

1

Introduction (or Everything I Wanted to Know About Micro Java Gaming But Was Afraid to Ask)

IN THIS CHAPTER

- A New Era of Gaming
- This Book's Mission
- A Bit About Game Design
- Show Me the Money: Micro Game Business Models

A New Era of Gaming

Ah, games.

Games have almost a religious, ritual aspect to them. They allow people to enter together into a higher state of being, pushing skills to new limits and experiences to new heights. They allow ordinary people to experience extraordinary emotions—the emotions of the warrior, the king, the spy, and the lover—while remaining protected in a safe environment.

Now all this might sound like a bit of a heavy-handed way to describe *Frogger*, but it's fair to say that games transport us and amuse us in ways that no other form of entertainment can.

A Brief History of Games

Games have been with humanity since the beginning. A 5000-year-old Mancala-like game board, carved from stone, was recently unearthed in the Sahara. The game of Go, popular in Oriental countries, has reportedly been around since 2000 B.C. Backgammon-like games such as Tabula and Nard are talked about in ancient Roman scripts, and

even in the Bible. And Tarot decks, initially used to help predict the future, evolved into today's Bicycle playing cards.

A decade or two ago, the only games that people spent much time with were professional sports, board games like *Monopoly* and Chess, paper and dice games such as *Dungeons and Dragons*, and card games like Poker or Hearts. Some games were for heavy money, some were bone-jarringly competitive, but most were just about good clean fun.

With the advent of computers, games entered a new era. Games became one of the main reasons many people brought these strange beige boxes called computers into their homes. Whether battling through a simple graphical tennis game such as *Pong*, or a rich, text-only world such as *Zork*, these were wholly new types of games that could be played anytime against a most formidable opponent: a game designer who had programmed your computer, long ago, showing it how to defeat you.

The arcade wave of the '70s and '80s, led by hits such as *Pac-Man*, captured the hearts and ate the quarters of millions of youths. Console systems such as the Magnavox Odyssey, the Atari 2600, Mattel Intellivision, and ColecoVision brought the fun of the arcade to the players' own living rooms. Then, in 1985, a box known as the Nintendo Entertainment System blew people away with stunning graphics and intricate gameworlds, typified by such hits as *Super Mario Brothers*.

Computer gaming entered a whole new stratum of mass popularity and acceptance with bestsellers such as *Doom*, followed by *Quake*, and later *Tomb Raider*. Clearly, ultra-realistic 3D worlds were a hit. The more a game made a player feel as if she were actually inside another reality, the better.

Graphics became richer and richer as 3D cards and engines doubled in speed and performance with each passing year. Super Nintendo gave way to the Sony PlayStation, and currently the Nintendo GameCube faces off against the PlayStation 2, not to mention Microsoft's daunting new Xbox.

Multiplayer Mania

A funny thing happened on the way to virtual reality-ville. In the late '90s and early 2000s, with games like *Ultima Online*, *Everquest*, and *Age of Empires II*, not to mention the spread of casual game Web sites such as Pogo, Yahoo Games, and Microsoft's MSN Gaming Zone, it became clear that what mattered to a whole slew of gamers wasn't only the richness of graphics or the detail of blood and gore—but the presence of other, real people. *Multiplayer gaming*, long popular with the geek crowd, had entered the mainstream.

In a way, games had come full circle. Once again, games were serving a social purpose, becoming a way for two or more people to enter new worlds and test new skills together, relating to each other in entirely new ways.

Micro Devices, Micro Lifestyles

While multiplayer gaming continues to grow in popularity, another big paradigm shift is happening.

It's becoming harder and harder to find people who don't carry network-enabled embedded devices with them wherever they go. Whether it's a PDA such as a Palm device or iPaq, or a mobile phone such as those crafted by companies like Nokia or Motorola, people are getting used to connecting and communicating with each other anytime, anyplace, and anywhere.

Today, there are more than 600 million mobile-phone users worldwide. In the United States and Europe, mobile phone users generally tend to be affluent, educated, and they often have lots of time on their hands. The picture is different on different continents. In Africa, Asia, and South America the masses have flocked to mobile phones because land-line access and Internet service are too expensive.

According to the Yankee Group, people in the United States spend 50% more time commuting than in any other country. This is the perfect time to pull out a mobile phone and play some quick games.

Additionally, Datamonitor has researched people's game-playing behaviors in Asia, Europe, and the United States, and has concluded that most people like to play wireless games on evenings and weekends.

In the near future, we will likely see micro devices become even smaller and more specialized. Phones the size of earplugs, voice-activated assistants on wristwatches, and smart chips on credit cards are all becoming a reality.

