KEMI-TORNIO UNIVERSITY OF APPLIED SCIENCES

3D Mobile Game Engine Development

Liu Danyang

Bachelor's thesis of Degree Programme in Business Information Technology

Bachelor of Business Administration

TORNIO 2012

ABSTRACT

Liu Danyang. 2012. 3D Mobile Game Engine Development. Kemi-Tornio University of Applied Sciences. Business and Culture. Pages 73.

With the rapid development of mobile games, more rich and colorful mobile games are demanded. While during the development of mobile games, a good mobile game engine is a key technology, since it can integrate some important functions in a framework. In my thesis, the objective is to design the framework of 3D mobile game engine. For completing the research, I put forward specific three research questions, and through understanding and learning the knowledge of professional technologies and the structure of 3D mobile game engine I find answers to the questions.

With a keen interesting as motivation, I research the basic frame structures and process of 3D mobile game engine design. The development of 3D mobile game engine design is based on J2ME platform. Specifically, my thesis introduces a framework, main components and operation principles of 3D mobile game engine.

During the designing process, the thesis expounds detailed architecture, specific engine layers, components and relative functions of 3D mobile game engine. Through the M3G and OpenGL API, the mobile game engine reasonably implants some 3D graphic technologies. In the implementation chapter, the thesis expresses some main functions around a scene management engine, which is a central sub engine in the whole mobile game engine structure.

This thesis can be perceived as a blueprint. The specific engine framework achieved practical values. However, with the limitations of mobile devices, implementing an integrative mobile game engine entity needs further researches and practices in future.

Key Words: Mobile Games, 3D Mobile Game Engine, J2ME, M3G, 3D Graphics.

CONTENTS

ABSTRACT

ABBREVIATIONS

FIGURES

ABBREVIATIONS

3D	3 (Three) Dimension
J2ME	Java 2 Platform Micro Edition
CLDC	Connected Limited Device Configuration
CDC	Connected Device Configuration
MIDP	Mobile Information Device Profile
API	Application Programming Interface
JDK	Java Development Kit
WTK	Wireless Tool Kit
IDE	Integrated Developing Environment
M3G	Mobile 3D Graphic API
I/O	Input/ Output
MVC	Model View Controller
AI	Artificial Intelligence
FSM	Finite-State Machine

FIGURES

1 INTRODUCTION

Today, the mobile games became common entertainments based on mobile devices. Behind the colorful mobile games, the mobile game engine plays an important role to boost the development of mobile games. My thesis work illustrates an architecture and operation processes of the main components in the 3D mobile game engine, and some parts of relative functional implementations.

1.1 Motivation and Background

Gaming is an important entertainment project in several people's daily life. At the same time, gaming has boomed the development of the game market. Game platform has experienced a lot of technical transformations, from game machines, video games, PC games, to mobile games. With the development of new technology and improvement of people's living standard, the mobile phone has become popular. Undoubtedly the mobile phones have evolved into some people's essential communication tools. Therefore, the games that are based on the mobile platform also have a huge market potential.

The game engine is a processing game basic technology platform. Using the game engine can shorten the game development time greatly. Therefore, the engine is peculiar in the game technology. And thus in the mobile game development process, the engine design is very important. A reasonable and efficient engine can simplify the design process, improve the efficiency of development and enhance the maintainability and expansibility of the mobile phone game. (Root & Donnelly & Yeganov 2008)

Currently, the 3D mobile game engine has developed into a set of complex systems constituting multiple subsystems. The 3D mobile game engine covers almost all

important projects in the development process, such as models building, animation lighting, particle effects, collision detection, file management, network characteristics and professional editing and plug-in unit (Zerbst & Duevel 2004, 35-38.) The main targets are to simplify the complexity of designing games, shorten the developing time, reduce production cost and increase production scales. Therefore, the specific objective is to design the framework of 3D mobile game engine in my thesis work.

1.2 Structure of the Thesis

This thesis researches the theme of research is 3D mobile game engine development. The thesis work is divided into 7 chapters.

Chapter 2 describes the research topic and discussed the 3D mobile game engine development. In addition, three research questions are put forward. The expected output in my thesis work is also discussed. Chapter 3 focuses on research methodologies, and it includes the detailing of the research method, i.e. the constructive research approach, and data collection and data analysis processes.

Chapter 4 is the Background Information. It consists of three sub- chapters of the background information, J2ME technology and platform, 3D space principle and rendering technologies, and API Graphic system. Chapter 5 and Chapter 6 are about the design and implementation of 3D mobile game engine respectively. Specifically, the design chapter illustrates 3 different kinds of sub engine functions and 3D effects application according to architecture of mobile game engine. In the implementation chapter, some main functions around the scene management engine are illustrated, which functions as a central implementing sub engine. Finally, the Chapter 7 concludes the content of the research about the 3D mobile game engine development, and evaluates the research limits and deficiency, and expects further implementation of 3D mobile game engine.

2 RESEARCH TOPIC, QUESTIONS AND EXPECTED OUTPUT

2.1 Research Topic

My research topic is 3D Mobile Game Engine Development. The research and development of 3D mobile Game Engine Development are based on Java 2 Platform Micro Edition (J2ME).

During the development of mobile games, designing the structure of mobile game engine is the core technology, because it plays a conclusive role in mobile games quality. Mobile game engine is a main program that can control every game function. The engine is at the core status of the whole system. Mobile game engine combines organically all elements in game environment together and commands orderly these elements work in the backgrounders.

Mobile game engine is a framework of mobile games. The mobile game developers can fill into game coding with their ideas of game designs based on mobile game engine. Due to the development of mobile games is a process, which costs a large of time and money. Therefore, an increasing number of developers of mobile game tend to use some existing mobile game engines. A reasonable and efficient engine can not only simplify the design process, but also shorten the game development time greatly. Sequentially improve the efficiency of development and enhance the maintainability and expansibility of the mobile phone game. (Root & Donnelly & Yeganov 2008)

2.2 Research Questions

In the thesis work, the general aim is to discuss 3D mobile game engine development, and the objective is to design the framework of 3D mobile game engine. During the research process, there specific research questions are put forward and they will be answered in thesis.

Q1. What is the fundamental structure of J2ME technology?

J2ME technology as a basic developing platform technique, it plays a significant role in mobile game development. J2ME is not only an essential platform in JAVA environment, but also a core technology for my thesis work. To start with, I need to have a clear understanding of the structure of J2ME technology. Operating J2ME platform needs some supports and environment. Besides, J2ME also includes different types of configurations, profiles and interfaces. The availability, compatibility and efficiency are considered balanced when J2ME platform was adapted into my thesis work.

Q2. How can J2ME platform be combined with the 3D render technology?

Due to the face that my thesis researches the 3D mobile game engine development, it should achieve the 3D effects and technologies through mobile game engine design. J2ME just only establishes a fundamental structure of mobile game engine, and 3D effects rendering designs needs professional 3D render technologies to get achieved. Finding answers to the question of how can 3D render technologies be adapted into the structure of the mobile game engine designed by J2ME platform is important in the research process.

Q3. How can 3D mobile game engine framework be designed?

A practical 3D mobile game engine contains complex structures and operation principles. Designing an integrative engine needs a large amount of knowledge, technologies and time. In the thesis work, the engine architecture and sub engines are described in detail. Depending on specific designs of 3D mobile game engine, they are supported by feasible design structures, theory structures and corresponding operation environment with the following pages.

2.3 Expected Research Output

The final research result is a 3D mobile game engine framework in my thesis paper, which is a representative engine framework in a whole mobile game system.

An achievement of the final implementation of 3D mobile game engine is assessed from three aspects, integrity of framework, expected 3D render effect and function characteristics. Firstly, the mobile game engine is a function setting, which can flexibly realize some functions in mobile games. Therefore, I concentrate on designing one type of mobile game engine models which is responsible for different functions. The mobile game engine models are shaped into an integrative engine framework. In addition, through some kinds of methods and technologies, the engine framework can achieve 3D rendering effects. Finally, different types of engine model also achieve expected functions. Therefore, a simple and mini type 3D mobile game engine framework will emerge in thesis work, and some additional implementation examples will describe relative engine design structures in detail.

2.4 Theoretical Framework

The theoretical framework in this thesis work relies on “Model View Controller (MVC) was a technique devised in the late 1970s by the SmallTalk community. It simplifies the development of user interfaces by clearly separating code into three layers: the model, the view and the controller” (Walnes & Abrahamian & Cannon-Brookes 2004, 83). The goal of MVC application is that it simplifies the follow-up modifying and expands for some programs, and makes those programs able to be reused possible. Sequentially, the program structures are more intuitional. There are some software design frames that can provide direct and rapid MVC frameworks for small and medium program application development. Therefore, I use the MVC to design function frameworks of the 3D mobile game engine.

The reason I chose the MVC is that it can simplify some specific sub engine structure designs. Specifically, the MVC can make one program expressed by different forms. In my engine design, the Controller combines Model and View with different design styles to satisfy different types of mobile game demands. The detailed designs are described in design and implementation chapters in the thesis. Although using the MVC can bring a lot of benefits in my thesis work, it also has a shortage. The MVC does not have a clear definition, so it is hard to be understood completely, and needs much time to consider.

Except for the MVC technique, a flow diagram method is used in my thesis work. This method can be expressed by different diagrams, such as logical process flowchart, activity flowchart and functional flowchart. These different types of flowchart in my thesis, they provide graphic representation of the complex flows in some specific engine frameworks, and reduce ambiguity and confusion in technical processes of 3D mobile game engine (William H. & Vernon H & Amy & Courtland s 1989, 21-22.)

3 RESEARCH METHODOLOGIES

3.1 Research Method

This thesis work is a constructive research with the objective of 3D mobile game engine development. “Research is a way of thinking: understanding and formulating guiding principles that govern a particular procedure; and development and testing new theories that contribute to the advancement of you practice and profession” (Kumar 2011, 1). For my research work, I want to understand the principles of 3D Mobile Game Engine Development and keep the theories about main structure of 3D mobile game engine, which include different function models, to design a 3D mobile game engine framework.

In the researching on 3D mobile game engine development, I use theories and techniques developed by research methodologies to consolidate, improve, develop refine and advance my research work (Kumar 2011, 4). Constructive research is the best choice for my research work. “The constructive research approach is a research procedure for producing innovative constructive, intended to solved problems faced in the real world and, by that means, to make contribution to the theory of the discipline in which it is applied” (Lukka 2003, 83-101). During my research process, I wanted to get a further understanding and developing about structure and functions of 3D mobile game engine, through answering three research questions in my research work. In addition, Järvinen (2001, 88) suggests that “the constructive research builds a new innovation and this process is based on existing knowledge or new technical advancement.” Therefore, I design a framework of 3D mobile game engine as an expected result. The framework of 3D mobile game engine needs some basic theories, technologies and software in mobile game engine domain, such as J2ME technique, 3D render system technologies and API graphic system.

3.2 Data Collection and Analysis

“Data collection requires considerable knowledge and skills, this stage of the research process is very important because once the data is collected, you cannot return to an earlier step to correct decisions that led to limitations in the study” (Hair & Money & Samouel & Page 2007, 16). I collected secondary data by reading related materials and resources, in order to decide the development direction and research scope. Data collection should be considered not only logical and consistent for integrative research, but also available for each specific research part. The data and materials come from Ebrary of Kemi-Tornio University of Applied Sciences and E-books.

The way of data analysis depends upon two factors: type of information (descriptive, quantitative, qualitative or attitudinal), and the way to communicate what finding to readers (Kumar 2011, 26). During this process, I processed the data analysis following three steps. At the first step, through analyzing my research work, I divided my research work into several categories based on research contents, and select the relative information from the collected resources according to different parts of research content. Besides, I analyze the resources to keep consistency and extract related contents that I need to support the research. Finally, I expect to find an information structure to keep the integrity of the whole research work. Therefore, it is advanced to design the detailed structure and implementation in my research, following the comprehensive understanding of theories and technologies.

4 BACKGROUND INFORMATION

4.1 J2ME Technology and Platform

During my thesis work, the J2ME is the core technology for design of 3D mobile game engine. J2ME is Java 2 Platform Micro Edition. J2ME is a development platform for the embedded equipment, which made in SUN Company (Sun Microsystems Company, 1986). J2ME is one of the important development platforms of JAVA technology.

4.1.1 Structure of J2ME

In the researching and designing of 3D mobile game engine, the J2ME is a basic development platform. At present, J2ME platform is considered to be one of the mainstreams of the cellular phone game platforms, and it is widely accepted with the developer, equipment manufacturers, network communication companies and consumers. “Configurations and profiles are the main elements that comprise J2ME’s modular design. These two elements enable support for the plethora of devices the J2ME supports.” (Piroumian 2002, 2)

Figure 1 illustrates a set of layers that support a basic running environment with the core Java libraries (Piroumian 2002, 4)

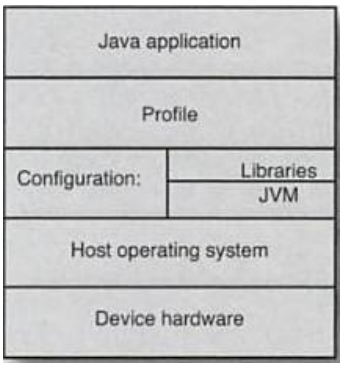


Figure 1. The J2ME platform consists of a set of layers

A set of system – level application programming interface (API) in a configuration, and a set of application – level API in a profile

Configuration

"A J2ME configuration defines a minimum Java platform for a family of devices. Members of a given family all have similar requirements for memory and processing power. A configuration also specifies a minimum set of features for a category of devices." (Piroumian 2002, 2)

Profile

"The profile specifies the application-level interface for a particular class of devices. A profile implementation consists of a set of Java class libraries that provide this application-level interface. Thus, a profile theoretically could specify all kinds of functionality and services." (Piroumian 2002, 2)

Two categories of pervasive devices are the following (Muchow 2002, 4):

- Personal, intermittently connect mobile devices- supported by the Connected Limited Device Configuration (CLDC)

- Constantly connected network devices – supported by the Connected Device Configuration (CDC)

4.1.2 MIDlets of J2M2 Platform

J2ME is a slimmed-down version of Java targeted devices that have limited memory, display, and processing power. MIDP (Mobile Information Device Profile) is a set of interface of Java Application program. It is referred to as API (Application Programming Interface). There is a specific application development for mobile devices using an application programming interface (API) known as the Mobile

Information Device Profile (MIDP). Applications written for this API are affectionately referred to as MIDlets. (Muchow 2002, 6)

Three software tools are needed to develop MIDlets:

·Java Development Kit (JDK)

·Connected Limited Device Configuration (CLDC)

·Mobile Information Device Profile (MIDP)

4.1.3 The Tools and Environment of J2ME

The J2ME is the core platform for design of 3D mobile game engine. In addition, the design and application of the J2ME needs to environment and tools to support as follow,

1. JDK (Java Development Kit)

J2ME is one part of Java, so it should support the development of JDK. The JDK provides the Java source code compiler and a utility to create Java Archive (JAR) files (Muchow 2002, 15).

