cameras, 102-104
stationary cameras, 99-100
UpdateFade, 422
UpdateInput, 419
UpdateLocal, 710
UpdateLocalGamer, 572, 653, 691
UpdateNetworkSession, 573, 690, 699
UpdatePlayer, AI Demo program, 379
UpdateRemote, 714
UpdateState, 678
UpdateWipe, 426
VertexPositionNormalTexture, 62
virtual methods, sealing, 51
WasButtonPressed, 599
WasPressed, 598
WriteNetworkPacket, 703
micro-benchmark testing, 41-42, 46, 50-51
CheckPerformance class, 42
Program.cs file, 46
time measurement relationships, 50-51
Microsoft Permissive License (Ms-PL), 711
min function (HLSL), 300
missileManager.CheckCollision condition, 544
mod function (HLSL), 301
models, 3D models
loading, 113-119
texturing, 119-120
modifiers (HLSL), 295
momentum, 331-332
conservation of, 332
monitoring performance on Xbox 360, 37-38
motion dynamics, 325
mouse input devices, 97-98
determining state of, 97-98
MoveRandomly method, 382-384, 680
moveSpeed, 481
Ms-PL (Microsoft Permissive License), 711
mul function (HLSL), 301
MultiplayerMenuState, 609
MultiplayerMenuState.cs, 605
multiple content projects, 578

N
namespaces, Input, 85
negative images, creating, 318
.NET Compact Framework, 31-32
garbage collection, 40-41
.NET Framework, garbage collection, 39-40
network demos
creating, 565-577
Zunes, 578-582
NetworkGameTemplate, 583-590
NetworkMenuState.cs, 613
networks
architecture, 549-552
bandwidth constraints, 556-561
game states, 605-630
latency, 554-555
packet loss, 555-556
prediction, 562-563
real-time multiplayer games, 689-709
requirements, 552-554
smoothing, 562-563
Xbox LIVE Community Games (XBLCG), 723
NetworkSession.Create method, 612
NetworkSession.FindGamerById method, 667
NetworkSession.Update method, 653
NetworkSessionEndedEventArgs parameter, 570
NetworkSessionType, 615
Newton’s First Law of Motion, 331
Newton’s Second Law of Motion, 331
Newton’s Third Law of Motion, 332-333
NightTime effect, 316-317
noise function (HLSL), 301
normal mapping, 403-407
normalize function (HLSL), 301
NormalMapping.fx, 403
normals, 399
NProf, 35

O
object-oriented design, 344-345
objects
Camera, 498
collisions
coefficient of restitution, 333
CollisionDemo program, 334-341
conservation of kinetic energy, 333
conservation of momentum, 332
final velocities, 333
impulse, 332
kinetic energy, 332-333
momentum, 331-332
overview, 331
Game, 84
motion dynamics, 325
object-oriented design, 344-345
objects (3D), 55
OnInviteAccepted method, 563
optimizing
collision detection, 241
games to run on Zune, 283-286

How can we make this index more useful? Email us at indexes@samspublishing.com
optimizing performance
  collections, 51-52
  micro-benchmark testing, 41-42, 46, 50-51
    CheckPerformance class, 42
    Program.cs file, 46
  time measurement relationships, 50-51
    overview, 41
    virtual methods, sealing, 51

options menus
  refactoring states, 591-605
  updating, 536-540

OptionsMenuState class, 360-361, 536

P

packet loss, 555-556
packets
  bandwidth constraints, 557
  compression, 558-561
parallax mapping, 407-410
parallax scrolling, creating, 220-229
ParallaxMapping.fx file, 407-409
parameters
  NetworkSessionEndedEventArgs, 570
  pipeline processor, 165-167

Pareto, Vilfredo, 34
Particle class, creating, 436-437
  capacity, 440
  Initialize method, 438-439
  ParticleSystemSettings.cs, 439-440
  randomness, 440-442
  Update method, 437

particle effects, creating, 458-468
  bubbles, 461-462
  colorful effects, 467-468
  laser scanners, 464-465
  laser shields, 462-464
  poisonous gas, 465-467
  rain, 458-461

particle system demo, 455-458
  Draw method, 457-458
  LoadContent method, 456

particle system engines, creating, 443-452
  CreateParticle method, 448
  Draw method, 449-450
  PopulatePointSprites, 451
  SetBlendModes method, 450

particle systems, 435-436, 541-542
  updating, 541-542
  VertexPointSprite, creating, 442-443

particles, VertexPointSprite, creating, 442-443
ParticleSystemSettings.cs, 439-440
passes (HLSL), 306
passing by reference, 39
passing by value, 38