This is a continuation of the paradigm shift that began in the 1970s, with microcomputers taking the power away from huge, monolithic mainframes. Clearly, millions of small devices working together yields much more distributed power than one big, central device.

Unsurprisingly, games are keeping up and even helping to lead this paradigm. While it might seem silly to try to achieve a rich, meaningful immersion on a tiny 100×100 pixel screen, there's one thing mobile phone games give you that even the best consoles can't provide: They're always with you, and can be played anywhere you go. This not only means that games can now be more convenient, but wholly new types of games can be designed that take advantage of new lifestyles.

Enter Micro Java

The Java language, created by Sun Microsystems, is another example of a paradigm shift. As a language that had no pointers or complicated memory operations, was object-oriented, secure, and could run on most any browser or platform, application development suddenly opened up to the masses in a way that never seemed possible

before. Java made it possible for millions of programmers to create quality applications in record time and quantities.

The Java 2 Micro Edition (J2ME), or Micro Java, as we'll call it in this book, is an attempt to take the best aspects of Java and pare them down for smaller devices such as mobile phones; set-top boxes that add interactivity to television, pagers, handheld organizers and personal data assistants (PDAs); as well as embedded chips that you find in devices such as refrigerators, microwaves, "smart" credit cards, and automobiles.

Most every major mobile phone and handheld device manufacturer immediately realized the potential of J2ME: If Java were to be placed on the gadget, hundreds of thousands of developers would immediately be able to create applications and add value. Furthermore, because it's Java, a program written for one device would be able to run on another device with little or no modifications. That certainly makes more sense than trying to force developers to learn a native language and API in order to create programs for your phone.

Seeing the opportunity for Java on the handset, almost every major mobile phone manufacturer joined with Sun to create something called the CLDC: The Connected, Limited Device Configuration, along with the MIDP: The Mobile Information Device Profile. In later chapters, we'll get into greater detail about what all these wacky acronyms really mean. But the point to remember here is that mobile phone manufacturers have embraced Java in a way that not even PC manufacturers and browser makers have. Java is clearly the future platform of choice for mobile devices, and an ideal platform for mobile games.

This Book's Mission

We have attempted to write the most in-depth guide showing you how to craft the most cutting-edge Micro Java games possible.

Whether you are a professional game designer hoping to expand your knowledge of various platforms, a game programmer who wants to port a game to a smaller device, a Micro Java enthusiast looking for a more entertaining book about more entertaining apps, or just a micro gamer hoping to catch a glimpse of what goes on behind the scenes, this book is for you.

The Game Plan

This book is divided into six sections:

Part I: Small Devices

The book begins with a tour of current Java-enabled devices, showing the full canvas upon which you'll be able to paint. These devices include powerful, full-featured computer systems, set-top television boxes, and tiny, smart credit cards.

Next, we'll look at the current state of micro gaming. We'll go on a whirlwind tour of some of the most popular and revolutionary games out there. Because most of these games are not written in Java, we'll try to distill the most successful element of these games so that you can take the best ideas and run with them.

Part II: Before, Between, and Beyond J2ME

In many cases, handheld games will not be written in Java alone. Rather, games will be built atop older mobile phone technologies. In the second section of this book we'll look at the technologies that surround and support J2ME gaming, such as the Wireless Application Protocol (WAP) and Standard Messaging System (SMS). Furthermore, we'll cover specific enhancements to the current crop of phones from brands such as Nokia, Siemens, Motorola, Ericsson, and NTT DoCoMo, allowing you to take games to a new level no matter what your target platform happens to be.

For example, some carriers provide location-based information. This is an extremely exciting and relevant tie-in to gaming. This will allow people to literally use their mobile phones to hunt down or otherwise play with each other through the physical, bricks-and-mortar world.

Part III: The Java 2 Micro Edition

This section dissects the J2ME in all its gory detail. You'll learn how to build J2ME applications, which tools to use, and key programming techniques.

Programming for handheld devices is often much different than coding for a full-blown desktop computer. However, it doesn't have to be more difficult.

Part IV: Let the Games Begin!

This is where things start getting deep. We'll thoroughly cover the nooks and crannies of J2ME, along with in-depth discussions on graphics, sounds, animation, multi-player networking, and other game-related topics.

Additionally, one of the most important things this book will show you are the limitations of Micro Java and, in certain cases, how to get around them.

Each section will include lots of source code, so that you can immediately begin compiling, tweaking, and testing things out.

Part V: J2ME Extensions

J2ME is a cross-platform standard. Any program you write in J2ME should work, more or less, on any other mobile phone or handheld device. However, every device has its own specialties and intricacies.