2. Configuration environment variables

Due to install some Java setups, I should set up some variables to satisfy Java Configuration environment.

3. WTK (Wireless ToolKit)

WTK also developed by SUN Company. This tool to develop for the game developer and it can protect the integrative generation tools, demo program and simulators.

The Sun JavaTM Wireless Toolkit for CLDC supports the creation of MIDP applications with the following main features (SUN Microsystems 2012):

■ Building and packaging: writing the source code and the toolkit takes care of the rest. With push button, the toolkit compiles the source code, preverifies the class files, and packages a MIDlet suite.

■ Running and monitoring: running a MIDlet suite directly in the emulator or install it using a process that resembles application installation on a real device. A memory monitor, network monitor, and method profiler are provided to analyze the operation of your MIDlets.

■ MIDlet suite signing: The toolkit contains tools for cryptographically signing MIDlet suites. This is useful for testing the operation of MIDlets in different protection domains.

4. The development tool – IDE (Integrated Developing Environment)

IDE is comprehensive tool. The tool is meant for designing all functions required program designing together with organically way, and put all functions in one graphical interface. The tool offers an integrative service for program developer. IDE include mainly three kinds of software, JBuider, Netbeans and Eclipse.

5. Eclipse Software Application

In my thesis design, I focus on Eclipse application. And then, I should set up Eclipse for applying J2ME. Finally, establish a mobile game engine design in eclipse environment.

4.2 3D Principles and Vector Movement

Moving of an object in 3D space is a basic action in mobile games design. This chapter introduces not only several basic 3D space concepts and position, but also basic moving principle and calculating elements.

4.2.1 3D Space Concept

According to design of 3D mobile game engine, I should establish some basic 3D principles, a simple positioning and 3D calculating to complete the concrete analysis for the 3D world objects realization.

Coordinate System

"The standard 3-tuple is written as (x, y, z). The components of the 3-tuple specify the location of a point in space relative to an origin. The components are referred to as the Cartesian Coordinates of the point" (Eberly 2007, 8).

Figure 2 describes the coordinate system in 3D space.

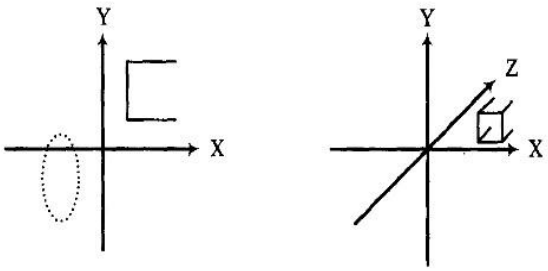


Figure 2. The coordinate system

In the first image, it establishes a 2D structure ant it just has X- axis and Y- axis. But in the second image, it adds the Z- axis. Therefore, the second image founds the 3D space.

“The important coordinate systems, called space, are Cartesian space (everything else is built on top of this), model space (or object space), world space, view space (or camera space or eye space), clip space (or projection space or homogenous space), and window space” (Eberly 2007, 8-9).

Transformations

If designers want to construct functions that map points in 3D space, designers will take an object built in model space and place it in word space. Transformation is an important element in 3D actions. Transformation describes the 3D space objects finishing all the transforms (scaling, rotating, moving). The Transformations include Liner Transformations, Affine Transformations and Project Transformations (Eberly 2007, 18.)

Figure 3 describes the process of transformations in 3D space.

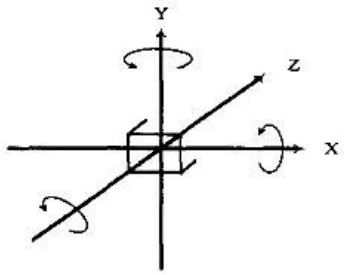


Figure 3. The transformations in 3D space

The principle of transformation in Figure 3 is a space transforming. There are specifically coordinates in every point of object, such as a cube. When the point in object start to transform, the coordinates will transform following X- axis, Y- axis and Z- axis.

4.2.2 Moving and Calculating in 3D Space

The collision detection is the core technology in physical model engine, and it also involves with other engine working. With collision detection application, it should be supported by some basic knowledge in 3D space.

The moving of an object in 3D space, the task should process a reasonable physics calculation. According to the vector in 3D calculation from Zerbst and Duevel's (2004, 132) principle of 3D engine design, generally, when an object is moving in 3D space, it needs three different velocity from physics calculation, Vx, Vy, and Vz, and describes the movement of object through those velocity. While the simple calculating to complete a single movement of an object, the calculating involves three vectors as follow,

Position (Coordinate) Vector. It is an important vector in 3D calculation. The position vector can save the current position of object(x, y, z coordinate) through keeping the path of moving. It provides accurate position and object for collision detection and rendering.

Velocity Vector. It defines how many units are needed to move next time for an object and corresponding coordinates. Therefore, the Velocity Vector also needs compute Vx, Vy and Vz.

Acceleration Vector. It has a same principle with Velocity Vector, but different from that the Acceleration Vector works for Velocity Vector, not in coordinate. It works for every unit and defines an Acceleration Vector.

4.3 Application Programming Interface (API) Graphic System

In the development of the 3D mobile game engine, 3D graphic technology is very necessary, especially in the J2ME platform. In the J2ME platform, there are different types of Java application programming interface packages to achieve the edit and

design 3D graphic process in mobile, such as Mobile 3D Graphic (M3G) API, OpenGL and OpenGL ES. Specifically, the API can give some mobile game engine platforms the ability to show 3D effect graphics or transform from other 3D graphic technologies.

4.3.1 Mobile 3D Graphic (M3G) API

M3G (Mobile 3D Graphic API for J2ME, also known as JSR 184), "provides for JAVA programmers an efficient and compact interface for managing and rendering 3D scenes" (Pulli 2008, 285) and gives MIDlets the ability to show 3D content.

It is a scaled-down version of the desktop Java platform 3D API, which means it knows how to render scenes that are described as a hierarchy of groups and objects. The API includes all the classes and methods developers need to create scenes from the ground up (Knudsen 2007, Chapter 13).

Position of M3G

Figure 4 shows an overview of the mobile Java software architecture and the positioning of Mobile 3D Graphic (M3G) API.

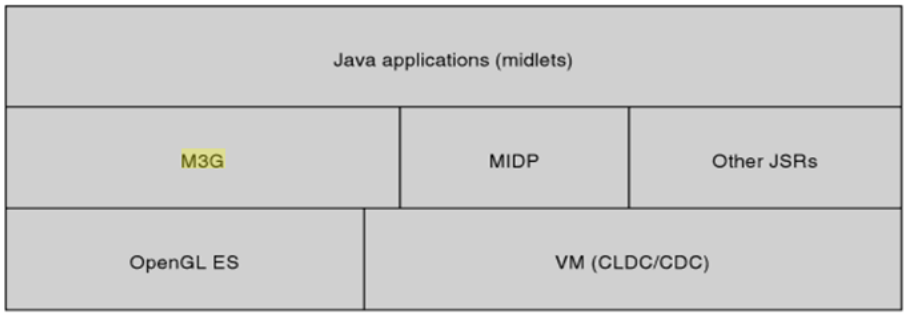


Figure 4. The positioning of M3G in Mobile JAVA Software Architecture

“M3G is a completely new high-level API that borrows ideas from previous APIs such as Java 3D and Open Inventor” (Pulli 2008, 19). It consists of nodes that encapsulate 3D graphics elements. The nodes can be connected to form a scene graph representing the graphics objects and their relationships. M3G is designed so that it can be efficiently implemented on top of OpenGL ES renderer (Pulli 2008, 19.)

Structure of M3G

The core functions of M3G include three classes, i.e. Loader, World, and Graphics 3D. They are responsible for completing to establishing game scenes and rendering.

The Loader Class can lead different types of images and objects into the mobile. Especially, it can take .m3g files and objects into a mobile and inherit relative data from Object3D of M3G, such as Camera data and mode information. Even though, .m3g file can be used directly in mobile program, but it will occupy a lot of storages, especially animation files. Therefore, the low complexity is required in loader class. “To satisfy this goal, the engines underlying the OpenGL ES and M3G APIs were required to be implementable, in software, in fewer than 50kb and 150kb, respectively. The key tools for reaching these targets were removal of redundant and seldom – used features” (Pulli 2008, 17).

The World Class is the most significant class in M3G. It is a top node (Pulli 2008, 270), because the game scene consists of nodes which has relative inheritably relationships. An integrative game scene design is combining all of nodes through World.

The Figure 5 as an example to explain the foundation of scene tree

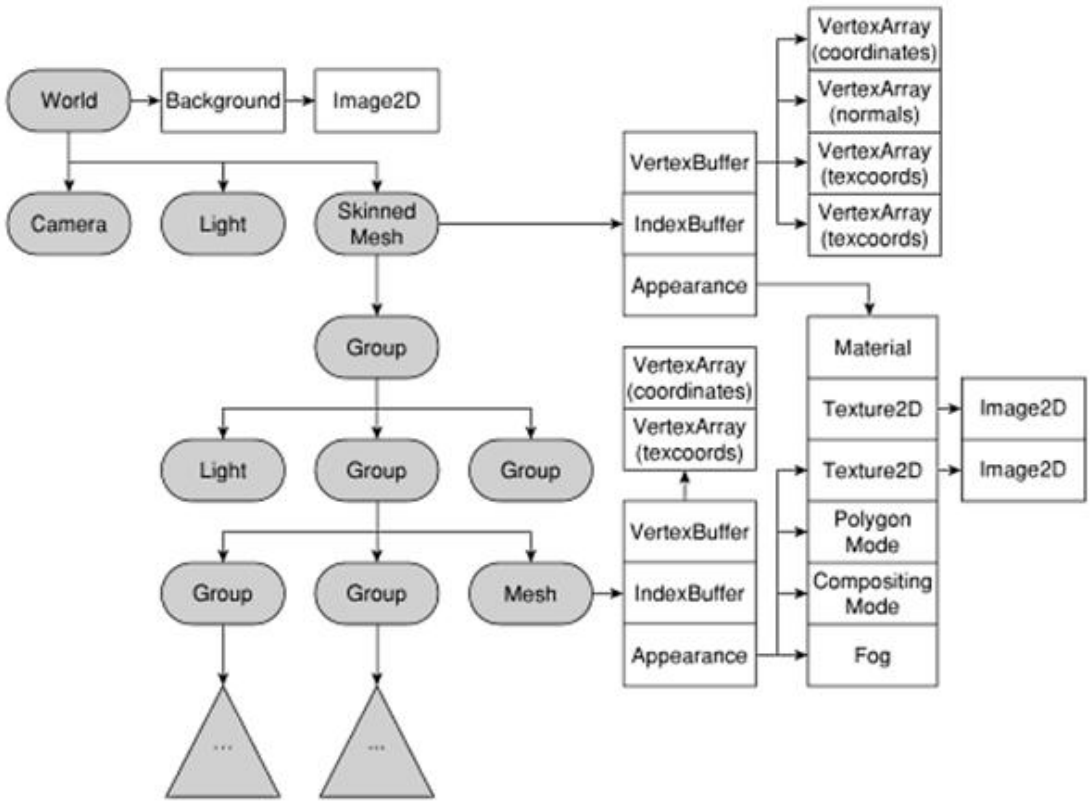


Figure 5. An Example of Scene Tree (Pulli 2008, 270)

According to Figure 5, the World Class provides a simple way to cover a 3D scene design and relative information. It also includes camera, light, mesh and other classes.

Features and Functions of M3G

The application of M3G, it has two developing modes, immediate mode and retain mode (Pulli 2008, 267-276). Specifically, for immediate mode, the game programmers should render every frame animation, in order to achieve a more effective program, but codes are more complex. For retaining mode, the game programmers just set ready key frames animations, it needs to be noted that one of the key features of M3G is its key frame animation engine. It can animate any property of any object by sampling a user-specified animation curve (Pulli 2008, 267-276) and

M3G can complete automatic the rest rendering.

Besides, “Minimal fragmentation lets content developers work on familiar ground. Therefore, both OpenGL ES and M3G attempt to strictly mandate feature, keeping the number of optional features as small as possible” (Pulli 2008, 17). Except for the two main functions above, the M3G also can control and set object properties, such as position, shape, size and other properties.

