# Computer Games

Computer games are programs that enable a player to interact with a virtual game environment for entertainment and fun. There are many types of computer games available, ranging from traditional card games to more advanced video games such as role playing games and adventure games. In this chapter, we first discuss the different types of computer games. The architecture of computer games is also described. Finally, the programming environment that is used to build the computer games is discussed.

## 1.1 Types of Computer Games

Although computer games mainly provide entertainment and fun, it also improves hand/eye coordination and problem-solving skills. Each game has its own strategy, action and fantasy that make each game unique and interesting. Generally, we can classify computer games into the following types: card games, board games, puzzles, maze, fighting, action, adventure, role playing, strategy, sports and simulation games. However, the classification is a fuzzy concept, as many games are hybrids that fall into more than one class. For example, *Doom* can be classified either as a maze game or an action game, while Monopoly can be classified as a board game or strategy game. The different types of computer games are briefly described as follows:

***Card Games***
They are computerized versions of traditional card games, or games which are essentially like card games in that they are primarily card-based (such as solitaire). Examples of card games include *Blackjack*, *Bridge*, *Casino*, *Solitaire* and *Video Poker*.

***Board Games***
They are adaptations of classic board games. Examples of board games include *Chess*, *Checkers*, *Backgammon*, *Scrabble* and *Monopoly*.

***Puzzles***
Puzzle games aim at figuring out of a solution, which often involves solving enigmas, navigation, learning how to use different tools, and the manipulating or reconfiguring of objects. *Mastermind* and *Tetris* are examples of puzzle games.

***Maze***
Maze games require the successful navigation of a maze. Mazes can be viewed in different ways. For example, they may appear in an overhead view (as in *Pac-Man*), or first-person perspective (as in *Doom*).

***Fighting***
Fighting games involve characters who fight usually hand-to-hand, in one-to-one combat situations. The fighters are usually represented as humans or animated characters. Fighting games include *Street Fighter*, *Avengers* and *Body Slam*.

***Action***
Action games involve the human player shoots at a series of opponents or objects. Traditional action games include *Space Invaders*, *Asteroids*, etc. The recent popular action games are *Doom*, *Quake, Descent*, *Half-Life* and *Unreal* that involve the human player to control a character in a virtual environment to save the world from the forces of evil by using deadly force.

***Adventure***
Adventure games are different from action games. They emphasize more on the story, plot and puzzle solving rather than simply catching, shooting, capturing, or escaping. The human player must solve puzzles while adventuring. Characters are usually able to carry objects, such as weapons, keys, tools, etc. The settings of these games often evoke a particular historical time period and place, such as the middle ages or Arthurian England, or are thematically related to content-based types such as Science Fiction, Fantasy, or Espionage. Examples of adventure games include *Adventure*, *Zork*, *Haunted House*, *Raiders of the Lost Ark* and *Superman*.

***Role Playing***
In role playing games, players can take on different types of character. The character's description may include specifics such as species, race, gender, and occupation, and may also include various abilities, such as strength and dexterity. In the virtual game world, the player goes on quests, fights monsters and improves the capability of the character on strength and magic. Example games include *Diablo*, *Dungeons & Dragons* and *Ultima*. Many role playing games are also networked games that allow more than one player to play and interact in the same game world over the network such as the Internet or LAN (Local Area Network). *Everquest* and *Ultima Online* are networked role playing games.

***Strategy***
Strategy games emphasize the use of strategy as opposed to fast action or the use of quick reflexes. Traditional strategy games include *Chess*, *Monopoly*, and *Othello*. In recent popular strategy games such as *Age of Empire*, *Warcraft* and *Close Combat*, the player can control many combat units to do battle against one or more opponents. In these games, the player needs to resolve the problem of resource allocation, and organization of defenses and attacks.

***Sports***
Sports games are adaptations of existing real-world sports or variations of them. The most popular sports games include *American Football*, *Baseball*, *Boxing*, *Fishing*, *Soccer*, *Tennis*, *Volleyball* and *Golf*.

***Simulation***
There are two types of simulation games: management simulation and training simulation. Management simulation games refer to those games in which players must manage the use of limited resources to build or expand some kind of community, institution or empire. Example management simulation games include *Railroad Tycoon*;

*SimAnt*, and *SimCity*. For training simulation games, it refers to games that attempt to simulate a realistic situation, for the purpose of training. Through the game simulation, it helps the player to develop some physical skills, such as steering as in driving and flight simulation games. Example training simulation games include *Police Trainer, Gunship* and *Flight Unlimited*.