This section will cover other forms, profiles, and configurations of J2ME. For example, you'll learn a little bit about coding for a set-top television box. In

addition, we'll focus on two popular Application Programming Interfaces (APIs) from the world's largest handheld hardware platforms.

Finally, this section will show you the best ways to take game elements from one platform and port them to others.

Part VI: Micro Racer

Every good thing must reach its end. But rather than just stuff you full of knowledge and then leave you alone in the vast, dry desert to figure everything out, this book includes the full code to a superior Micro Java game that we call Micro Racer. Check out Figure 1.1 for a sneak preview.



FIGURE 1.1 You will learn how to build this game.

Micro Racer is a fast moving, multiplayer experience. The game pushes the envelope on Micro Java's graphical, sound, and networking abilities.

You begin the game with a simple racecar. You can race around all you want, picking up bonus points, avoiding crashes, and exploring new tracks.

Over time, however, your car will experience wear and tear and might even break down. You will need to log into The Garage to fix up your car.

At The Garage (see Figure 1.2), you'll be able to buy new parts, trade away old parts, and compare your score and standing. As you gain more and more money, you'll be able to soup up your car with turbo boosters, nitro packs, monster tires, spiked wheels, oil slicks, smoke screens, and other extras.

The people you trade with at The Garage are not artificial intelligences; rather, they are other actual players.



FIGURE 1.2 The Garage: Where you log online and trade car parts with other users.

Although Micro Racer is an advanced game, we believe you'll be able to do even better.

It is our hope that you will take this game, and the knowledge learned throughout this book, and go on to create bigger and better things.

A Bit About Game Design

Before you can begin the fun/tedious/interminable process of actually typing Java code, compiling it, testing it, debugging it, and so on, you'll actually need to design the game you're interested in.

If you already have a game design written, or are working based on somebody else's game design, you can skip this section.

But if you're interested in a brief discussion of how the heck people think up new types of games, you've come to the right place.

Game design is always hard. Designing for a medium as new as mobile phones is even harder. But it is the best of worlds, as well as the worst of worlds. Although the devices you'll be designing for are limited compared to game consoles or PCs, they are also an entirely new phenomenon being used in entirely new ways.

If you can understand the way mobile phone users really think and act, you might be able to create a type of game that nobody has ever thought of before.

The Game Design Process

Every game designer develops his or her game using a different process. Some people like to jump in and begin coding straight away; others like to create a monolithic 500-page design document outlining every last variable and button.

The type of process you use depends on the size and experience of the development team, as well as your personal philosophy on what makes a good game.

No matter what approach you choose, pretty much every game goes through the four P's:

1. Preproduction
2. Prototyping
3. Programming
4. Playtesting

Preproduction

Preproduction usually involves generating a whole lot of paperwork.

Different game designers work in different ways. Some are technically minded, and like to jump right into the thick of things and create use-case diagrams, specifications, and so on.

Others are more artistically minded, and enjoy storyboarding the graphics, letting somebody else worry about how to make nitty-gritty interactions happen.

But pretty much everybody, at some point, needs to use regular pen and paper (or Microsoft Word) and just spell out the *story* of the game—the feel, the depth, the breadth, and the intent.

Taking the time to write clear design documents and storyboards during preproduction will pay off later during development. The more you can describe every bit of art, sound, and interaction, the easier it will be to put all these pieces together during the frantic phase of actual development.

The bigger your design team, the more helpful a solid design document will be in keeping everyone speaking the same language, understanding the same goals, and working on the same product.

Answering Questions

Good design documents usually answer an implicit question. No matter how or when exactly you do it, every game designer will need to and answer the following questions:

- What is the game's genre?
- What are the limitations of the game?
- What is the game's central mission?
- What are the inputs, and what are the outputs?
- How will the game play out?

Picking a Game Genre

There are literally millions of games in the world, and tens of thousands of computer games. But all these games can be broken down into genres.

A genre is more than a style of gameplay; it is also a mood. Different genres appeal to wholly different audiences. Clearly, a gory first-person shooter is expected to have a different interface, feel, sound effects, and speed than a long, drawn-out, and detailed military simulation game.

Genre will help define how the game looks, how it feels, how it plays, and who it is targeted to.

This section will briefly cover various genres, helping you to hone in on a gameplay experience.

Copying, Stealing, and Cloning

A sad fact of life is that most games on the market are basically clones of other, more successful games.

When Java applets first came out, most of the games that people created were exact copies of old hits from the Apple II, Atari 2600, or Commodore 64 era. Often, the only thing that a programmer would change would be the name and a few graphics: *Pac-Man* might become something like *Pork Man*.