4.3.2 Open GL ES

“OpenGL ES is a compact version of the well-known OpenGL graphics standard. It is a low-level rendering API adapted for embedded system” (Pulli 2008, 18). The OpenGL ES development aims at mobile device. “Since its launch, support for OpenGL ES has grown quickly, with most of the major cell phone and PDA manufacturers adding support for it, initially with software-only implementation” (Astle & Durnil 2004, 36).

“The latest version, OpenGL ES 2.0, provides a completely revamped API, and support for a high-level sanding language (GLSL ES): it replaces several stages of the traditional fixed-function graphic pipeline with programmable vertex and fragment sharers, and is therefore not backward-compatible with the 1.x series” (Pulli 2008, 18). OpenGL ES 2.0 can greatly improve the rendering effectiveness with different electronic devices, and realize the comprehensive programmable 3D graphics in embedded system.

5 3D MOBILE GAME ENGINE DESIGN

The Game Engine originated from PC game development, while the 3D mobile game engine is based on basic technologies and frame structure of PC games engine. An engine is comparable to the heart of a machine. It means the core status the in whole system. During the development of mobile games, the core technology is mobile game engine, because it relates the quality of a mobile game.

The mobile game engine can be understood as exclusive tool software for developing mobile games. The contents of game stories, game levels, the quality of graphics and voice, and operation are controlled directly by game engine. Mobile game engine is combined close all elements in mobile game environment together and commands orderly these elements to work in the backgrounder.

The sub-chapter to follow is an important part in my thesis work since I focus on introducing the architecture of 3D mobile game engine design and specifically component structure and technologies.

5.1 3D Mobile Game Engine Architecture

Architecture of developing mobile game should cover every function controlling models. Considering Rilkey Scan's (2004, 7) ideas about game architecture, Zerbst and Duevel's (2004, 37) principle of 3D engine design, and combined with my understanding of mobile game engine, I designed the hierarchical structure of mobile game engine as shown in Figure 6,

Figure 6 illustrates the hierarchical structure of mobile game engine and position relationship of each component.

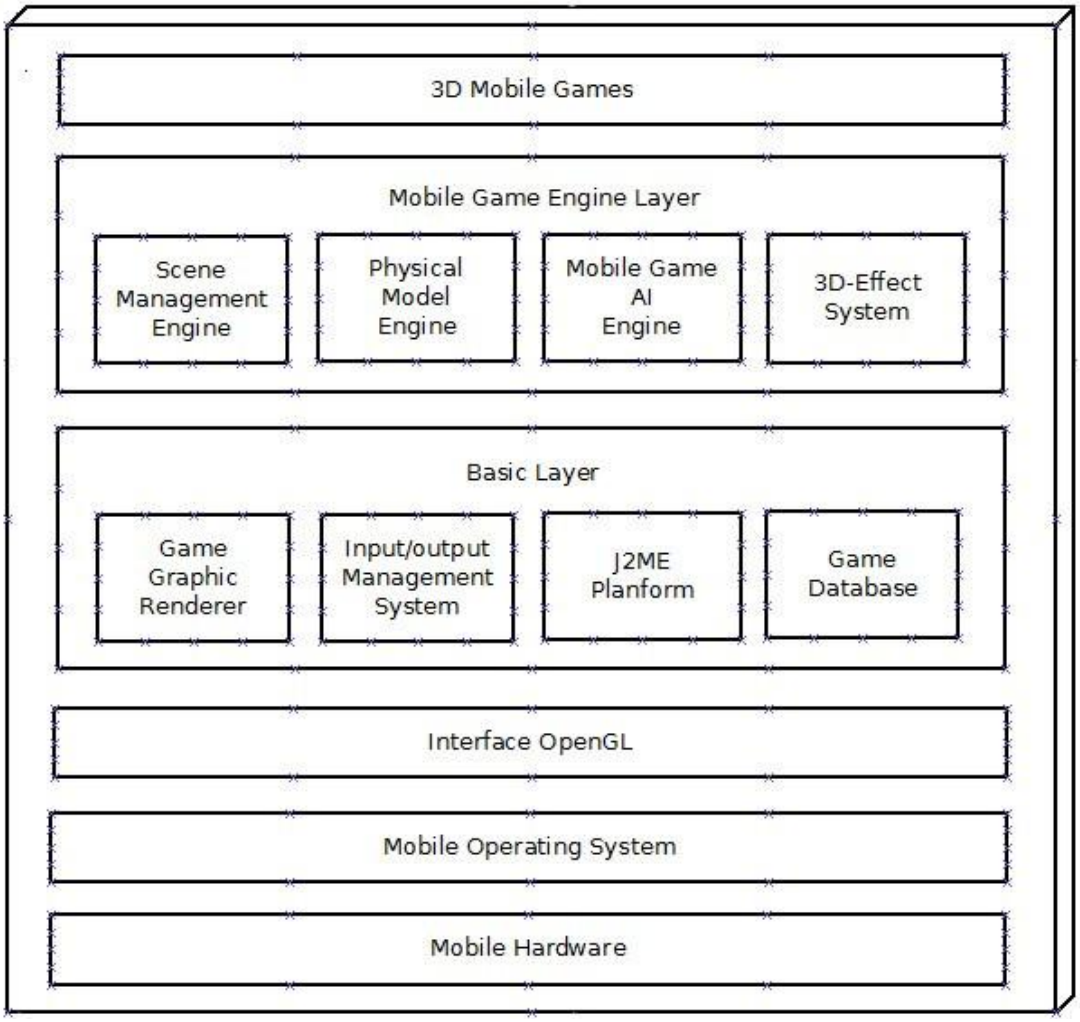


Figure 6. Hierarchical structure of mobile game engine

According to the hierarchical structure of mobile game engine, it involves the relative structures and technologies. At the bottom layer, there are mobile hardware devices to support the basic operating environment for mobile. In their daily life, people could have chosen different mobile operation system, such as iphone operation system, android operation system, and other operation systems. In the development of mobile game engine, the operation system is a necessary component. “OpenGL has an extension mechanism; this allows implementers of the library to add extra functionality. Applications can query the availability of specific extensions at runtime, making it possible for programs to adapt to different hardware.” (Gustavsson 2008). Even although the game engine technologies keep developing and updating, but the high level layers technologies always need a unified interface criteria to support different applications and adapt to mobile hardware.

The mobile game engine layer and basic layer are core structures in my thesis work. The basic layer is a foundation of mobile game engine. It contains the basic components, platforms, technologies and supports, which high level game engines needed. In basic layer design, I expect the architecture of mobile game engine can achieve a benefit that can adjust and shorten automatic difference with different mobile operation systems and hardware.

Mobile game engine layers cater to mobile game developers. The mobile game engine can provide different function models for developers. In my thesis work, I introduce and design a type of general mobile game engine layer, which includes scene management engine, physical model engine and game AI engine. While the 3D effects system is the same important as those types of function model engines. Due to that the function model engines and 3D effects technologies work of together, they can play a core role in game engine. Specifically, the mobile game engine layer embodies some concreted working of components. I will make detailed descriptions in the following pages and design the particular frameworks of mobile game engine.

“Most games share certain common infrastructure components. These components and the relationships between them comprise the architecture of the game each game’s architecture is different, but these common components provide an overview of how games work on the in the inside.” (Rilkey 2004, 7) Therefore, the mobile game engine has some common components provide a working overview in the inside. According to analysis about operation principle of mobile games, I design a diagram to describe relationship among the infrastructure component and specific engines as follow in Figure 7,

Figure 7 illustrates the structure of infrastructure component and relationship in mobile game engine.

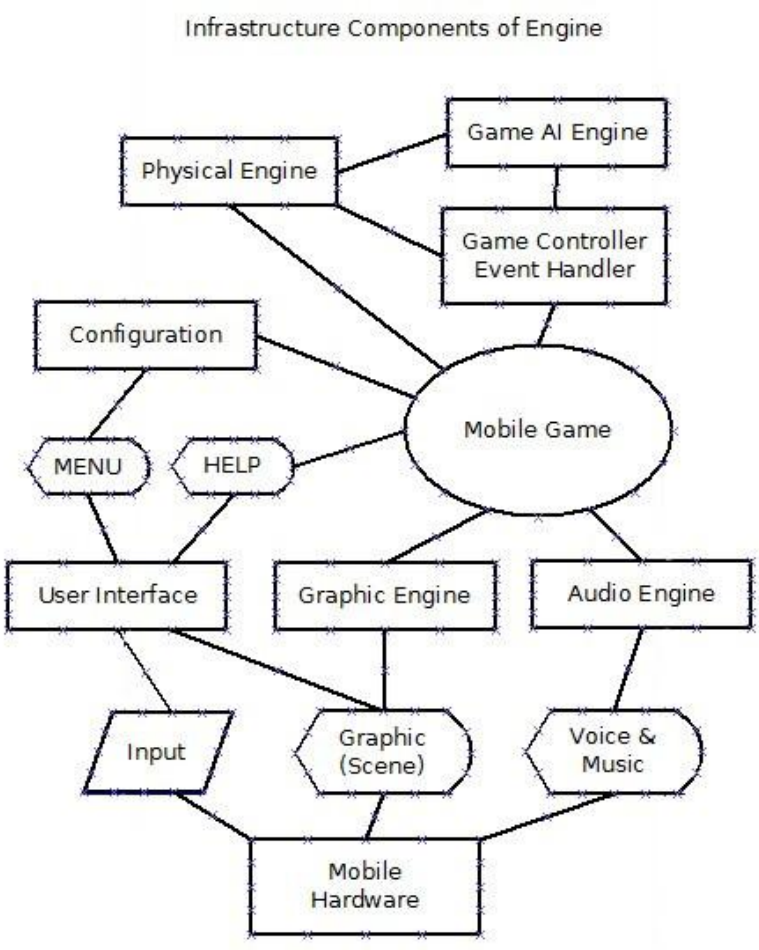


Figure 7. Structure of infrastructure component of mobile game engine

According to the Figure 7, it is easy to understand structure of infrastructure component of mobile game engine. Behind the every function or display, different types of engine play different important role in whole system.

User Interface

About the most of mobile games, user interface is a simple and basic interface layer. During the designing of my thesis work, I particular introduce the OpenGL as the interface criteria. Generally, the user interface can monitor the states and data of user input, and save those data and information into game database. On the other hand, the user interface also can display the menu and other option for users. Moreover, the input function involves game control system, and then I explain specifically in Game AI engine design chapter.

Configuration

The configuration system is a setting management system. It is responsible for reading the configuration files and executing game setting. Configuration system involves many functions application in mobile games. When the other sub engines are operating, it will ask the configuration system to execute corresponding configuration.

There is an example for controlling system, before starting the mobile games, the player can set and edit the shortcut key for their control style, and then configuration system will save the controlling configuration. When the shortcut key benefits are achieved in game actions, the configuration system will be asked to execute the player's personal control configurations. Finally, players will get the expected results through automatic asking and executing configuration. Configuration system manages all configurations in mobile games, such as resolution configurations, graphic quality configurations, sound effects (Pardew & Pugh & Nunamaker 2004, 17) configurations and other configurations. Configuration system is required for to adapt different types of mobile hardware. At the same time, it can save the players' setting to satisfy their favorite and personal game style.

Graphic Engine & Audio Engine

The graphic and audio performances are the most intuitive sense components in mobile games. They are core elements to attract players' eyes and enjoys. Graphic engine is easy to understand. "Graphic engines are components for drawing output to the screen. Images on the screen are the primary form of feedback that users receive during game play, so the component that produces these images – the graphic engine- is one of the most critical portions of any game" (Rilkey 2004, 11). Images and sounds are two simple elements to found a scene, therefore, the graphic engine and audio engine are important components in Scene Management Engine as two sub engines. Designing 3D mobile game engine should involve 3D graphic effects technologies. Renderer is one of important functions in graphic engine. At same time, rendering technology is more difficult and complex part. The rendering technology cover some professional acknowledge, calculation and 3D techniques. It also will directly affect the 3D graphic quality. The differences among different 3D mobile games are attractive styles of 3D graphic effects.

"Graphic engines are another core piece of game architecture that can have relationship to almost any component" (Rilkey 2004, 11), it involves audio engine, physical model engine and other relative sub engines. The responsibility of audio engine is load and play sounds and music within mobile games. "A key contributor to the success of any game is the addition of sound effect. They can add an exciting depth to a positive gameplay experience, but also take away from that experience if not carefully planned"(Pardew & Pugh & Nunamaker 2004, 13).

Physical engine and Game AI engine

Game designs include extensive amount of art to help the game developers in creating the final games. There are some mainly art typically found in game designs, storyboards, environment illustrations, character designs and other art designs (Pardew & Pugh & Nunamaker 2004, 13-17). Environment illustrations, character designs and other graphic effects can achieve by graphic engine and audio engine with

some 3D effect technologies. However, the most of mobile games adapt to event models. Every mobile game covers different storyboards. “Storyboards are sketches that indicate how sequences of events are to take place” (Pardew & Pugh & Nunamaker 2004, 106).

The principle of event model is that the players or entity in mobile game make actions about the events, and the physical engine and game AI engine will react relevantly actions. To updating a new game state in mobile screen and waiting for next actions. And the actions can form a cycle follow the principle of event models above. Specifically, the actions can be cause by the input or controlling from players, or simulations of physical engine. Physical model engine is an independent sub engine in whole mobile game engine structure. According to the theory of game simulation from Rilkey Scan (Rilkey 2004, 36-37), I understood that the physical model engine mainly simulates an entity moving with stress situation and how entity acts in the real world. Physical model engine in a complex design, it involves some physical calculations, such as mechanics and kinematics. I give a detailed description and explain in physical model engine design chapter.