## 1.2 Game Design

A computer game can be just a C application program. Figure 1.1 shows the architecture of a typical computer game. It consists of the following components: Input, Game Logic, Graphics/Sound Support, Game Output and Networking. They are briefly described as follows:

- Input – Users interact with the game program through input devices. Common input devices include keyboard, mouse or joystick.
- Game Logic – It implements the game logic or game code that handles most of the basic mechanics of game. Generally, before the game logic is developed, the story line on how the game is played and how the players should interact should be designed. Simple physics, networking support and animations should be planned. In some advanced games, artificial intelligence (AI) and collision detection are also implemented in this step.
- Graphics Rendering Engine - It has complicated code to efficiently identify and render the game objects and background from a two-dimensional (3-D) model of the environment. It supports transformation of objects that are moved, rotated and scaled when required.
- Graphics/Sound Drivers – The graphics drivers receive requests from the rendering engine to the graphics library using APIs. Windows APIs and Microsoft Foundation Classes (MFC) provide two-dimensional (2-D) graphics support for PCs. For supporting both 2-D and 3-D graphics, OpenGL and DirectX are the two most popular graphics libraries. DirectX also provides libraries for music and sound support.
- Game Output – The generated 2-D or 3-D graphics is output to the display. The generated sound effect or music is output to the sound card.
- Networking - It provides networking protocol support that allows several users in remote locations to play and interact in the same game environment. In a networked game environment, a server is needed to maintain information on which the virtual game world is supporting, communicates with game clients that are used by players to provide them with information about the shared environment. The server also needs to synchronize the information, and maintain the consistent scenes of the virtual game world among the networked clients.

When a game program begins execution, it should first initialize the memory, loads images and sound files, starts the graphics and set up variables such as scores. After initialization, the game logic then starts. When the game session ends, housekeeping is also needed to update sound effects, update and display scores, update data structures, etc. The user can end this session or go back to the game logic to start the game again.

Figure 1.1: Computer game architecture.

## 1.3 Story Line

To design a successful computer game, it is important to have a good story line, together with good 2-D and 3-D graphics, and sound effects to make the computer game seem realistic. The story line should be first developed before we start developing a game. To develop a story line, we need to determine the following:

- The type of the game – We need to classify the nature of the game into one of the categories that we have discussed in Section 1.1. Different types of game have different requirements, for example, role playing games requires realism in graphics and sound effects, whereas some strategy games emphasize on strategy rather than realism, and simplified display would be enough for such games.
- The goal of the game – We need to tell the player what to do to succeed in the game. In a space war game, the goal is to shoot down as many space invaders as possible. In a fighting game, the goal is to defeat or kill the opponent.
- The player's performance in the game – We need to decide how to rate the players for their performance. This should indicate how close they have achieved the goal of the game. In general, we use a total score as a player's performance indicator. In addition, when the player has achieved the goal, we also need to determine how to reward the player.
- The rules of the game – We need to tell the players how to play the game. Instructions on how to operate the keyboard or mouse in order to play the game should be explained.

## 1.4 Computer Graphics and Sound Effects

Visual effects of graphics and special sound effects are important to help establish the story line and provide the illustrations that make the player to feel part of the game. Many simple games rely only on 2-D graphics, which draws points, lines, and filled shapes such as rectangle and polygons in a plane. Graphics functions provided from Windows API (application Programming Interface) or Microsoft Foundation Class (MFC) library on Windows environment support 2-D graphics. Advanced computer games often require to handle 3-D objects. Microsoft DirectX supports advanced 3-D graphics on Windows platform. Instead of just supporting the Windows platform, OpenGL supports 2-D and 3-D graphics on both Unix and Windows environment.

Apart from 2-D and 3-D graphics, image display and manipulation is another important technique for game development. In many games, we can use a scanner to digitize pictures into images and used them as graphical elements in the display. We can also manipulate the images through scaling and rotation. Animation of images is another

popular technique employed in many games. Windows API and MFC library provides functions for image display and manipulation.

Sound is another important element that can make a computer game look realistic. Windows and MFC library only support the generation of only one tone at a time. However, with an optional sound card, most computer games can generate more complex sound effects. To support game development, Microsoft DirectX provides the DirectX Audio subsystem that supports music and stereo sound effects. OpenGL library does not provide any support for sound effects.

In this section, we briefly review some of the graphics libraries including Windows API, MFC library, DirectX and OpenGL.

**Windows API and MFC Library**

Windows is a very important subsystem in Microsoft's operating system. It makes Microsoft's 32 bit Windows API available to application programs. Windows API contains functions for Graphics Device Interface (GDI) that allows users to draw and write in a window. The GDI functions let users display graphics in Windows using a device context object such as a display or a printer. Device context is designed to isolate a Windows program from the physical output device, so that when you call GDI functions for all graphics output, it accesses the specific device driver. GDI provides vector drawing functions that can draw graphical objects such as lines, rectangles, ellipses and polygons; text output functions to display text in a window; bitmap manipulation functions to display and manipulate images; and palette management functions to exploit the colors.