Likewise, it is tempting to take existing games and create Micro Java versions of them. Furthermore, there's nothing wrong with it. After all, classic games have been time-tested and proven to be popular with the masses.

CAUTION

If you are creating games as a hobby, then there's no problem with taking your favorite arcade games and squeezing them into a mobile phone so that you, and others, can enjoy them portably.

However, if you are creating games commercially, not only is copying an existing game illegal, but you'll likely find that there won't be a big market for it. As much as people like to play their standard favorites, the world is thirsting for something new. History has shown us that the company or person that uses Micro Java to design a game genre that nobody has ever seen before will be the one that triumphs in the end.

All that being said, some of the best games ever created borrow familiar elements from one or more forgotten genres and breath new life into them. For example, real-time strategy games—games in which the player controls many discrete units, all at once—have existed for the past few decades. But it took Westwood Studios to create a game in the genre with a strong story, well-balanced play, and distinctive military units. The game was *Command and Conquer*, and it became an instant hit.

Because Micro Java game designers are stuck writing to such a limited platform, you are forced to think about unique game design itself, and not rely on fancy graphics and sounds to make sales. Some of the best games were black and white, 8-bit, and had less than 64K of memory. Try to analyze those games and understand what made them great. Using classic games for inspiration is not only acceptable, it is essential.

What Types of Games Are Possible?

Ultimately, the most successful games will combine genres in entirely new ways. For example, the *Tomb Raider* series is so popular because it blends action, adventure, puzzles—and the shapely Lara Croft.

The following list of genres is just a starting point to get you thinking. This list is in no way complete.

- **Action Games**—These are games that involve fast reflexes. The graphics are generally as realistic as possible, and the audio is usually rich and loud. The play is usually fast paced, and multiplayer versions are usually very responsive. The audience consists generally of adolescent males.

Because of the speed, responsiveness, and powerful graphics, action games are probably the hardest genre to implement on mobile phones and other hand-held devices. This book will show you how to do it, anyway.

Examples of such games include first-person shooters such as *Quake*, space games such as *Defender* or *Missile Command*, maze games such as *Pac-Man*, and paddle games such as *Pong*.

- **Combat Games**—These games usually involve two characters facing off against each other and trying to beat each other up. Often, the characters will have special powers. Winning the game requires that the player have quick reflexes as well as memorize all the possible “moves.”

Examples include *Virtua Fighter*, *Street Fighter*, and *Mortal Kombat*.

- **Adventure Games**—These are games that involve a quest of discovery through new worlds. These are usually structured similarly to a good movie or book, with a strong sense of story, character, plot, and locations.

Originally, these games were wholly text-based, such as *Zork*; but more modern games such as *Monkey's Island* and *Riven* use advanced 3D graphics, strong artificial intelligence, and rich audio to flesh out the game worlds.

- **Puzzle Games**—These games require the player to use logic, and often involve the arrangement or matching of symbols. *Tetris* is the king of all puzzle games.

The audience for puzzle games is usually made up of intelligent, crafty adults.

- **Strategy Games**—These games often involve lots of pieces, lots of possibilities, and rewards for thinking ahead.

War games such as *Panzer General* are a popular type of strategy game in which you try to recreate a famous battle and pit various armies against each other. The audience for war games is very enthusiastic, but very small.

Real-time strategy games such as *Command and Conquer* and *Warcraft* are much more popular with the masses. These games often involve more tactics than long-term strategy. Players must manage resources such as electricity and money while assembling specialized armies consisting of many different units. Quick reflexes are as important as long-term planning.

Finally, classic two-player board games such as chess, Reversi, Connect Four, and checkers are strategy games. The audience for this type of classic turn-based game is truly mass market.

- **Role Playing Games (RPG)**—These games generally allow you to fill a role. Your character has certain attributes such as Strength and Wisdom, and these attributes can change over time as your character explores new dungeons and fights new monsters.

Paper and dice games such as *Dungeons and Dragons* invented this genre. The typical audience for this type of game is similar to those who read science fiction—usually intelligent, male adolescents.

With more graphical RPGs such as *Diablo III*, *Everquest*, and *Ultima Online*, the genre has moved online as the basis for a rich, social, active community.

- **Simulation Games**—These games allow the player to control a character, a machine, or system. Often, these games rely upon ultra-realistic graphics and control panels.

The more specialized the simulation, the smaller the audience. A very detailed flight simulator may only appeal to real pilots. Real-life simulation games such

as *SimCity* or *The Sims*, however, are widely popular with males and females, children and adults.