On the other sub engine, Game AI is a high level control language. It makes the computer-controlled opponents (or cooperative elements) appear to make smart decisions when the game has multiple choices for a give situation, result the behaviors that relevant, effective, and useful (Schwab 2004, 7). The game AI engine likes a logical engine in whole system. It handles all of simulations in mobile games, and cause intelligence results. AI engine can not only expects and changes the game states, but also avoids some self-contradictory situations in mobile games. During the process of my designing, the physical model engine and game AI engine will be considered comprehensively. Sequentially, to making their work together to keep the integrative of mobile game engine design.

5.1.1 Basic Layer Design

The basic layer of engine is a basic part. It is in a lower position in whole engine structure. The basic layer of engine includes all of the basic components and support functions what high-level engines needs. For the game developers, the basic layer of engine provides a unified developing platform and avoids some problems among different types of mobile device and configuration criteria. In other word, the basic layer of engine makes the whole engine system to satisfy the most of mobile games development. In my design of engine structure, I determine four contents in this layer, game graphic renderer, I/O management, game database and J2ME platform.

Game Graphic Renderer

One of goals of 3D Game is imaging reality. Therefore, the most of game developers try to improve the image quality and object precision in game scenes. In my design of mobile game engine, I recommend combining JSR 184 (M3G) based on J2ME to complete 3D rendering. “The Render function is called every time a change to the board is made. It is a relatively simple function that clears the screen, loads the identity matrix and then draws the background if necessary. ” (Root & Donnelly & Yeganov 2008)

The rendering technology is an important technique in 3D graphic engine. It plays a core supporting role in 3D graphic performance, physical performance and 3D effect system. The game graphic renderer involves many graphic processing technologies, such as light rendering, texture application, particle system and physical simulation. Therefore I design the game graphic render in basic engine layer, in order to provide forceful supports for completing high – level engines design, especially 3D effect system.

Input/Output Management

I/O management is a basic program in engine design. It mainly consists of three parts,

devices management, documents management and programming application management. Specifically, the device-manager supports the input from different types of mobile devices. The keyboard and joystick, as two kinds of commonly used input devices are managed and supported by device-manager. On the other aspect, the document – manager is responsible for reading and processing data and information about scene files, graphic files, audio files and other format files, and then assigning those data to corresponding sub engines. Besides, the programming application – manger is required to complete processing with device events. It can offer specifically case solutions for different events happen, such as call incoming, message incoming or audio controlling.

Game Database

The game database is a base component of mobile game engine. It usually involves other application in mobile engine. In my engine design, the game database is a solid supporter for I/O management. During the working process of I/O manager, the game database is responsible for saving, deleting, and updating the game information and data, such as user information, and device information. It also completes to transmit the data to corresponding components of mobile game engine.

According to Rilkey Scan's design about game architecture (Rilkey 2004, 7), I think the game database can be as a resource manager in my engine design. The resources manager is required to keep art assets and audio assets for support graphic engine and audio engine working.

On the other important aspect, in some particular sub engine architectures and implementation, the mathematic calculating is a necessary element in those engines, such as physical model engine and game AI engine. For high-level engine designs, they should be explained and described working theories and functions through mathematic calculating, for example, vector calculating, triangle function and other complex calculations. "The basic math functions will improve the support library

within the game engine" (Harbour 2009, 213). Therefore, I expect a basic math library in game database design. The functions and basic mathematic calculation in math library are manly provided for high-level engines in my engine design. Especially, in physical model engine, the engine should simulate some physical situations, such as collision detection in auto playing games, the balls moving and roles sporting in role playing games. In addition, the mobile games can be divided many classification as adventure class, action class, role-playing game, simulating management game, etc. The design of basic math library can base on API, but also development objects. As I considering, I will put the complex mathematic calculations in basic layer of engine, in order to simplify the whole engine design and expect a clear engine architecture.

J2ME Platform

J2ME play a significant role in whole engine design. The most of technologies and structures of game engines are established based on J2ME platform and technology. Firstly, J2ME support the JAVA development environment for every component and technology of game engine. To Base on JDK and WTK tools configuration set up some variables to satisfy JAVA configuration environment. During the developing of mobile game engine, I need install the development tool – IDE (Integrated Development Environment). IDE are comprehensive tools, which combine with all functions required by programming designing together. IDE includes mainly three software, JBuider, Netbeans, and Eclipse. Especially, in my engine design, I recommend Eclipse software as the basic developing tool.

J2ME technology is an integrative programming, which has a good scalability and open developing. Based on J2ME platform, I add the particularly optional packages to support my engine design. Specifically, I take configuration JSP184 (M3G) to complete 3D graphic handling. Absolutely, other kinds of API also could be added, such as Open GL, to concentrate by 3D programming open API feature.

5.1.2 Main Engines Structure Design

This chapter concentrates to specific sub engines design, including scene management engine design, physical model engine design and Game AI engine design. At the same time, the 3D effect graphic processing will be adapted and permeated in all engine designs.

5.1.2.1 Scene Management Engine Design

Scene management engine is a core sub engine in game engine layer. The Scene management engine is not only an engine responsible for establishing a scene in mobile game, but also manages coordinate of specific components' working in whole game engine.

When Comparing with scene management engine design with PC games, the mobiles' design is simpler. In my engine design, based on the J2ME technology, the M3G can provide two kinds of developing mode, immediate mode and retained mode. Specifically, if game developers use the immediate mode, they do not need to form a game scene tree structure, just lead the .m3g files, which includes camera, light mesh, etc, into mobile program and complete rendering by Graphics3D. On the other mode, retain mode, establishing a game scene should base on World Class in M3G. With the foundation of game scenes, the developer can manage functions of scene follow structure of game scene. Besides, the retain mode also needs Graphics3D to render game scene.

In the developing of PC scene management engine, the game programmers always spend more time and costs designing high quality rendering effects for images. However, during the process of my scene management engine design, duo to that I put the renderer design in basic layer, therefore I should concentrate that how to combine every components of engine and form an integrative engine working.

According to the Figure 7, the structure of infrastructure component of mobile game engine, the Figure 8 describes the position and content about Scene Management Engine in whole engine design,

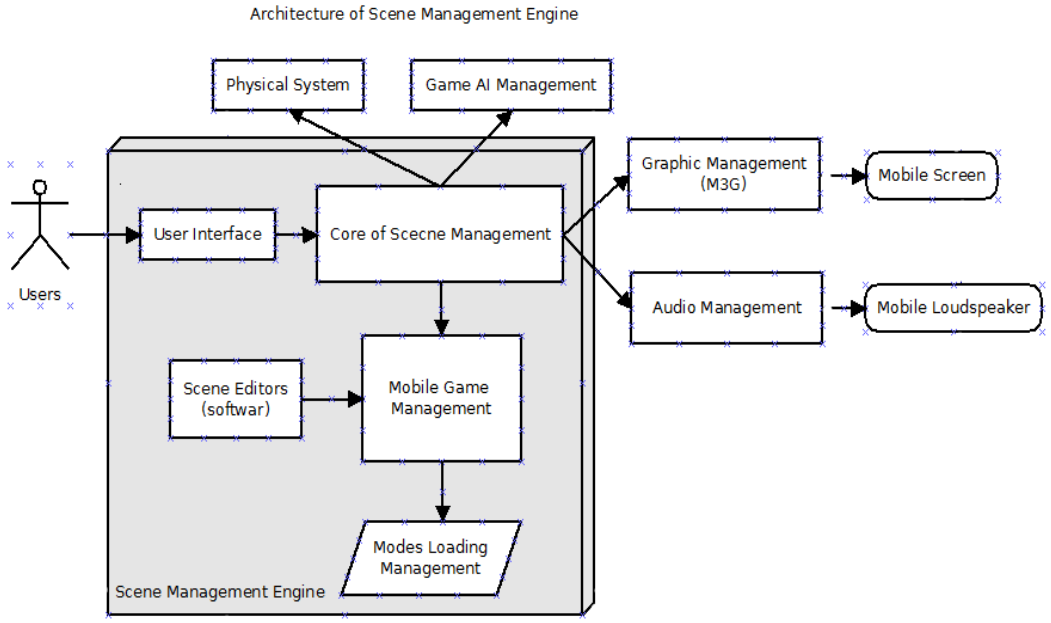


Figure 8. Architecture of Scene Management Engine

Figure 8 is emphasized the core position of scene management engine in game engine. In the Figure 8, the gray structure expresses the content of scene management engine, and other management model expresses the other parts in game engine design. The scene management engine can be a central sub engine, which combines with other components in engine structure.

There is a benefit that easy to extend engine extra functions within architecture of engine. Specifically, the extra function models are provided only API for scene management engine, the other comments and sub engines will extend corresponding functions.

MVC Mode Design

“Model View Controller (MVC) was a technique devised in the late 1970s by the SmallTalk community. It simplifies the development of user interfaces by cleanly separating code into three layers: the model, the view and the controller” (Walnes & Abrahamian & Cannon-Brookes 2004, 83). Due to those authors’ opinion gives me some ideas to design scene management engine. MVC is a classical method in UI structure. Although MVC can initially seem daunting, it can simplify some specific sub engine structure designs in my thesis work (Walnes & Abrahamian & Cannon-Brookes 2004, 83-86.) Specifically, I expect Model can encapsulate core data and information in system program, and keep the logical functions applying and independent I/O operations. Besides, View expresses user interfaces, and displays different types of views according to engine models. In addition, the Controller layer is important in MVC structure design, it responsible for processing and transmitting input information from users and event data in system, in order to ensure the corresponding relationship between user interface and engine models.

MVC has some benefits features to satisfy my engine design. I most likely will end up with many views, models, and actions. There must be a way to easily manage all these various parts in application in a standard way (Walnes & Abrahamian & Cannon-Brookes 2004, 83-86). I will design a particular MVC structure to define a criteria way, to found a scene management engine. The Model layer mainly includes KeysManager, Configuration, and M3G technology. As I expressed above, the components of Model layer of responsible to receive data and react to corresponding actions and functions. In View layer, duo to the M3G can provide the Graphic3D to render images. I determine the View express through M3G technology and other relative assistant tools. Besides, the Controller is a transfer tool. It includes three Classes, Camera, Player and EngineCanvas, to complete every component following input commands and transmit processing data to View layer.

Figure 9 illustrates the Architecture of MVC in Scene Management Engine design.

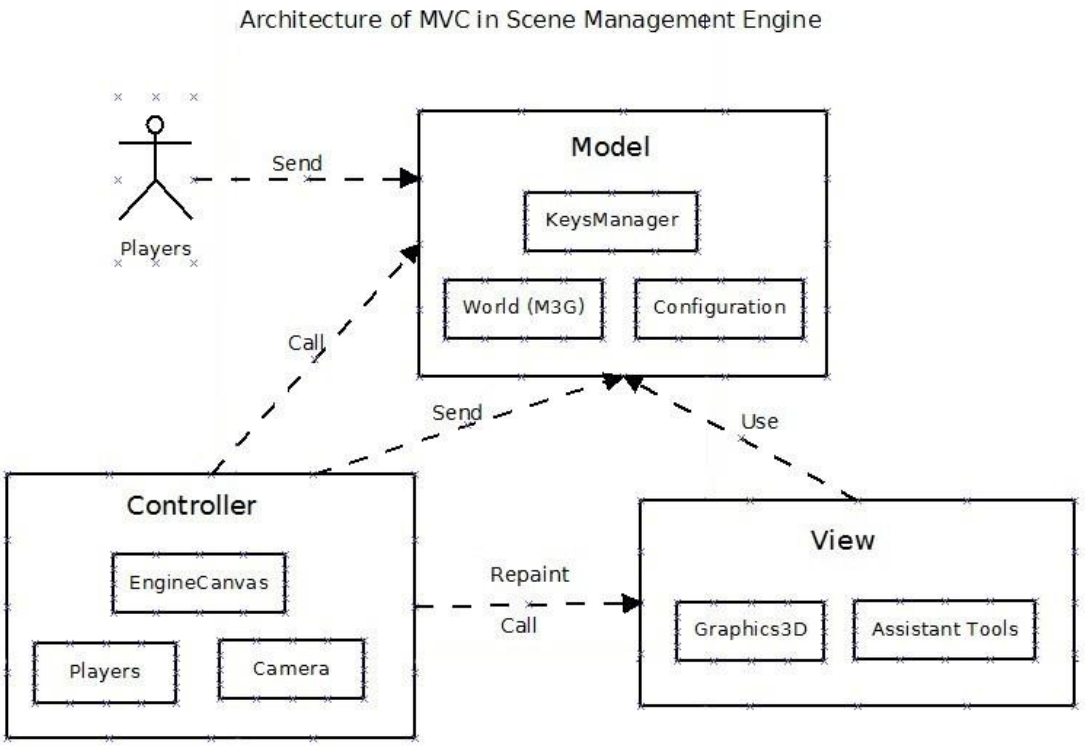


Figure 9. Architecture of MVC in Scene Management Engine

As shown in Figure 9, MVC determines specific function classes in scene management engine design, “because the MVC layers are decoupled, each becomes easier to refractor, break down into components, and reuse” (Walnes & Abrahamian & Cannon-Brookes 2004, 83-87). Therefore, every component to complete independently their corresponding functions. Each action class will make unit testing much easier (Walnes & Abrahamian & Cannon-Brookes 2004, 87), in implementation chapter, I detailed expound specific classes and actions among them.

5.1.2.2 Physical Model Engine Design

In order to make the mobile games more approximate with the real world, physical model system plays a very important role in mobile game engine design. The objects

in mobile game usually process a physical simulation. “A physical simulation is a representation of a situation that includes objects with characteristics that approximate the real world” (Rilkey 2004, 9).Generally, the core technique in physical model system can attribute to the physical model engine, it can make the object in mobile games follow a constant rules to move or to sport. For example, the flight path of one bullet, parabolic or free falls movement of an object under gravity environment.