MFC library provides a set of functions to control text and graphics output. It is object-oriented and the key class is the CDC class that defines a class of device context objects. The CDC object provides member functions for working with a device context. Similar to Windows GDI, the member functions provide operations to support drawing of lines, simple shapes, ellipses and polygons, drawing of text and working with fonts, colors and palettes. In addition, member functions are also provided for working with viewport, working with regions, mapping and clipping.

**DirectX**

The goal of DirectX is to make Microsoft Windows a desirable platform for game development. It aims at shifting the burden of hardware support from the game developers to the hardware manufacturers, who are more qualified to write drivers for their products than the game developers. However, DirectX is not a game-creation package. It only aids in the development of applications through the use of APIs designed to interface directly with the computer’s hardware. If the hardware is equipped with DirectX drivers, access can be granted to the accelerated functions, which that device provides. If no accelerated functions exist, DirectX will emulate them. Hence, the programmer can continue to work on a consistent interface without worrying about things such as hardware features. If a feature does not exist on the card, it is likely that the feature will work through DirectX’s emulation functions.

Thus, DirectX is a set of low-level application programming interfaces for creating games and other high-performance multimedia applications. It includes support for 2-D and 3-D graphics, sound effects and music, input devices, and support for networked applications such as multiplayer games. DirectX 8 has the following major components:

- DirectX Graphics - It is a complete 3-D graphics system.
- DirectX Audio - It includes sound and music systems that provides a complete system for implementing a dynamic soundtrack.
- DirectPlay - It is a set of tools that simplify communications across the networks, the Internet or modems. The tools allow game players to find game sessions easily to help manage the flow between servers and players.
- DirectInput - It provides the game developer with an interface to myriad input devices, such as keyboards, mouse and joysticks.

**OpenGL**

OpenGL was developed by Silicon Graphics Inc. (SGI) as a multi-purpose, platform-independent graphics API. The development of OpenGL has been overseen by the OpenGL Architecture Review Board (ARB), which is made up of major graphics vendors and manuafacturers since 1992. ARB is responsible for establishing and maintaining the OpenGL specification. The current release of OpenGL is at version 1.3. Unlike DirectX which is now at its eighth version, the OpenGL specification is quite stable which does not get updated often.

OpenGL is a collection of several hundred functions providing access to all the features offered by the graphics hardware. This includes 2-D image scaling, rendering 3-D objects including spheres, cylinders, and disks, coloring, lighting, blending, and so on. The API is a powerful, low-level rendering and modeling software library available on all major platforms. It is designed for use in any graphics applications, from games to modeling to Computer Aided Design (CAD). Many computer games such as Quake 3 have used OpenGL for their core graphics-rendering engines.

Under Windows environment, OpenGL provides an alternative to using the Graphics Device Interface (GDI). GDI is designed to make the graphics hardware entirely invisible to Windows programmers. However, the layers of abstraction that help programmers avoid dealing with device-specific issues have caused applications lacking the speed required for games. In OpenGL, GDI can be bypassed entirely. OpenGL API can access directly with graphics hardware.

OpenGL does not directly support any form of windowing menus or input. The OpenGL Utility Toolkit (GLUT) is a set of support libraries available to provide basic functionality in many areas, while remaining platform independent. For instance, GLUT-based applications can be easily ported from Windows to Unix. GLUT is easy to use and learn. Although it does not provide all the functionality that the operating system offers, it works well for simple applications.

## 1.5 Programming Environment

Her, we intend to use computer games as programming examples to illustrate the different concepts in C such as branching, looping, functions, arrays, strings, structures and file I/O. Advanced computer games such as role playing games, adventure games and simulation games require complex 3-D graphics to make the virtual game world realistic. As such, only traditional, simple games that only require simple 2-D graphics such as drawing lines, rectangles and polygons are discussed.

Windows API, MFC library and DirectX are only available in Windows platform, while OpenGL is an open source that can be available in both Windows and Unix platforms. Here, we have chosen OpenGL and GLUT as the graphics driver for supporting different 2-D and 3-D graphics API for the developed game programs.

Microsoft's Visual C/C++ is used for the development of the game programs in the Windows environment. OpenGL and GLUT are required to be installed within the Microsoft's Visual C/C++ environment. We will also discuss the installation of OpenGL and GLUT for Microsoft's Visual C/C++.