- **Trivia Games**—These games are tests of (often useless) knowledge. Trivia games can be played in a straightforward question-answer format, such as *Who Wants to Be A Millionaire?* or *You Don't Know Jack*, or by using a more sophisticated game board, as with *Trivial Pursuit*.

Most game shows are based on trivia. The audience for trivia games is the mass market.

- **Word Games**—These games involve the creation of words, based on specific rules. The more words the player knows and is able to build, the better the player does. Examples of this genre are word builders such as *Scrabble* or word searches such as *Boggle*.

Word games often appeal to an intelligent, middle-aged female audience.

- **Card Games**—Card games usually combine chance with skill. A player is dealt out a hand and must play out the hand, given a set of rules.

A card game such as poker involves bluffing and betting, appealing to a much more hard-core gaming crowd than social trick games such as Hearts or Spades.

Additionally, collectible card games such as *Pokemon* or *Magic: The Gathering* combine elements of the RPG, allowing players to collect decks of cards, battle the decks against other players, and combine cards to achieve unexpected results. This type of game usually appeals to adolescents or hard-core RPG gamers.

- **Games of Chance**—Any game based upon random result. Most casino games are games of chance, with a little skill thrown on top. Roulette, slot machines, or the card game War are the most basic games of chance.

Games such as Backgammon involve chance, but also require a great amount of strategy.

- **Sports Games**—These games allow the player to experience physical sports such as football, basketball, wrestling, or skateboarding. The games usually have excellent graphics and highly realistic physics. These games usually appeal to the same fans that enjoy the sport itself.

Some sports games are coaching or managing games, and allow the player to take a more strategic, top-down, and sideline approach to team building, player trading, or game-playing.

A special subset of sports games worth singling out is racing games. These games usually involve very detailed roads and landscapes, very specialized user input, and very responsive physics.

- Toys—This is the rarest category of games, but also one of the most interesting. These games generally have no winner or loser, but allow the player to build or play with virtual pieces.

Virtual pets, virtual mousetraps, virtual robots, digital musical instruments, and other educational and kids games often fall into this category.

Know Thy Limits

The most important part of the game design process is to know the limitations of the medium. This book, especially Chapter 2, “The Mobile World,” will help you to define exactly what your target platform can achieve.

Designing Within Restrictions

In this book we’re focusing on handheld devices such as mobile phones. A mobile phone typically has a tiny black and white screen, tiny bins of memory, ultra-slow screen refresh rates, turtle-like processor speed, and painfully limited sound.

So a game with instant trigger finger reactions, endless 3D dynamically shaded passageways, a massive multiplayer environment, and with a soundtrack by Green Day is *not* going to be possible on mobile phones. Not today, at least. There will definitely be a day—even relatively in the near future—when chipsets are fast enough and small screens are colorful enough for this to be possible.

In a way, designing a game for a mobile phone is a blast back to the olden days of game design, for platforms such as the Apple II and Commodore 64. You’re now back in a world where every bit counts, only worse: You now have to fit it all on a postage-stamp size screen.

There is another drastic difference: One thing most J2ME-equipped mobile phones enable is easy interactivity with other mobile phones. For the first time, communication might become more important than gameplay.

Designing Around Restrictions

It is useful to remember here that no matter how good a game’s graphics are, the real action always occurs inside the player’s head.

A game’s graphics and other elements are only useful if they transport a player to a different *mindset*, and allow the player to experience a believable fantasy.

Your challenge, then, is to transport the player to a rich, believable, exciting, and emotional fantasy world while using minimal graphics and audio. Sound hard? Not really. Novelists and storytellers have been doing just that for centuries, using no graphics at all.

That is the first clue: Good writing in Micro Java games becomes more essential than ever.

A good Micro Java game designer is also about turning lemons into lemonade. Good designers can actually take new devices such as mobile phones and use them in ways that nobody has ever imagined or expected, but that are wholly intuitive and logical.

For instance, one of the most ingenious mobile phone games out there is a Japanese game called *Turibaka Kibun* (which means Crazy for Fishing), created by Dwango. To go fishing, you pick your bait, choose a fishing hole, and then literally extend the antenna of your phone and hold it out. Eventually your phone will vibrate, which means you have a fish on the line. If you get lucky, you'll be able to reel in a nice trout or bass.

While the game, shown in Figure 1.3, might sound a little strange to Western audiences, it is wildly popular in Japan. In fact, DoCoMo had to limit the number of fish one could catch each day because consumers were spending *too much* time and money with the game.

More information about this and other games can be found in Chapter 3, "Big Games, Small Screens."