“In the real world, objects have physical attributes such as position, velocity, acceleration, and orientation. As object in a physical simulation are an approximation of these real objects, they tend to have attributes that are similar to real world objects” (Rilkey 2004, 9-11). Some physical attributes will affect the physical simulation results. On the other aspect, some physical phenomenon will express with those attributes in mobile games, in order to enhance the reality of games. Therefore, “Physical simulations are an area of game development that involves a great deal of math and potentially, a large number of objects” (Rilkey 2004, 9-11), in physical engine design, it involves lots of physical calculations.

Collision Detection

Collision Detection function is a necessary component in physical engine design. “Collisions are one of the most common ways that objects interact, and are certainly one of the most visible portions of a game” (Rilkey 2004, 69-95), it is usually used to detect the collisions between different objects in virtual space. In physical simulation, the value of collision detection is to avoid the “through” or “cross” phenomenon in mobile scene. If not, the game effects will lack of reality, because two or more objects cannot occupy in a same space in real world.

During the designing of collision detection, I consider two types of collision detections, target object with stationary object in game scenes detection, and the other type of detection between target object and a moving object.

Generally, in the development of mobile game, the common method of collision detection is to adapt to optical principle. According to the example from Crooks' designs (Crooks 2005, 129-133), the principle of collision detection is that regarding the target object as a point, it can be defined the point a coordinate X, Y in a frame, and launch a beam of light starting from the point to direction of its movement, until the light collides some objects in mobile scene. When the light collide objects, the detection system will receive feedback from light test, and process relative calculations, such as how long distance or time the target object can move. The collision detection will compute collision happen according to result of the distance and time. Moreover, the "light" mentioned before is not actual meaning light, I expect to design the collision detection based on property of light launching followed straight line.

The light collision detection is very simple principle, but I think the key point is that how to set a reasonable point to reference measure. Specifically, the key point is a center point with an object generally. For regular objects, the center point can be found easily through the geometric methods. However, for irregular objects, it is hard to find a center point. Depend only on single collision detection for mobile cannot satisfy collision detection for irregular objects, I recommend extra collision detection methods from PC game in my design, which is going to be introduced as below design.

Here I provide an irregular objects example to express the result about improper center point selecting with Figure 10,

Figure 10. Car moving through a wall

Due to improper center point selection, the collision cannot be detected in a correct situation. With the example description, the car was moving through a wall. This kind of design will lose the reality of games and players' enjoys.

To solve the problems, I combine some collision detection methods with PC. According to the example and theory from Rilkey Scan, the Bounding Box and Bounding Spheres (Rilkey 2004, 69-95) are adapted into my design. Specifically, I will introduction these two kinds of collision detection method following Figure 11 and Figure 12.

Figure 11 shows both types of bounding boxes

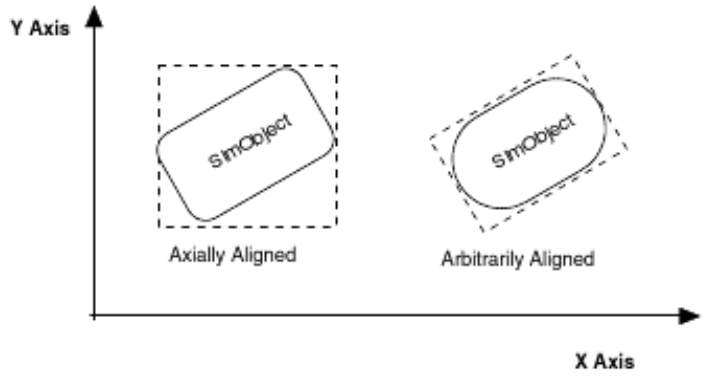


Figure 11. Bounding Boxes

"Bounding Boxes are a collision detection method in which a conceptual box is drawn around each object in the simulation, and collision checks are performed between the boxes rather than object themselves. It can be used in 2D or in 3D and provides a fairly good approximation of collisions, especially for simple objects" (Rilkey 2004, 79).

Figure 12 shows Bounding Spheres for some objects

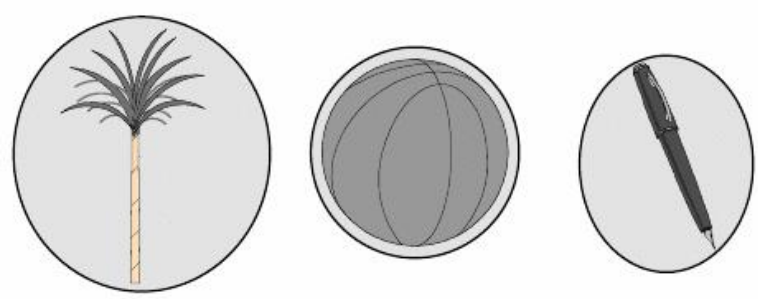


Figure 12. Bounding Spheres

"Bounding Spheres are a method of collision checking similar to bounding boxes. A sphere or circle is created around the objects in the simulation, and intersection tests

are performed between the spheres rather than the actual objects" (Rilkey 2004, 76).

Because of the recourses limited of mobile devices, the engine designs also have many limits. Comparing with PC game development, it concentrates how to achieve the best game effectiveness. However, for mobile game development, it should be considered how to flow a good game effective within the least mobile resources. Therefore, for designing of collision detection in my thesis work, I combined bounding boxes/ spheres methods based on light collision detection method. Because of "the two types of bounding boxes: axially aligned and arbitrarily aligned. Axially aligned bounding boxes (AABBs) have the property that their sides must be perpendicular to the principle axes. Arbitrarily aligned bounding boxes can be aligned is any direction with no regard for the principle axes" (Rilkey 2004, 79), therefore, the bounding boxes or spheres methods can find the center point for all over objects. Based on the "rule" methods, and following the light principle method, I think it can become an effective and simple collision detection method for my engine design.

Physical Model system

Collision detection is a core technique to support the designing of physical model engine in thesis. Generally, the physical model system is more complex, it has some multiply objects in a frame, not only one or two objects. While collision detection can be computationally expensive, especially when there are many objects involved. At the same time, it is a math-intensive area of game programming (Rilkey 2004, 9-11). A good collision detection and reaction is fundamental to a positive gaming experience. Collision detection is central to 3D games that simulate interactions of the physical world (Clingman & Kendall & Mesdaghi 2004, 464), through the collision detection, the physical engine modifies and calculates these physical behaviors, and makes the mobile games look like more reality with accurately physical computing and correct. Through combed with collision detection, I expect an integrative physical model engine design in implementation of my thesis work.

5.1.2.3 Game AI Engine Design

AI in Mobile Game

Artificial Intelligence (AI) is the creation of computer programs that emulate acting and thinking like a human, as well as acting and thinking rationally (Schwab 2004, 4), to apply AI technology in development of mobile game, is can not only increase immersion in mobile games, but also improve the process of game developing, even the design style of game design. “Game AI is the code in a game that make the computer-controlled opponents (or cooperative elements) appear to make smart decision when the game has multiple choices for a give situation, resulting in behaviors that are relevant, effective, and useful” (Schwab 2004, 4), while in design of mobile engine, the game AI engine is a high-level and significant design.

According to the AI theory from Lecky-Thompson Guy, “AI is limited to the connection between the player and the game. This intelligence is encapsulated in the way that the vehicle, object, or player-controlled character reacts to the control mechanism” (Lecky-Thompson 2008, 169). I can conclude a frame to express the AI working principle as Figure 13.

Figure 13 illustrates the working principle of AI system.

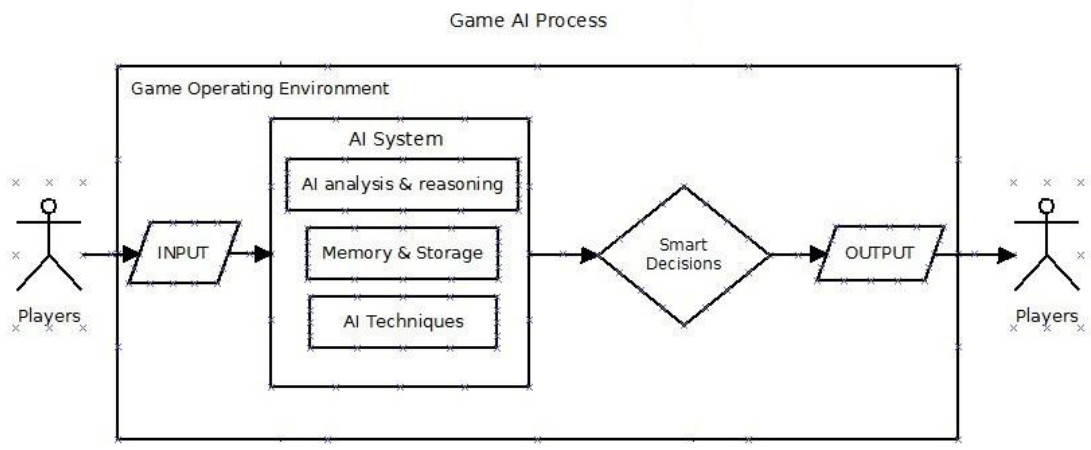


Figure 13. Working Process of AI system

In Figure 13, the input interface in game AI engine receive the input information and data from players to react into AI system, and to process the analysis and reasoning according for those data. The AI system is the core component in game AI engine. And then AI system computes a reasonable decision based on data analysis and reasoning. Specifically, the reasonable decisions can be single or more, in multiple descriptions will be listed with a reasonable order. The memory and storage layer are particular in allusion to those information and data from players' controlling. They will be expressed and stored with a proper way in mobile. Finally, the decisions or behaviors are expressed through different coordinating objects' actions or behaviors in mobile games. Therefore, the players will enjoy the Artificial Intelligence actions in mobile games.

Game AI engine

"AI can help adapt the game to the player, beat the player, let the player win, and model the environment so that everything hangs together properly, making the player feel at one with the game – while also providing a way for the opponents to behave in an intelligent way" (Lecky-Thompson 2008, 251), but also the AI can process multiple situations and make smart decisions for other mobile game classifications.

According to the part II Game Genres from Schwab Brian' Design, (Schwab 2004) different genres of game needs different AI components, techniques, supports and environment, such as RPGs, RTS games, and FTPS games. Although different game AI system has different property and operation mechanisms, they involve a basic AI engine techniques and principle. On the other reason, the mobile devices have some limits with mobile recourses, with increasing the playable of mobile games through AI techniques, the limitations of mobile resources should be considered at the same time. In my thesis work, I expect to complete the game state prediction and change-operation through two AI techniques for mobile games as follow,

Finite-State Machine (FSM)

“In the world of game AI programming, no single data structure has been used more than the FSM (unless you count the switch statesmen as data structure, perhaps, but a switch can actually used a simple form of stat machine). This simple yet powerful organization tool helps the programmer to break the initial problems into more manageable sub problems and allows programmers to implement intelligence system with flexibility and scalability” (Schwab 2004, 241-242) .

Figure 14 shows a classic FSM frame.

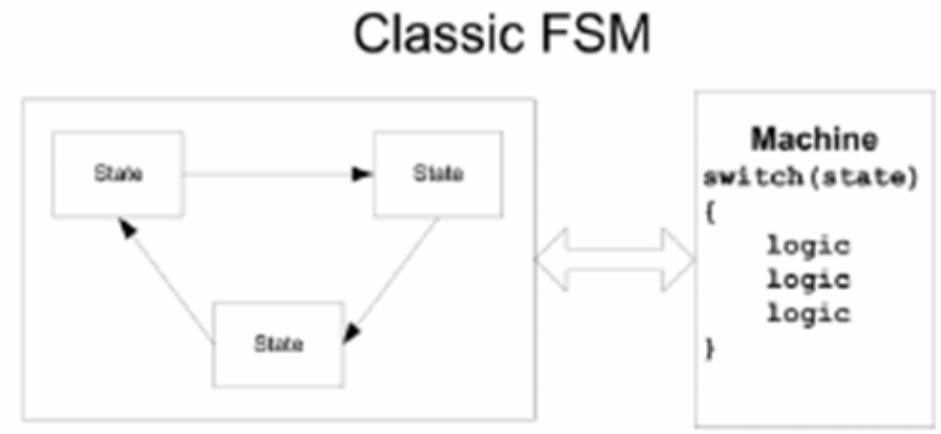


Figure 14. A Classic FSM

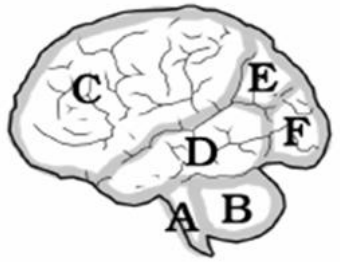
The FSM consists of switches among state and state in mobile game. Specifically, the FSM only keeps someone state. It always is used to manage the whole game scene and operate single object in game scene.

Scripting System

“A common technique for getting more hands into the guts of the AI system, without having to hire more programmers, is called scripting. Scripting means using a simplified programming language (although scripting tools can be mad visual as well as written) to build AI element, logic, or behaviors” (Schwab 2004, 337). The scripting system can drive game events, model role play AI behaviors and automatic tasks in mobile engines.

Game AI in Mobile Game Engine Layer

Game AI is a high-level control language in mobile programming, it can adapt high-level scripting to model the program based on AI techniques. However, the game AI engine is not an independent engine, it combines other sub engines to form an integrative mobile game engine. The senior place of game AI engine is that it likes a human brain to dominate the "thinking" of every component in mobile game engine. "The organizational model of the brain has merit when setting up an AI engine, as seen in Figure 15, which shows the relative tasking between brain and game systems" (Schwab 2004, 11-12).