PausedState method, 601

Peer Review Terms and Conditions page, 729
peer reviews, games, 727-730
peer-to-peer networks, 550

performance
  of 3D objects, checking, 72-76
  garbage collection, 39-41
  measuring
    80-20 Rule, 34-35
    benchmarks, 35-37
    overview, 33-34
    profiler tools, 35
    on Xbox 360, 37-38
  memory management, 39
  optimizing
    collections, 51-52
    micro-benchmark testing, 41-42, 46, 50-51
physicalObject class, 328, 334

physics

collisions

coefficient of restitution, 333
CollisionDemo program, 334-341
conservation of kinetic energy, 333
conservation of momentum, 332
final velocities, 333
impulse, 332
kinetic energy, 332-333
momentum, 331-332
overview, 331

kinematics

acceleration, 326-330
force, 330-331
overview, 325-326
velocity, 326
mass, 331
Newton’s First Law of Motion, 331
Newton’s Second Law of Motion, 331
Newton’s Third Law of Motion, 332-333

pipeline processor parameters, 165-167

pixel input semantics, 296
pixel output semantics, 296
pixel shaders, 305

PointList member (PrimitiveType enumeration), 64

poisonous gas, creating with particle effects, 465-467

PopulatePointSprites method, particle system engines, 451

porting games to run in Zune, 273-280

Position property, 709

postprocessing

blurring images, 319
chalk effect, 321
defined, 311
effect code, 316-317
embossed images, 319
game code, 313-316
grayscale images, 320
negative images, 318
NightTime effect, 316-317
overview, 313
sharpening images, 318-319
switching RGB values, 318
wavy effect, 321

PowerCurve helper method, 498

prediction, 562-563

real-time multiplayer games, 709-716

Premium members, 728

prevHighlightedCard field, 634

primitiveType enumeration, 64

profiler tools, 35

How can we make this index more useful? Email us at indexes@samspublishing.com
profiles, Xbox LIVE Community Games (XBLCG), 723
programming for dual platforms, 28-31
progress bars, sprite batches, 183-187
projection, 57-58
projects
  AdvancedTexturingDemo, InitializeVertices method, 395
  AI Demo, 375-380
    Draw method, 377
    DrawModel method, 376
    Initialization method, 377
    KeepWithinBounds method, 379
  Player class, 376
  private member fields, 376
  skybox, 375
  sphere member field, 375
  UpdatePlayer method, 379
demo program (Xbox 360), 25-28
fire, 428-434
GameStateDemo, 348
  BaseGameState class, 349, 356-357
  ChangeState method, 354
  ContainsState method, 354
  Draw method, 373
  FadingState class, 368-369
  GameI constructor, 373
  GameState class, 348-351
  GameStateInterfaces.cs file, 355-356
  GameStateManager class, 351-352
  IGameState interface, 348
  interfaces.cs file, 349
  LostGameState class, 367
  OptionsMenuState class, 360-361
  PlayingState class, 365-366
  PushState method, 353-354
  RemoveState method, 353
  StartLevelState class, 361, 363
  StartMenuState class, 358-360
  TitleIntroState class, 357-358
  WonGameState class, 367
  YesNoDialogState class, 363
libraries, creating, 86-88
Load3DObject, 114-119
multiple content, 578
NetworkGameTemplate, 583-590
programming for dual platforms, 28-31
Spacewar project
  creating, 19
  deploying to Xbox 360, 24
  distributing, 24-25
  exiting, 24
Spacewar Windows Start Kit project
  compiling and running, 18
  creating, 17
TransitionsDemo, 417-422
  cross-fade (dissolve), 422-423
  directional wipes, 423-428
XNA PerformanceChecker, 42-46
properties
  AlwaysDisplay, 536
  GameState.PlayerIndexInControl, 601
  Guide.IsTrialMode, 721
  IsGameFinished, 649
  Position, 709
  ReadyToCheckCards, 646
  Velocity, 709
purchasing XNA Creators Club memberships, 20-21
PushState method, 353-354
Quick Match feature, 723