Another example of a game-like event that could only happen in today's mobile phone era is a performance called Dialtones. This is a symphony concert performed entirely through the ringing of the audience's mobile phones! Visit

<http://www.flong.com/telesymphony/> for more information and sample songs.

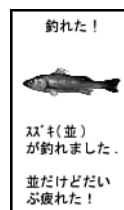


FIGURE 1.3 *Turibaka Kibun*: A game that would only be possible on a mobile phone.

The Game's Mission

After you've decided what the genre will be, one of the first tasks of the game designer is to define the game's mission.

Most good games can be summed up using a simple sentence. The sentence should evoke an entire mood, and explain the central challenge of your game from the player's perspective.

For example, *Tomb Raider* can be summed up like this: You are a hot, sexy adventurer who must explore secret passageways within ancient tombs, collecting treasures while fighting off deadly creatures.

Tetris could be summed up this way: Different-shaped puzzle pieces are falling down a chute; you must rotate and arrange the shapes so that they land at the bottom forming complete rows.

Every design, artwork, and programming decision must then stem from this mission statement.

Inputs and Outputs

A game, at its core, consists of user input, followed by some sort of output. You should try to list every type of input, and what the effect will be.

Some input occurs because the player does something. Other types of input occur just because the game state has reached a specific point.

Typical input and other events to keep track of and define include the following:

- The keyboard: Which keys on the handset will be used, when, and for what?
- The mouse or joystick: Most handheld devices do not have a mouse, but some do have a touch screen or stylus. How will this affect the input?
- Menus: What main and top menus will there be in your game?
- Buttons: What buttons will there be?
- Form widgets: How will elements such as pull-down menus, radio buttons, checkboxes, and text fields work together?
- Time: Will there be any countdown timers? How does time play a role in the game?
- Collisions: What happens when graphical elements collide?

Next, you should try to create a list of every element that will actually be in the game. These elements vary widely. Some will be visible on screen, and some will be hidden game state variables that your program will need to juggle:

- Graphical elements: What will the user see? These are usually 3D models or sprites.
- Sound effects: What audio effects should play, and when?
- Background music: What music will be playing?
- Background art: What will the environment look like, and how should it be rendered?
- Levels: Will the game have multiple levels? If so, what will differentiate them?

- **Interface:** In order for input to happen, there will need to be an interface. How will this interface look, roughly speaking? The interface also usually includes a readout of variables such as score, number of lives remaining, amount of ammunition, and so on. What information needs to be here?
- **Artificial intelligence:** Will there be any computer players? What will they look like?
- **Global variables:** Try to create a list of global variables that will change as the game is played. This includes the score, the round number, and so on.

Often, a design document will list each input and output element in a table and explain how different elements interact with each other. You should also try to explain the different classes and subclasses of elements, and how they all relate.

This document can often be used to define exactly how the program should be structured in an object-oriented manner. This will help the object-oriented Java programmer design the actual software. For example, a typical unit in your game may be a `DeathMosquito`. This `DeathMosquito` may be part of the `FlyingUnit` class, which may descend from the `WarriorUnit` class, which will be derived from the generic `Unit` class, which in turn may be a child of the `Sprite` class.

Gameplay

The next step is to actually define the rules of the game. This is where you can begin to determine all the variables, graphical elements, and other gameplay elements.

Ultimately, you should be able to create a *game state*—a list of variables, or perhaps just an array of bytes, that defines the exact state of the game. Strip away the fancy graphics, graceful animations, streaming TCP/IP sockets, and eardrum-beating sound effects, and you’ll notice that games—no matter their genre or complexity—amount to nothing more than a pile of bytes. Every player’s move and every artificial intelligence decision eventually expresses itself as a change to this core game state data.

You should be able to stop the game at any time, restart it, plug in the game state, and be at the exact same place you left off.

Java makes it quite easy to keep an abstract notion of game state. Just create a class with all the data structures you need, tap in methods to access or change that data, and you’re off and running. By designing state as an object, various parts of the state can quickly be accessed and altered.

Multiplayer games often keep the main copy of the state on the server side, with additional copies in each client. This permits the server to be the final judge of what the “game” actually is. The client, meanwhile, can contain just enough information

to be responsive. In other words, the client should be able to tell whether a player's move is legal or illegal, but the server will actually register the move and make changes to the game state accordingly.

The other important piece of this picture is how the game is won, and exactly how to determine winners and losers. You should be able to analyze the game state variables and determine whether the game has reached the winning condition.

For multiplayer games, it is usually useful to draw a client-server diagram and show which messages will need to be sent over the network. This can help you create use-case scenarios to take care of any eventuality.