Organization of:		
	The Brain	A Game AI System
A	Brain Stem - Reflex - Lower Functions survival	- Collision - Animation Selection
B	Cerebellum - Sensory Mixing and Coordination	- Physics - Navigation
C	Frontal Lobe - Higher Brain Functions - Emotions - Learning	- Decision Making
D	Temporal Lobe - Memory (visual and verbal)	- Learning
E	Parietal Lobe - Sensory Center	- Perceptions
F	Occipital Lobe - Visual Processing	- Perception

Figure 15. Object-oriented nature of the brain relative to game AI system

5.1.2.4 3D Effective System Design

There is an excellent mobile game in issuance into market. It is the more favorite with players. The colorful graphics images and funny game scenarios attract to a large number of players, while those kinds of visual effects were completed by various 3D effect technologies in mobile game engines.

3D effect system is not independent. In the whole mobile game engine design, it achieves a great visual effect through coactions with other sub engines. In my design of thesis work, I expect a comprehensive 3D effect system to complete effects processing of 3D mobile game images. There are many 3D effects design expressed through renderers. As my considering, I design the game graphic renderer into basic layer design before, which combined M3G as general API criteria.

Camera

The two important considerations when writing a renderer for the first time are camera and light sources, which are the premise conditions for rendering. “The camera may be positioned anywhere in 3D space, as well as pointed in any direction in 3D space.” (Harbour 2009, 48). In the mobile games, the Camera will be as a class to design within engine. It can be player’s eyes, and plays a significant role in feeling of 3D world for players. And then I illustrate the working principle as Figure 16.

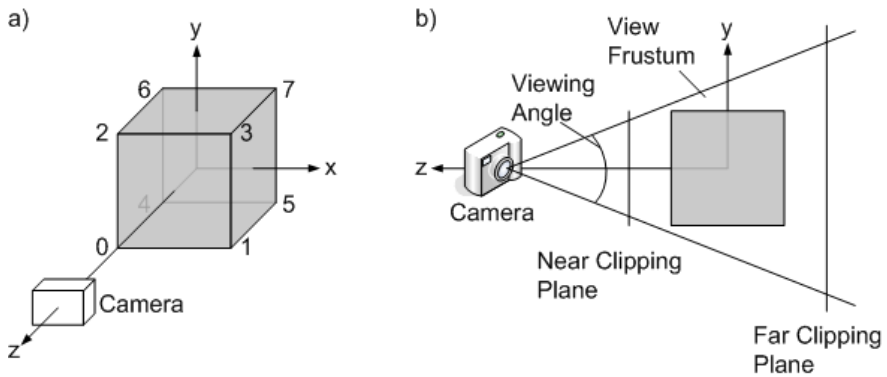


Figure16. Camera Sketch Map

For part a, there is assume about the cube in a mobile game scene, which faces the negative direction of Z axis, but positive of game scene. The position and property of Camera defines the images which display in mobile scene. For part b, the Camera can change the view angle and near or far clipping plane of camera cause by different mobile imaging.

Animation Rendering Technology

The animation rendering is one of common techniques in mobile particular effects. Due to the M3G can provide extra software and tools API, there are so many developers usually cite directly some animation renderings which have been designed by other designers, in game developing process. According to the inspiring of those developers, I cite the animation rendering technologies in my engine design. Specifically, I will follow the Kennedy Sanford's description, "the 3Ds Max can feature by exploring the capabilities of the skeletal deformation animation system" (Kennedy 2004, 6). Considered the limits of mobile device, I think this kind of animation rendering technology is enough. And then the graphic engine will be required to complete loading and transmitting with .obj modeling data from 3Dx Max, through API provided M3G.

6 3D MOBILE GAME ENGINE IMPLEMENTATION

Through the previous descriptions and introductions of mobile game layer structures, it can be understood that the mobile game engine consists of many components and sub engines. At the same time, these elements need completive and orderly operation together. Therefore, the mobile game engine design will achieve its values.

This chapter concentrates on implementation of parts of engine functions to support design theories. Specifically, I expect the implementation of scene management engine as a center to extend a basic mobile game engine development.

6.1 JAVA Installing and Environment

J2ME is a basic development tool in my thesis. Therefore I will establish a J2ME platform to support the implementation of mobile game engine development. Because in chapter 4.1.3 has detailed introduced J2ME and relative its development environment, I simply list installation steps below,

1. Download and install the JDK (Java Development Kit)
2. Configuration environment variables
3. Download and install the WTK (Wireless ToolKiy)
4. Install the development tool – IDE (Integrated Developing Environment)
5. Download and install the Eclipse Software Application

The Eclipse is the mainly development tool in my thesis work. Firstly, it should keep operating the JAVA development environment, the Figure 17 shows the checking of the JAVA development as follow,

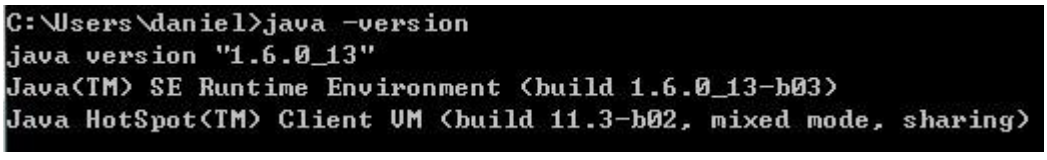


Figure 17. Java Version

Typing the “java -version” in cmd commander, the computer system shows the Java version of JDK. That means the Eclipse software can be operated. Especially I choose 3.2 version of Eclipse software in my mobile game engine development. And then, I set up the J2ME platform.

Adding M3G API

Considering the development of my thesis work is 3D mobile game engine, therefore, I should add the M3D API to development environment for supporting 3D effects and technologies.

Firstly, the Eclipse defined the JSR 184 as a Mobile 3D Graphic API for J2ME, therefore, I add the JSR 184 development packet in Eclipse environment.

Figure 18 shows the adding JSP 184 in Eclipse Libraries.

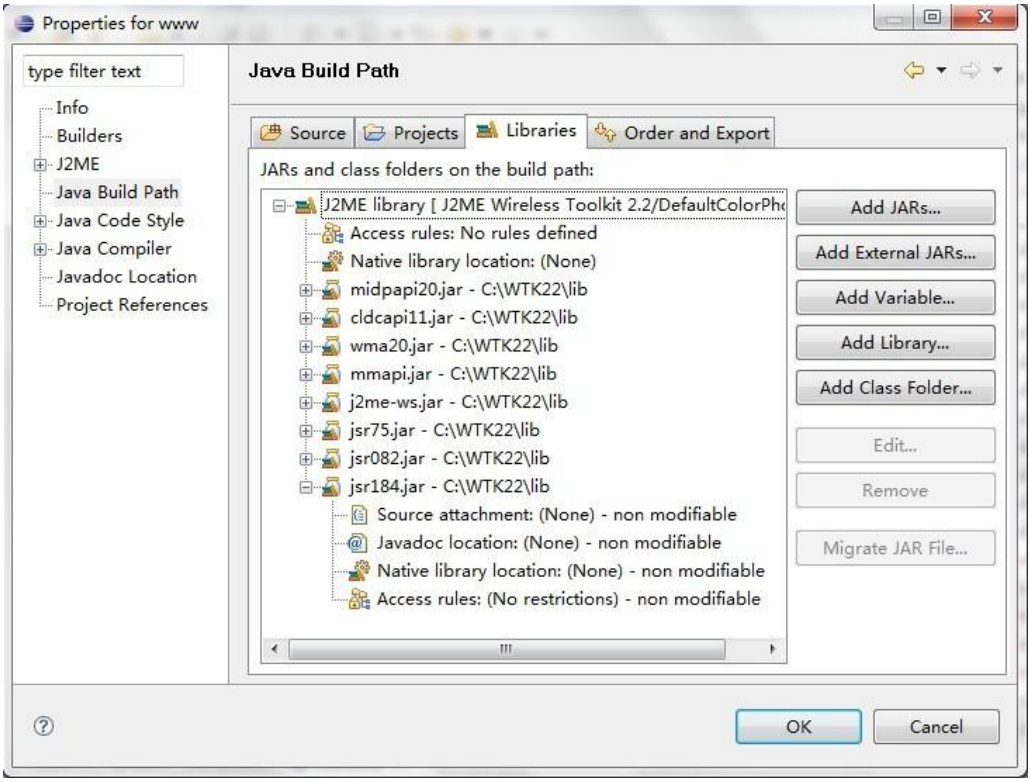


Figure 18. JSR 184 in Eclipse Libraries

According to the description of M3G in chapter 4.3.1, the JSP 184 supports importing and loading the .m3g file into J2ME platform. That is an important technology in the implementation of scene management engine later.

6.2 A Center Implementation – Scene Management Engine

Mobile game engine is a complex system, which consists by some complex components and sub engines. According to the analyzing in design chapter, 5.1.2.1 Scene Management Engine Design, I think the scene management engine is the most representative engine implementation.

Scene management is a core sub engine in game engine layer. The Scene management engine is not only an engine responsible to establish a scene in mobile game, but also having duty to coordinate of specific components' working in whole game engine. According to the Figure 8, scene management engine involves other sub engines interactions and reactions. Specifically, I focus on the implement scene management engine and the relative functions and the sub engines interactions which depend on design in my thesis.

6.2.1 Scene Management Engine with MVC

Model in Scene Management Engine

The model of scene management is mainly responsible for the interaction with the mobile games. During the development of mobile game engine, with the design of MVC architecture, the Model contents of interaction in games includes input from players, configurations in game engines, and some game scene models loadings. In the modes loading management part, which includes three different aspects, one is coming from processes of Word in M3G. Besides, there are KeysManager and

Configuration.

KeysManager Class mainly process the key buttons commands by players, it will react the action methods in KeysManager Class to execute the commands. Here, I illustrate the main function code and methods through an example.

Figure 19 shows the game example, PogorooMIDlet.

Figure 19. An example mobile game, Pogo

According to analysis the design in PogorooMIDlet, I concluded the main functions code and methods as Figure 20,

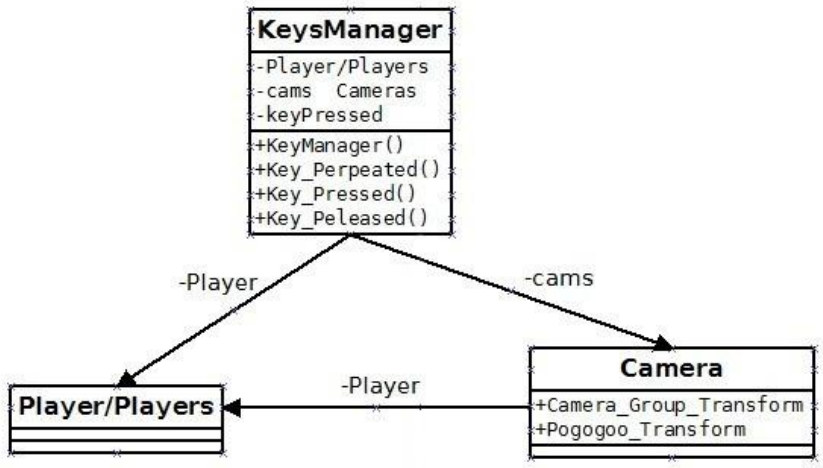


Figure 20. The structure of KeyManager and Interactions

In the design of KeyManager Class, the definitions methods include, testing the key buttons, gathering the change information about key buttons, and updating the

positions about roles or cameras in mobile game. The Figure 20 also expressed the interactions among other Classes, such as Player Class and Cameras Class, and execute or adjust depending on coordinated commands.

Specifically, for example PogorooMIDlet, the designer defined the key event type IDs as follow,

```
// Key event type IDs
public static final int KEY_REPEATED = 0;
public static final int KEY_PRESSED = 1;
public static final int KEY_RELEASED = 2;
```

Figure 21. Define the Key event type IDs

Configuration Class is mainly responsible for loading relative Configuration files and executing some game sets to different sub engines. I will place configuration as a set management system, and some specific applications will express in scene management engine implementation.

Depending on the basis of the example PogorooMIDlet, it also has some relative configurations used in the example, such as keyForward, keyBackward, keyLeft, keyDown and lookup tables.

Figure 22 shows the details in example PogorooMIDlet.

```
// lookup table for roo hops
private float[] hopSteps = {0.0f,

int viewport_x;
int viewport_y;
int viewport_width;
int viewport_height;
```

Figure 22. Define the Lookup Table in PogorooMIDlet

On the other experience, I completed a Mobile Game Design in Competence development project of KTUAS. Although the mobile game is only a simple game, it also should call some configurations. Generally, with development mobile games, the design always should call lots of general configurations, such as putInKey(), toEngine(), getInt(), gerFloat(), getCameraPosition() and getDefault().

Figure 23 show parts of relative Configuration function.

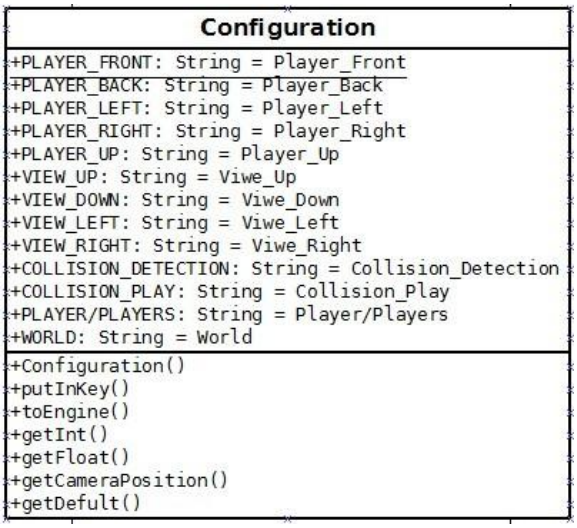


Figure 23. Parts of Relative Configuration Functions

Controller in Scene Management Engine

The implementation of scene management engine involves games objects and scene management. In scene management, the core Class is defined by EngineCanvas. The EngineCanvas also has interaction among with other Classes.