radar
  creating, 501-503
  Draw method, 502
radian function (HLSL), 301
rain, creating with particle effects, 458-461
random movement algorithm (MoveRandomly method), 382-384
randomness, particle systems, 440-442
RBG values, switching, 318
ReadIncomingPackets method, 653, 699
ReadInputs method, 573
ReadNetworkPacket method, 710
ReadyToCheckCards property, 646
real-time multiplayer games
  creating, 671
  functionality, 672-688
  networks, 689-709
  prediction/smoothing, 709-716
  templates, 672
ref keyword, 39
refactoring menu states, 591-605
reference, passing by, 39
referencing libraries, 88
reflect function (HLSL), 301
reflief mapping, 410-414
ReliefMapping.fx, 411
ReliefMapping.fx file, 413
reminders, storing, 447
RemoveState method, 353
removing players, 668
render targets, configuring, 502
requirements, networks, 552
  hardware, 554
  membership, 553-554
restitution, coefficient of, 333
reviewing games, 727-730
rich presence data, Xbox LIVE Community Games (XBLCG), 724
right-handed coordinate systems, 60
RollingAverage.cs, 711
RotateAndScaleDemo, 201-202
  blending mode example, 203
rotating objects
  definition of, 56
  example, 79-80
rotation, RotateAndScaleDemo, 201-202
  blending mode example, 203
round function (HLSL), 301
rsqrt function (HLSL), 301
running Spacewar Windows Start Kit project, 18

saturate function (HLSL), 301
save state modes, sprites, 174-175
scaling
  RotateAndScaleDemo, 201-202
  RotateAndScaleDemo blending mode example, 203
  sprites, 173
scaling objects
  definition of, 56
  example, 76-78
scoring
  adding to tunnel vision game, 511-512
  high score screens, creating, 532-537

How can we make this index more useful? Email us at indexes@samspublishing.com
tracking high scores, tunnel vision game, 512-516, 523
scrolling, parallax scrolling, 220-229
sealing virtual methods, 51
selling games, 727-737
semantics (HLSL), 296-298
servers, 549
services (game)
  creating, 83-86
  GraphicsDevice, 84
SessionEndedEventHandler method, 570, 626
SessionListState.cs, 616
SessionLobbyState.cs, 620
SetBlendModes method, particle system engines, 450
SetCameraProperties method, 686
SetPlayerColor method, 698
SetPresenceInformation method, 642, 684
SetupGame method, 509, 642, 684
SetValue method, 312
SetVibration method, 96-97
The 7 Habits of Highly Effective People, 33
shaders
  AmbientTexture.fx demo, 307-310
  application data, passing to GPU, 306-307
  definition of, 291
  passes, 306
  pixel shaders, 305
    compared to vertex shaders, 291
  process flow, 293-294
  techniques, 305-306
  vertex displacement, 311-313
  vertex shaders, 304-305
    compared to pixel shaders, 291
sharpening images, 318-319
ShuffleCards method, 642
sign function (HLSL), 301
SignedInGamer.SignedOut, 723
SimpleNetworkDemo, 565-577
  Zunes, 578-582
sin function (HLSL), 301
sincos function (HLSL), 301
sinh function (HLSL), 301
size of textures, 208
skybox compilers, creating, 160-161
skybox content objects, creating, 152
skybox processors, creating, 152, 157-160
skybox readers, creating, 162-163
skyboxes
  adding to tunnel vision game, 472
  AI Demo program, 375
    creating, 151-152
      skybox compilers, 160-161
      skybox content objects, 152
      skybox processors, 152, 157-160
      skybox readers, 162-163
    using in games, 163-165
smoothing, 562-563
  real-time multiplayer games, 709-716
smoothstep function (HLSL), 301
software
  DirectX runtime, installing, 15
  Visual C# Express, installing, 11-14
sort modes
  sprite batch demos, 179-183
  sprites, 173-174
sound banks, creating, 125-126
sound demos, creating, 146-149
sound managers, 139, 145-146
sounds
  adding, 542-544
    in 2D games, 251
  sound banks, creating, 125-126
  sound demos, creating, 146-149
variations, 126-133
wave banks, creating, 124
XACT (Microsoft Cross-Platform Audio Creation Tool), 124