Other Resources

There are many books, magazines, and Web sites that discuss game design. Some of the best resources can be found online:

- <http://www.gdse.com/>
- <http://www.gamasutra.com/>
- <http://www.gamedev.net/>

Prototyping

The more original your game idea, the more important it is to prototype it. Until you and some friends are actually playing the game, you will never have any idea how successful your genius idea really is.

To prototype a game, one can commonly use a notepad, a few index cards, and some pencils. Each index card can be a game output element. You can position these relative to each other, or move them around accordingly.

Get a few friends together, explain the rules of the games, and “play it.” You can act as the computer and game master, keeping track of the score and making sure everybody is playing correctly.

After a few minutes of play, it will become remarkably evident what the weaknesses and strengths of your game design are. Continue redesigning the game and retesting it, until your friends get sick of it or until you're happy with the results.

Additionally, you can easily prototype most games using Java Standard Edition (J2SE). This is another joy of Java—it is extremely easy to create a simple application that takes in command line input, processes some simple rules, and then spits out an output.

For example, if you are creating a new type of card game, you can have your Java prototype shuffle the cards, deal them out, accept valid moves, and keep track of who has what.

Eventually, if you only use text for input and output, it will be easy to transport the prototype in the Java 2 Micro Edition environment. The prototype can become the actual rules engine for your final game.

Programming

This part of game development is similar to developing any other application. You've got a specification and you've got to carry it out, on time and on budget.

You've got to create your Java classes, possibly create a server, create any artwork or audio assets, and fold it all together.

Most games are basically an endless loop. Speaking in the most general terms, the loop works as follows:

1. Paint the screen.
2. Get any user input.
3. Make any game state changes.
4. Redraw the graphics or sounds accordingly.

Most games also have engines for each major multimedia aspect. The advantage of having a generalized engine is that it can be reused for future game products. Typical engines include some of the following:

- Graphics engines are a quick way of drawing the graphics. 3D games will have a special graphics *3D engine* that knows how to take three-dimensional X, Y, and Z coordinates and transform them onto a flat screen. Other games will have *sprite engines* that enable you to take many graphical components and animate them and move them around the screen relative to each other. Still other games will have *isometric engines* that draw 3D-looking graphics from a set perspective, actually using a series of two-dimensional overlays.
- Audio engines will play the soundtrack or other audio effects. Often, the engine will mix together different effects and be smart about fading music in or out depending on what is currently happening in the game.
- Artificial intelligence (AI) engines act as a separate player in the game. The AI player is able to compete in the game, often head-to-head against human players.

- Physics engines simulate real movement. Making a ball fall and then bounce appropriately takes a very complicated series of equations. A physics engine can provide this.
- Multiplayer engines will communicate with the network, often through a central server, and enable game sessions to speak with each other.

Playtesting

After the entire game has been coded, debugged, and released, the development has just begun.

Because a game is not a cut-and-dry business application, there is usually no right or wrong. There is only fun and not fun. You may think your game is highly entertaining, but you're biased—you've been working on the sucker for the past few months. You also may not be representative of the market you're trying to appeal to.

Big game companies often hire focus groups to playtest their game. They also might release the game to a small group of beta testers. They'll try to get as much feedback as possible.

Many of the most popular games became huge successes because beta testers loved the game so much they worked hard trying to communicate small requests that would make the game even better. When the game company fulfilled these requests, beta testers felt a sense of ownership. They told all their friends to buy the game, and the news spread like wildfire.

The first playtester should be you. Be honest with yourself. What improvements can be made? What strategies are too hard, and why is it so easy to gain points if you know a certain trick?

Continue to tweak the game until you're absolutely certain there's nothing wrong with it. You should then have some friends of yours play your game. This will be more useful if your friends are avid gamers, and if they are game designers themselves. Watch them closely while they play, and ask them many open-ended questions about their experience. Notice when they get frustrated or bored. Notice when they get angry, or when they laugh.

In nearly every case, you will need to go back to redesign and reprogram your game. This might be as simple as changing a few values, adding a few power-ups, or removing a few restrictions. Or you might need to totally redo your graphics engine to make it animate more smoothly.

Often times, you'll need to drastically change your game design. And you will need to go through the entire prototyping and programming process again before you can be absolutely sure your new design idea works. Fun, huh?

As a rule of thumb, professional game companies often spend as much as a third of the game development cycle on playtesting and redesign.

Show Me the Money: Micro Game Business Models

If you are a commercial game developer, then you are lucky indeed. You get to spend your days hacking, designing, and creating objects of joy and entertainment. You get to be a kid for a living.

But if you want to stay in business, you'll need to make money. Clearly, the business model you choose will differ depending on your end platform and upon your target audience.