Figure 24 shows the interaction about EngineCanvas.

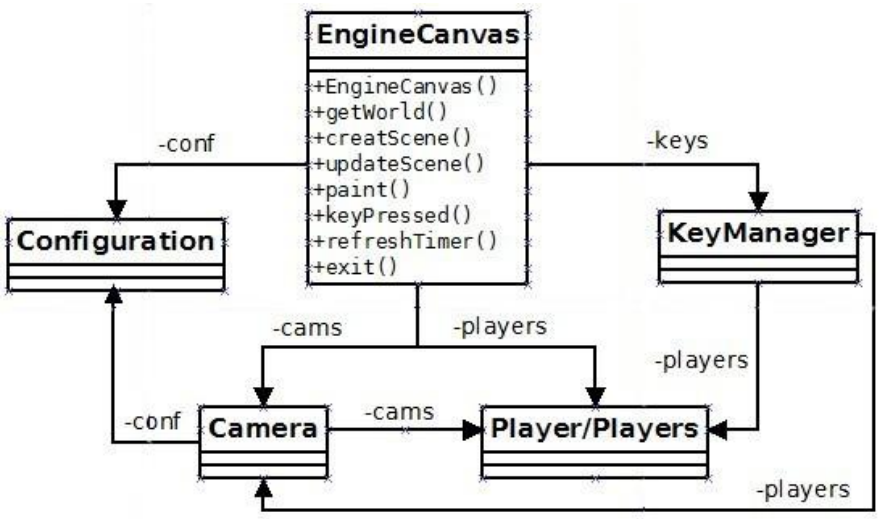


Figure 24. EngineCanvas Interactions

The EngineCanvas manages lifecycle of object, rendering with M3G API, calling the coordinated configurations, founding Camera and other functions in game engine. refeshTimer()can define a game life clock, it will be called in game time orders by developer setting. exit() can stop for task executing. When the game is over, updataScene() updates the game scene. keyPressed() and keyReleased() is called when KeyManager Class send key button commands, that means when players pressed buttons.

With the management of game scene and objects in game scene, it can divide static and dynamic object management. For the static object management, I expect directly use the World in M3G to support loading and manage. However, for dynamic object in game scene, its achievement should depend on Camera Class and manage separately Camera and other interactive Classes. According to Figure 24, the Camera Class has interaction between Configuration and Player Class. The Player Class has close relationship. Player Class can save the position, angles, size and other data for Camera, through calling configuration file from Configuration Class, to create or

adjust Camera Class, but also call for other function Class, such as Collision Detection or updateScene.

Camera in mobile game engine

During the development of mobile game, the Camera Class plays an important role in mobile game engine. The Camera can be players' eyes to bring them into mobile game world. The Camera Class is also a part of implementation of game engine. Comparing with different mobile game designs, I concluded that a Camera Class embedded scene management engine implementation.

Figure 25 expresses the Camera Class in engine.

```
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.game.GameCanvas;
import javax.microedition.m3g.Camera;
import javax.microedition.m3g.World;
public class camera extends MIDlet {

    public camera() {
        class Camera
        {
        public Camera(World w, int fov) {
        }
        public World getWorld;
        public boolean inView;
        public void captureScene;
        public int getModelDistance(Model m) {
            return 0;
        }
        public static hat get_xScreen;
        public boolean move_Camera() {
            return false;
        }
        public void camRotateMesh() {
        }
        public void AutoLookAtHowe() {
```

Figure 25. Camera Class in mobile game engine

camera(): Camera constructor

getWorld: return the world (game scene) through camera

inView: ensure the object in the camera horizon is visible

captureScene: change the coordinates of visible scene

getModelDistance: return the distance between camera and object in game scene

get_xScreen: a functional function. It is a calculation that computing the crosspoint between screen and the light from object position currently, in x axis.

Move_Camera(): when the position of camera change, move the camera horizon against object

camRotateMesh(): set a focus for camera

AutoLookAtHowe(): a functional function. Camera enables to veer automatically.

View in Scene Management Engine

Along with developing of scene management engine, the View of game scene model is completed by Graphics3D Class supported by M3G. In game engine design, the reason why I chosse to adapt to Graghic3D technology in View is that it can not only simplify the processes of development, but also support advanced functions to satisfy the commands of rendering in engine. Specifically, the Graphics3D is an interface for screen rendering. Based on J2ME platform the Graphics3D technology should achieve through M3G.

Figure 26 illustrates the Graphics3D application in J2ME platform.

```
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.game.GameCanvas;
import javax.microedition.m3g.Camera;
import javax.microedition.m3g.Graphics3D;
 public void paint(Graphics g)
 {
     try {
         g3d.bindTarget(g);
         // update the scene
         // render the scene
     }
     finally
     {
         g3d.releaseTarget();
     }
 }
```

Figure 26. Graphics3D in J2ME platform

Depending on Figure 26, I added Graphics3D through M3G API. Especially, there is a simple application about graphics3D to paint above. Generally, the Graphics3D can support several rendering classifications, but the most common classification is void (World). The game scene very often was established by World Classes in M3G, when the scene tree was completed, the Graphics3D rendered to game scene from World Class. Because when the Graphics3D is rendering, it needs to gain a unique render object, even unique game scene. In addition, I also recommended other simple method to finish mobile game engine rendering. The specific methods and functions will be mentioned after thesis work.

6.2.2 Game Interface

During the primary process of developing mobile game engine, through the analysis the game system, I defined a simple operation process. The game interface is a necessary component in scene management engine depending on Figure 8. Therefore, I will implement some different interface code based on J2ME as follow,
Figure 27 illustrates systematic procedure in mobile game engine.

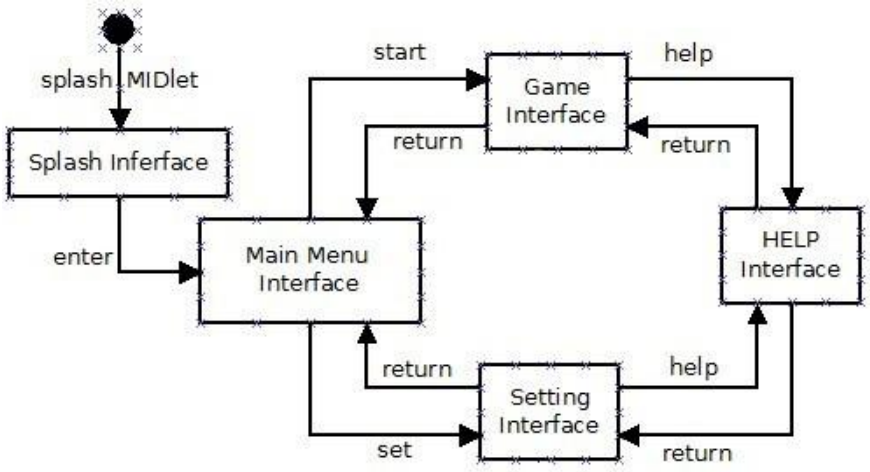


Figure 27. System Procedure about Game Interface

In the game interface, it includes five parts of game screens, splash screen, main screen, set screen, help screen and game screen.

Splash Screen is used in game starting loading, is usually show some game themes or developers' trademarks.

Figure 28 shows the cording of Splash Screen.

```
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.Image;
 public class Splash extends Alert{
     public Image Imgwelcome;
     public Splash(String s) {
     }
     }
```

Figure 28. SplashScreen Coding

Because SplashScreen is loaded at beginning of mobile game, I think it satisfies Alert.

Figure 29 shows the cording of MainScreen

```
import javax.microedition.lcdui.Displayable;
  class MainScreen{
      private static Displayable instance;
      public Displayable getInstance() {
          return null;
      }
      private MainScreen(){
      append();
      }
  }
```

Figure 29. MainScreen Coding

getInstance has better only one in mobile game cording, it can not only keep its state, but also save the mobile memory.

Figure 30 shows the cording of HelpScreen and SetScreen.

```
import javax.microedition.lcdui.Form;

class HelpScreen extends Form{
    public HelpScreen(String c) {
    }
}
class SetScreen{
    private static Displayable instance;
    public static Displayable getInstance0;
    private SetScreen() {
    }
}
```

Figure 30. HelpScreen and SetScreen Cording

Generally, the HELP function is that introduces game rules, score standards and how to control. Therefore, the HelpScreen just need to show text function, I expect Form to achieve it. On the other interface, SET function can offer personal options for game setting, such as audios and hard levels. This kind of Class relates close with Configuration system, different options with different players should save and call coordinating configurations to satisfy the mobile game commands.

Figure 31 shows the coding of GameScreen.

```
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Graphics;

class GameScreen{
    private GameScreen() {
    }
    public void commandAction(Command c,Displayable s) {
    }
    private void worldInit() {
    }
    private void draw_3D(Graphics g) {
    }
    private void setLights() {
    }
    private void draw_Feature() {
    }
}
```

Figure 31. GameScreen Coding

This kind of coding supports an engine to load a 3D game scene finished by other 3D modeling software, specific functions as below,

worldInit(): allow to load a 3D scene and adjust it

draw_3D(Graphics g): add a 3D object or entity in game scene

setLights() and draw_Feature(): add the light effect and other kinds of effects

6.2.3 Physical Model System

"Having a robust and efficient physical simulation is a very important part of any game" (Rilkey 2004, 9-11), through the collision detection, the physical engine will modify and calculate these physical behaviors, and make the mobile game look like more real with accurately physical computing. The modifying and calculating aims at two physical reactions, interactive two or more objects reaction and self object reaction.

Due to the physical model engine involves a large of mathematic and physical calculation. I am not going to implement the computing details and methods. However, I implement the processes of collision detection and physical modeling.

According to the Collision Detection Design in chapter 5.1.2.2, I focus on implement processes of collision detection in a simple situation, and the Bounding Box or Bounding Spheres detection method is in the top of the method lists.

Situation: there is one or less Bounding Box or Bounding Spheres taking place collisions.

This method aims only one object with collision detection in the game scene, and then distinguishes the shape of object is keeping regular or irregular.

Figure 32 illustrates a simple situation collision detection process.

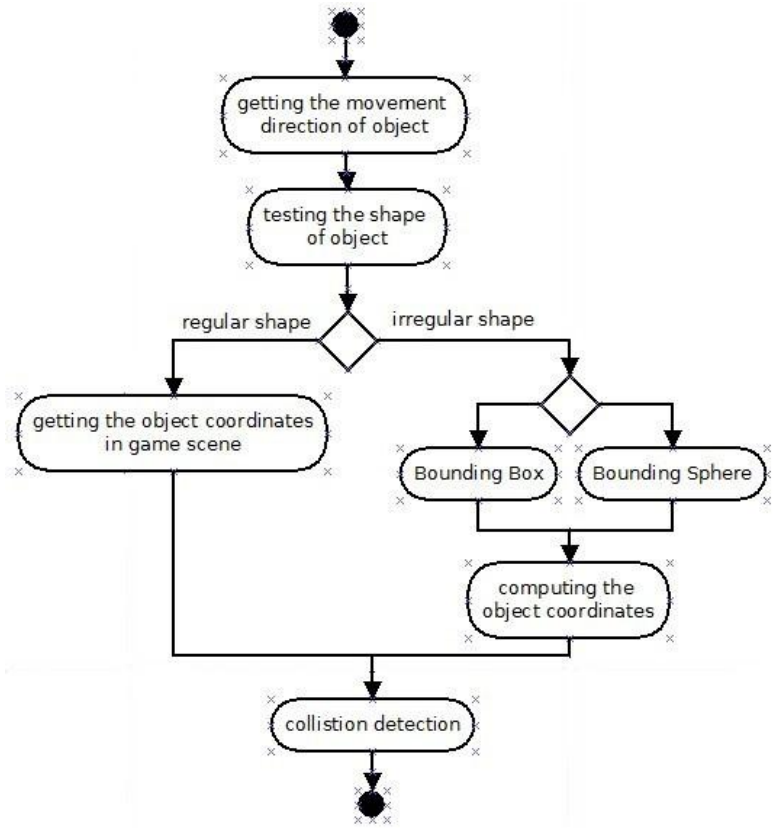


Figure 32. Collision Detection Process

The model should get movement direction of an object in game scene, and then the model will test the shape of the object, is it regular or irregular. For the regular object, it can get specific coordinate from game scene and then combine with the light detection to process physical collision detection. Besides, for irregular object, it should be means of Bounding Box or Bounding Sphere method to compute the coordinate, (x, y, z), in game scene space, and then process collision detection with common light detection method.

While I had a reasonable collision detection technology as core supporting, I implement a simple physical model. Different types of mobile games can have different physical model style to support whole engine. In order to establish a usable

physical mobile engine, I give a specific example, auto type of mobile game, to explain detailed.

During the auto type of mobile game developing, for physical model I considered two kind of physical simulations, reaction after collision and velocity changes, that means interactive two or more objects reaction and self object reaction mentioned before.

Figure 33 illustrates the executing process of physical model.