**Spacewar project**
creating, 19
deploying to Xbox 360, 24
distributing, 24-25
exiting, 24

**Spacewar Windows Start Kit project**
compiling and running, 18
creating, 17

**special effects**
fire, 428-434
transitions, 417-421
cross-fade (dissolve), 422-423
directional wipes, 423-428
TransitionsDemo project, 422

**spin rate**, 91-93

**splash screen demos, sprite batches**, 176-177

**split screens**, 104-108
Draw method, 108
DrawScene method, 108
player index, 106-107
viewports, defining, 105-106

**sprite batches**, 173
blend and sort mode demos, 179-183
drawing multiple sprites from one texture demo, 177-179
progress bar demo, 183-187
splash or loading screen demos, 176-177

**SpriteFont**, 473

**sprites**, 171-172
2D text, drawing, 189
blend mode, 173

**fonts**
bitmap fonts, creating, 188
TrueType fonts, 187
TrueType fonts, importing, 188

**point sprites**, 452-455
defined, 436

**save state modes**, 174-175
scaling, 173
sort modes, 173-174

**stacks**
definition of, 346

**GameStateDemo project**, 348
BaseGameState class, 349, 356-357
ChangeState method, 354
ContainsState method, 354
Draw method, 373
FadingState class, 368-369
Game1 constructor, 373
GameState class, 348-351
GameStateInterfaces.cs file, 355-356
GameStateManager class, 351-352
IGameState interface, 348
interfaces.cs file, 349
LostGameState class, 367
OptionsMenuState class, 360-361
PlayingState class, 365-366
PushState method, 353-354
RemoveState method, 353
StartLevelState class, 361-363
StartMenuState class, 358-360
TitleIntroState class, 357-358
WonGameState class, 367
YesNoDialogState class, 363
overview, 346-348

How can we make this index more useful? Email us at indexes@samspublishing.com
start menus
- formatting, 590-591
- refactoring states, 591-605
- updating, 528-533
**StartGame method, 588**
**StartLevel method, 509**
**StartLevelState, tunnel vision game, 491**
**StartLevelState class, 361-363**
**StartLocalMultiplayerGame method, 612**
**StartMenuState class, 358-360**
**StartMenuState.cs, 595**
**StateChanged method, 347, 619, 626**
- tunnel vision, 491
- tunnel vision game, 488
**statements, using, 456, 504**

states
- adding, 709
- FSMs (finite state machines), 343
  - creating, 384-387
- of game pads, 93-95
- games
  - creating, 471-472
  - formatting, 605-630
- of keyboard, 89-91
- of mouse input devices, 97-98
- PlayingState, 499
- switching between (2D games), 229-230
- updating, 539-541
**states, managing**
- enumerated types, 346
- GameState class, 348-351
- GameStateInterfaces.cs file, 355-356
- GameStateManager class, 351-352
- IGameManager interface, 348
- interfaces.cs file, 349
- LostGameState class, 367
- OptionsMenuState class, 360-361
- PlayingState class, 365-366
- PushState method, 353-354
- RemoveState method, 353
- StartLevelState class, 361-363
- StartMenuState class, 358-360
- TitleIntroState class, 357-358
- WonGameState class, 367
- YesNoDialogState class, 363
- goals of, 345-346
- stacks
  - definition of, 346
  - overview, 346-348
- **static modifier (HLSL), 295**
- **stationary cameras, 98-100**
- **step function (HLSL), 301**
- **storage**
  - reminders, 447
    - Xbox LIVE Community Games (XBLCG), 724
- **storedTime, 715**
- **storedTime field, 509**
- structs (HLSL), 296-298
- structures, VertexPositionNormalTexture
  - struct, 393
- submitting games, 732-735
- subscribing to XNA Creators Club, 20-21
- **Summary page, 731**
- switching RGB values, 318-319
- switching between states (2D games), 229-230
- swizzling, 294
- **System Link, 552**
- **System.Diagnostics class, 50**
tan function (HLSL), 301

tangent binormal normal (TBN), 406
tanh function (HLSL), 301
targets, render targets, 502
TBN (tangent binormal normal), 406
TCP (Transmission Control Protocol), 555
techniques (HLSL), 305-306