Additionally, business models are strikingly different in the United States, Asia, and Europe. In Europe and Asia, for example, carriers such as NTT DoCoMo offer a profit split with content providers: The more a user chooses a particular piece of content, the more the content provider gets paid. This model is exciting, because it encourages thousands of developers to take their best shot at entertaining the masses.

To date, few North American carriers have been able to offer such a deal. Instead, most content providers must approach specific carriers and strike specific content deals. This makes it difficult for small developers to compete or earn any real revenues.

NOTE

United States carriers such as Cingular Wireless, Sprint PCS, and Nextel have expressed interest in creating profit-splitting services within the next year.

The Business Outlook

Datamonitor predicts that the wireless gaming market in the United States will have grown to \$3 billion in 2006, with 125 million players hungry for good new games.

Advertising and Sponsorships

Advertising and sponsorships are probably the easiest business models to implement, but the most difficult in which to achieve solid revenues.

The idea is simple and well-known: Find a company that has a message, put that company's name, logo, or other creative elements within your game, and you've created a valuable vehicle for the company's message.

In fact, many companies have opted to create their own mobile phone games in order to deliver their brand to a cutting-edge audience.

Often, advertisements change from day to day. Ads appear below or to the side of game content. Alternatively, a full screen ad “interstitial” can be shown to the player before or after the game session.

Some of the best advertisements don’t even seem like ads at all. For example, many racing games include logos “painted” on the racecars, and football games often include ad banners on the side of the stadium. This touch of realism actually makes the game better, while providing a permanent and well-seen home for a lucky advertiser.

The problem with micro devices, of course, is that there is not a lot of room for ads. Company logos are often small and washed out, and it is often hard to track the number of times a given ad is seen.

As screen resolutions improve, however, advertisements and sponsorships will likely become a smart choice. Top games will be able to charge hefty fees for ad placement. After all, if a mobile phone game really takes off, it has the potential to be experienced by more people, and more regularly, than any television, radio, or print ad campaign.

Content Deals

Wireless service providers, cable companies, and other companies that provide the infrastructure for small devices have a lot to gain if a popular game comes along.

Because most mobile phone providers charge their users per minute, the longer a user is connected and playing a favorite game, the more minutes are being used up.

In addition, games could come with incentives. For example, if you pass a certain level in a game, you could get a coupon for 50 free minutes of mobile airtime. Players would work on the game for hundreds of minutes trying to earn a slight discount.

WARNING

Because some carriers charge a flat monthly fee for Internet use, these carriers desire games that *don't* stay connected and waste precious bandwidth!

But the point remains that carriers have invested more than \$100 billion to create a faster, next-generation wireless infrastructure known as third-generation (3G). Current networks run 9.8 kilobits per second. A 3G network will run up to 50 kilobits per second—almost as fast as a computer’s modem.

Clearly, carriers are counting on faster and richer applications, such as games, to attract new users and get a good return on their investment.

Currently, there are an estimated 16 million wireless game players in the United States alone. The ARC Group predicts that by 2006 there will be 280 million players. In Japan, NTT DoCoMo recently announced that 52% of wireless Internet revenues are due to games.

Several game companies have been able to strike content deals with major wireless carriers. If you have created a game that you believe will appeal to the masses, it's definitely worth talking with major carriers and figuring out a deal that makes sense for everyone.

Pay-For-Play or Subscription

Charging players for a subscription to play a game is a clear path to revenues. For example, a player may be willing to pay \$10 per month, \$1 per game, or an additional 10 cents per minute.

However, most current users aren't willing to pay anything for mobile phone games. The main reason for this, of course, is that while there are many nice micro games out there, there are few that are so darn fun, so darn special, so darn enthralling, and so darn exciting that users would be willing to part with their cash.

This will change.

Bigger and more colorful screens, better audio capabilities, quicker network access times, and faster processors will allow for better games.

Additionally, carriers will begin to offer content providers more ways to bill users. Carriers and phone manufacturers are already beginning to create portals whereby users can use their credit cards to purchase and download Micro Java applications.

Someday soon, a company will create a micro game so good and so addictive that people will *have* to play it. Paying a buck or two per game will become second nature. The company that does this will make a fortune, and the world of micro gaming will be changed forever.

Perhaps that company will be yours.

Summary

It is our hope that readers of this book will go on to do more than create an excellent crop of new games. We believe that given the paradigm shifts of always-on networking and in-pocket interactivity, the J2ME games of tomorrow have the potential to redefine the whole medium. In fact, given the pervasiveness of handheld devices and the potential for reaching a wider audience than ever before, a truly original game concept can revolutionize the world.