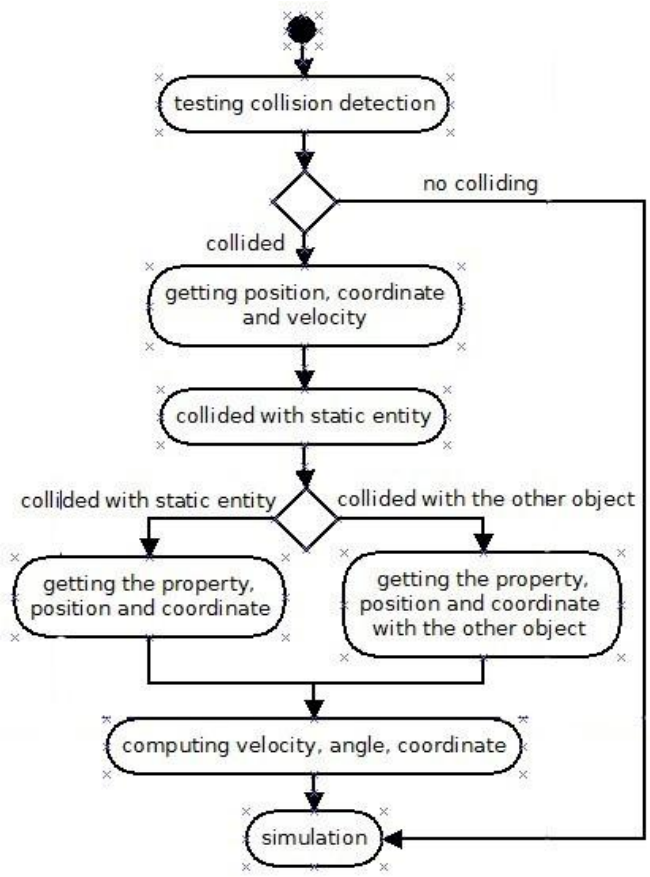


Figure 33. Executing Process of Physical Model

According to Figure 33, I set an auto type of mobile game as example to implement the simulating process in physical model. During the gaming, the mobile system will test the collision detections when the scene management engine receiving commands from players. Through analysis from the game AI engine, assign the intelligent

decisions and actions to coordinating sub engines. Specifically, the physical model engine will react relative computing, such as velocity, accelerated velocity, colliding angles and coordinate, and simulating. “Physical simulations are a core component of the game architecture, and interface with game simulations, graphics engines, and sound system” (Rilkey 2004, 9-11). Therefore, reflecting is a nearly real action in mobile game through comprehensive mobile engine system.

6.2.4 3D Effects Rendering

In the development of scene management engine, there are advances in some 3D effects rendering to complete processing of 3D mobile game images. However, the limits of mobile devices and compared to PC devices, the 3D effects in mobile cannot be achieved into a perfect state, such as application about models, animations, lights, shadows and other particular 3D effects applied by mobile devices limitations. Therefore, by combining with the component, Game Graphic Renderer in the basic layer, I implement parts of simple functions cording as illustrated below in Figure 34,

Figure 34 illustrates the coding for rendering.

```
abstract class Renderer{
private iut[]transformation;
public abstract void drawLine(int x1,int y1,int x2,int y2);
public abstract void clearSpace(int x,int y,int z,int l,int w,int h);
public abstract void setDrawRegion(int x,int y,int w,int h);
public void drawCuboid(int x,int y,int z,int l,int w,int h) {
}
public void drawScene(ViewPort view) {
}
```

Figure 34. Rendering Coding

drawLine: draw a line, defining two point of a line. It is a basic element in 3D space rendering.

clearSpace: clear a specific space, defining coordinate, (x, y ,z) and length, weight,

height, (l, w, h). It is also can be a 2D space.

setDrawRegion: define a specific region in mobile screen, coordinate (x, y) or (x, y, z) is the left vertex of region or space.

drawCuboid: define a cuboid, it is a common skill in 3D effects rendering. Developers usually put a camera in the cuboid or cube to form other effect, such as flowing.

drawScene: add the light effect to some vertex. Light effects are very important in rendering. It can form different visual effects depending on different light applications.

There are a lot of rendering technologies which can be rendered in the mobile game development, such as lights, colors, shadows and textures. Specifically, I give an example to explain the flowing effect as a representative effect in 3D game scene.

Figure 35 shows the flowing effect in game scene.

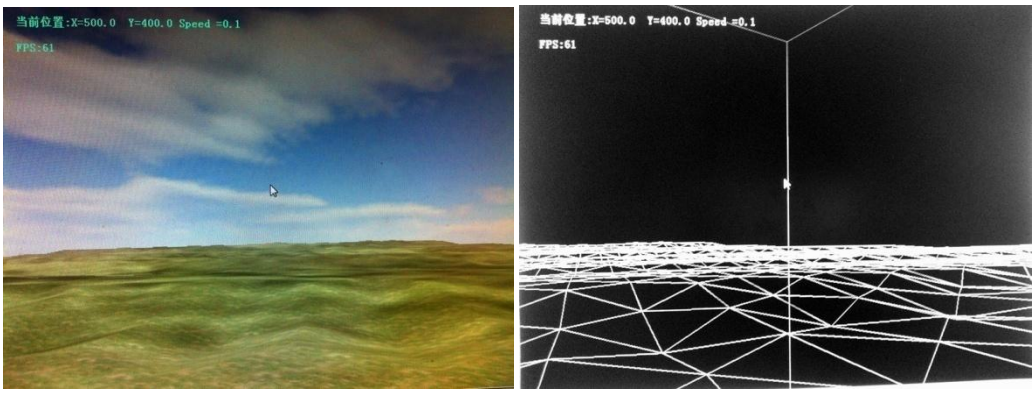


Figure 35. Flowing Effect – Sky and Terrain

This example of game scene, sky and terrain, is supported by OpenGL ES. The developers put the camera in a cube, and turn the cube spinning slowly with setting coordinating factors. When the camera executes the moving commands, it will move as a unit with the cube. As horizon of camera, the clouds achieve the flowing effect.

7 DISCUSSION AND CONCLUSIONS

The final chapter consists of the results of this thesis, discussion of limitations and challenges of this research, together with suggestions for further development and conclusion of my thesis work.

In this thesis, the objective is to design the framework of 3D mobile game engine. The specific engine framework achieved practical values. I proceeded from designing the main architecture of 3D mobile game engine, and processed detailed designs and analyses for the core structures and engine layers. By analyzing the existing 3D mobile game examples, I put forward some examples of implementations of specific engine functions supported by coordinating design frameworks and theories.

Specifically, the architecture of the engine design adapted hierarchy structures depending on different functions and model objects. For the mobile game layers structure, I describe detailed designs and analysis within components. The scene management engine as a central design involves game interface, camera, I/O manager and scene manager. I illustrated further design structures with MVC model and technique. Sequentially, map to other relative sub engine designs and functions, such as collision detection, physical model and 3D effects rendering.

Based on J2ME platform, I recommend some particular technologies and methods on PC, which include collision detection methods, game AI and 3D rendering technologies, to combine 3D principles and mobile rendering technologies into mobile game engine design. By considering that mobile game engines have miniaturization and multi-module characteristics, and those particular technologies and methods from PC were adjusted to meet configuration requirements and application standards. Therefore, I designed a mini type and simple 3D mobile game engine framework.

The mobile game engine design is a time-consuming and complex project. Therefore, this kind of time-consuming project can apply to different types of game development. Each game developer does not need to process engine project again and again. Therefore the engine development is worth once for all. However, the complex structures, confused logical processes, and a large of mathematical and physical calculations are big challenges for game engine developers. Completing an integrative mobile game engine takes a lot of time, knowledge of professional technologies and experiences, and strong device supports.

My thesis work is a basic research, not involving the use of enhanced design models, such as the application game AI engine. In a comprehensive mobile game engine, except for the flowery 3D graphic effects, the “brain” of the engine is also important. Game AI can make a lot of intelligent processes, and enhance playability of the mobile games. It is anticipated that soon designing games with fancy artificial intelligence will be game developers’ higher pursuits. However, my thesis has some distance to implement the seriously game AI functions.

In addition, my thesis can be perceived as a blueprint. It needs further research and practices to realize a mobile game engine. Even so, I increased understanding about mobile game engine design through the thesis work. In the java programming and mobile games domain, I broadened my professional horizon. Undoubtedly, the research was a big challenge for me, through continuous researching and studying, I sketched my own knowledge, skills, and ideas with logical structures. At the same time, thesis work is also a good opportunity for me to accumulate my professional knowledge and experience.

Due to the limitations and challenges during the mobile game engine development process, such as a more complex project, large of workload and limits conditions under the mobile devices. Therefore, many technologies and problems need more reasonable applying and processing, and more knowledge and technologies will be

understood in practical developing process. Implementing an integrative mobile game engine, it needs further researching and studying in the future.

REFERENCES

Printed

Astle, Dave & Durnil, Dave 2004. OpenGL ES Game Development. Course, Technolgy. Boston, MA, USA 2004.

Clingman, Dustin & Kendall, Shawn & Mesdaghi, Syrus 2004. Practical Java Game Programming. Charles River Media. Hingham, MA, USA 2004.

Crooks, Clayton, II 2005. Mobile Device Game Development. Charles River Media. Hingham, MA, US 2004.

Hair, Joseph F. Jr. & Money, Arthur H. & Samouel, Phillip & Page, Mike 2007 Research Methods for Business. Chichester, West Sussex, England; Hoboken, N.J. John Wiley & Sons, 2007.

Harbour, Jonathan S 2009. Advanced 2D Game Development. Course Technolgy. Boston, MA, USA 2008.

Järvinen, Pertti 2001. On Research Methods. Opinpajan Kirja, Tampere, Finland.

Kennedy, Sanford 2004. 3Ds Max 6 Animation and Visual Effects Techniques.Charles River Media. Herndon, VA, USA 2004.

Lecky-Thompson, Guy W. 2008. Video Game Design Revealed. Course Technolgy. Boston, MA, USA 2007.

Lukka. K. 2003 . The constructive research approach. In L. Ojala & O-P., Hilmola. (Eds.) Case study research in logistics. Publications of the Turku School of Economics and Business Administration, Series B 1: 2003. 83-101.

McCuskey, Mason 2002. Special Effects Game Programming with DirectX. Course Technolgy. Boston, MA, USA 2001.

Pardew, Les & Pugh, Scott & Nunamaker, Eric 2004. Game Design for Teens. Course Technolgy. Boston, MA, USA 2004.

Rilkey, Scan 2004. Game Programming with Python. Charles River Media. Herndon, VA, USA 2003.

Schwab, Brian 2004. AI Game Engine Programming. Charles River Media. Hingham, MA, US 2004.

Walnes, Joseph & Abrahamian, Ara & Cannon-Brookes, Mike 2004. Java Open Source Programming : With XDoclet, JUnit, WebWork, Hibernate. Wiley. Hoboken, NJ, USA 2004.

William H. Michael,Jr & Vernon H, Miles & Amy, Janik & Courtland s, Lewis 1989. Engineering Infrastructure Diagramming and Modeling. National Academies Press. Washington, DC, USA 1986.

Zerbst, Stefan & Duevel, Oliver 2004. 3D Game Engine Programming. Course Technology Crisp. Boston, MA, USA 2004.

Not Printed

Eberly, David H 2007. 3D game engine design: a practical approach to real-time computer graphics. Downloaded 18 March, 2011. <http://books.google.com/books?hl=zh-CN&lr=&id=VAwm6GGWxCgC&oi=fnd&pg=PR21&dq=Java+Game+Engine+Development+&ots=r8bY81ES7a&sig=Ee4-SxLnzmwl3kFupdg26ZlqObw#v=onepage&q&f=false>

Gustavsson, Mikael 2008. 3D Game Engine Design for Mobile Phones with OpenGL ES 2.0. Downloaded 15 March, 2011. <www.nada.kth.se/utbildning/grukth/exjobb/rapportlistor/2008/rapporter08/gustavsson_mikael_08024.pdf>

Knudsen, Jonathan 2007. Kicking Butt with MIDP and MSA: Creating Great Mobile Applications, Chapter 13. Downloaded 4 February, 2012. <http://books.google.fi/books?id=kIAtRLW5dXYC&pg=PT278&dq=M3G&hl=zh-CN&sa=X&ei=V1ItT6eAAan-4QSez6GADg&ved=0CD8Q6AEwAg#v=onepage&q=M3G&f=false>

Kumar, Ranjit 2011. Research Methodology. Downloaded 14 March, 2011. <www.google.fi/books?hl=zh-CN&lr=&id=i9NtQV-ZsZMC&oi=fnd&pg=PP1&dq=research+methodology+constructive&ots=ra02te046b&sig=xAYHepnsnbqpOPWrFG0ctU0F20A&redir_esc=y#v=onepage&q=research%20methodology%20constructive&f=false>

Muchow, John 2002. MIDlet development with J2ME and MIDP. Downloaded 4 February, 2012. <carfield.com.hk/document/java/tutorial/J2ME/j2me_guide.pdf>

Piroumian, Vartan 2002. Wireless J2ME platform programming. Downloaded 14 March, 2011. <http://www.google.com/books?hl=zh-CN&lr=&id=fvR4W81VTFMC&oi=fnd&pg=PT22&dq=J2ME+platform&ots=ZBjS5e7JNL&sig=w_TvlYwt318g5PsSdIaJ-0-d Rw8#v=onepage&q&f=false>

Pulli, Kari 2008. Mobile 3D Graphics: With OpenGL ES and M3G. Downloaded 18 March, 2012. <http://www.google.fi/books?hl=zh-CN&lr=&id=anzTkO5xJWsC&oi=fnd&pg=PP1&dq=M3G&ots=4fIAGSFIlG&sig=E4lFz-hjK2qFoAFmfP-lXNEVceI&redir_esc=y#v=onepage&q=M3G&f=false>

Root, Aaron & Donnelly, Christopher & Yeganov, Aleksander 2008. 3D Mobile Game Engine. Downloaded 7 March, 2011. <www.wpi.edu/Pubs/E-project/Available/E-project-042908-001255/unrestricted/MQPfinal.pdf>