templates
  creating, 583-590
  real-time multiplayer games, 672
  turn-based multiplayer games, 632
testing, micro-benchmark testing, 41-42, 46, 50-51
  CheckPerformance class, 42
  Program.cs file, 46
  time measurement relationships, 50-51
tex1D function (HLSL), 302
tex1Dgrad function (HLSL), 302
tex1Dlod function (HLSL), 302
tex1Dproj function (HLSL), 302
tex2D function (HLSL), 302
tex2Dbias function (HLSL), 302
tex2Dgrad function (HLSL), 302
tex2Dlod function (HLSL), 302
tex2Dproj function (HLSL), 302
tex3D function (HLSL), 302
tex3Dbias function (HLSL), 303
tex3Dgrad function (HLSL), 303
tex3Dlod function (HLSL), 303
tex3Dproj function (HLSL), 303
texCUBE function (HLSL), 303
texCUBEBias function (HLSL), 303
texCUBEgrad function (HLSL), 303
texCUBEproj function (HLSL), 303
text, updating title screens, 525-529

Texture2D class, 65
Texture2D.SetData, 213
textures, 65-66
  3D lighting, 391
    ambient, 397-398
    animation, 414-416
    bump mapping, 403
    creating custom vertex formats, 391-394
demos, 394-397
    directional, 399-403
    normal mapping, 403-407
    parallax mapping, 407-410
    relief mapping, 410-414
AdvancedTexturingDemo, InitializeVertices method, 395
  normal mapping, 406
  relief mapping, 413
  size of, 208
  space, 405
texturing 3D models, 119-120
time measurement relationships, 50-51
timeDelta, 711
timers
  level timers, creating, 508-509
  for vertex displacement, 311-312
title screens, updating, 525-529
TitleIntroState, 720-723
TitleIntroState class, 357-358
tracking high scores, 512-523
TrackPlayer method, 381-382, 682
TrackPlayerStraightLine method, 380-381
transformations, 76
  definition of, 56
  multiple transformations, 57
  rotation
    definition of, 56
    example, 79-80

How can we make this index more useful? Email us at indexes@samspublishing.com
scaling
  definition of, 56
  example, 76-78
  translation, definition of, 56
transitions, 417-421
  adding in 2D games, 243-246
  cross-fade (dissolve), 422-423
  directional wipes, 423-428
  TransitionsDemo project, 422
TransitionsDemo project, 417-422
  cross-fade (dissolve), 422-423
  directional wipes, 423-428
translation, definition of, 56
Transmission Control Protocol (TCP), 555
transpose function (HLSL), 303
Trial Mode, Xbox LIVE Community Games (XBLCG), 721-722
TriangleFan member (PrimitiveType enumeration), 64
TriangleList member (PrimitiveType enumeration), 64
TriangleStrip member (PrimitiveType enumeration), 64
troubleshooting
  bandwidth constraints, 556-561
  latency, 554-555
  packet loss, 555-556
TrueType fonts, 187
  importing, 188
try/catch block, 616
tunnel vision game, 471
  compiling, 472
  crosshairs, creating, 494-495
  game logic, 473-486, 490, 494
    AddEnemy method, 483
    AddMissile method, 477
    CheckCollisions method, 487
    CollisionDetection method, 478
  Draw method, 479-480, 488, 493
  InitializeMissile method, 478
  Load method, 476
  LoadContent method, 476-482
  PhysicalObject class, 474
  StartLevelState, 491
 StateChanged method, 488, 491
  UnloadContent method, 476-482
  Update method, 493
  UpdateEnemies method, 487
  VertexDisplacement.fx, 489
game states, creating, 471
game-specific cameras, creating, 495-499
HUD, creating, 510-511
level timers, creating, 508-509
radar, creating, 501-503
scoring
  adding, 511-512
  tracking high scores, 512-516, 523
skyboxes, adding, 472
tunnels, creating, 504-508
turn-based multiplayer games
  adding, 652-665
  creating, 669-670
  design, 631-651
  functionality, 633-651
  players joining/leaving, 665-668
  templates, 632
types (HLSL), 294-295

UDP (User Datagram Protocol), 555
uniform modifier (HLSL), 295
unit matrix, 56
UnloadContent method, 28, 418, 431, 476-482, 502, 533
tunnel vision game, 476-482
Update method, 406, 419, 509
  AccelerationDemo program, 329-330
  CollisionDemo program, 336
  first person cameras, 102-104
  Particle class, 437
  progress bar, sprite batches, 186
  stationary cameras, 99-100
tunnel vision game, 493
UpdateEnemies method, 487
UpdateEnemyPosition, 680
UpdateFade method, 422
UpdateInput method, 419
UpdateLocal method, 710
UpdateLocalGamer method, 572, 653, 691
UpdateNetworkSession method, 573, 690, 699
UpdatePlayer method, AIEnemy program, 379
UpdateRemote method, 714
UpdateSinglePlayer, 680
UpdateState method, 678
UpdateStateMenu method, 278
UpdateWipe method, 426
updating
  input handlers, 134-139
  options menus, 536-540
  particle systems, 541-542
  start menus, 528-533
  states, 539-541
  title screens, 525-529
  XELibrary, Zune, 258-260
User Datagram Protocol (UDP), 555
using statements, 456, 504
Utility class, 441

V
value abs function (HLSL), 299
value sqrt function (HLSL), 301
value, passing by, 38
variables
  passing by reference, 39
  passing by value, 38
variables (HLSL), types, 294-295
variations, 126-133
vector keyword, 294
Vector3 types, 715
vectors
  normals, 399
  overview, 55-56
velocity, 326
  final velocities, 333
    CollisionDemo program, 338
Velocity property, 709
vertex buffers, 59-63
vertex displacement, 311-313
CreateTranslation method, 313
defined, 311
SetValue method, 312
timers, 311-312
vertex formats, creating custom, 391-394
vertex output semantics, 296
vertex shaders, 304-305
compared to pixel shaders, 291
VertexDisplacement.fx, tunnel vision game, 489
VertexPointSprite, creating, 442-443
VertexPositionNormalTexture method, 62
VertexPositionNormalTexture struct, 393
vertices
  overview, 55
  vertex buffers, 59-63
vertex displacement, 311-313
  CreateTranslation method, 313
  SetValue method, 312
timers, 311-312
view (camera), 58-59
viewing Xbox LIVE Community Games (XBLCG), 719
viewports
cameras, 709
defining for split screens, 105-106
virtual methods, sealing, 51
Visual C# Express, installing, 10
visualization demo, Zune, 262-268
visualizations, Zune, 268-271
voice, bandwidth constraints, 557
volatile modifier (HLSL), 295

X

XACT (Microsoft Cross-Platform Audio Creation Tool), 124
XACT projects, 543
Xbox, 360
  connecting to computer, 21-23
debugging, 25
deploying, 23-24
deploying to, 24
garbage collection, 40-41
monitoring performance on, 37-38
.NET Compact Framework on, 31-32
network demos, creating, 565-570
test demo program, 25-28
Xbox LIVE Community Games (XBLCG)
  artwork, 720
  audio, 722
  controls, 720-721
  creating, 719
displays, configuring, 719
marketing, 725-726
menus, 721
networks, 723
profiles, 723
rich presence data, 724
storage, 724
Trial Mode, 721-722
Xbox LIVE Marketplace, 727-737
XELibrary, updating for Zune, 258-260
XNA Creators Club, subscribing to, 20-21
XNA Framework
  Content Pipeline
    Content Compiler, 113
    Content DOM, 113
    Content Manager, 114

W

WasButtonPressed method, 599
WasPressed method, 598
wave banks, creating, 124
wavy effect, 321
Windows
garbage collection, 39-40
mouse input devices, 97-98
determining state of, 97-98
network demos, creating, 565-577
winning 2D games, 243
wipes, directional, 423-428
WonGameState, 715
WonGameState class, 367
world, 59
WriteNetworkPacket method, 703
loading 3D models, 113-119
overview, 113
texturing 3D models, 119-120
definition of, 9
XNA Game device Center application, deploying Zune, 258
XNA Game Studio Express, installing, 15-17
XNAPerformanceChecker project, 42-46
.xnb file extension, 114

Y

YesNoDialogState class, 363

Z

z buffer, 118
Zune
deploying, 258
DisplayAlbumArtDemo, 255-258
FireDemo, 260-262
games
optimizing, 283-286
running in landscape mode, 280-283
limitations of, 286-287
local ad-hoc Wi-Fi gaming, 578-582
porting games to run on, 273-280
visualization demo, 262-268
visualizations, creating, 268-271
XELibrary, updating, 258-260
ZuneSoundManager.cs, 274-276

How can we make this index more useful? Email us at indexes@samspublishing.com