Journal of the Korea Convergence Society
Vol. 9. No. 8, pp. 17-24, 2018

ISSN 2233-4890
https://doi.org/10.15207/JKCS.2018.9.8.017

안드로이드 게임 프로그래밍을 위한 설계 패턴

김동관
목포해양대학교 해양컴퓨터공학과

Design Patterns for Android Game Programming

Dong Kwan Kim
Department of Computer Engineering, Mokpo National Maritime University

요 약 설계 패턴은 소프트웨어 개발 시 반복적으로 발생하는 특정 문제들에 대한 효과적인 해결책을 제공하는 소프트웨어 재사용 기술이다. 특히, 객체지향 설계 패턴들은 다양한 플랫폼이나 프레임워크에 포함되어 소프트웨어 생산성을 높이고 있다. 본 논문은 설계 패턴을 고려한 안드로이드 모바일 플랫폼 기반의 게임 프로그래밍에 대한 지침을 제공한다. 적절히 설계 패턴을 활용함으로써 소프트웨어 개발 생산성뿐만 아니라, 개발 후 유지보수를 위해서도 효과적일 수 있다. 본 논문은 안드로이드 플랫폼 기반의 게임 프로그래밍에 설계 패턴을 적용하기 위한 지침과 사례를 제공한다. MVP, Singleton, Observer, State 설계 패턴과 같은 대표적인 객체지향 설계 패턴을 안드로이드 게임 프로그램 개발에 적용한다.

주제어 : 안드로이드 게임 프로그래밍, 설계 패턴, Unified Modeling Language, 소프트웨어 품질, 소프트웨어 재사용

Abstract Design patterns can be classified as software reuse technique that might provide effective solutions to specific problems that occur repeatedly in software development. In particular, object-oriented design patterns are incorporated into various software platforms and frameworks to increase software productivity. This paper aims to support general guidelines relating to design patterns for Android-based game programming. The proper use of the design pattern could be effective in not only for software development productivity, but also for post-development software maintenance. This paper provides fundamental procedures and case studies for applying design patterns to game programming on the Android platform. Typical object-oriented design patterns such as MVP, Singleton, Observer, and State have been applied to the development of Android game programs.

Key Words : Android Game Programming, Design Patterns, Unified Modeling Language, Software Quality, Software Reuse

1. 서론

모바일 프로그래밍을 위한 개발 도구, 개발 및 운영 플랫폼, 하드웨어의 대중화는 전문적인 프로그래머뿐만 아니라 일반인들도 손쉽게 앱 개발에 참여할 수 있는 환경을 제공하고 있다. 또한, 프로그래밍 즉, 코딩(Coding)의 대중화는 모바일 앱 영역에도 영향을 미쳐 앱 인벤터(App Inventor) [1]와 같은 보다 사용하기 쉬운 개발 환경이 배포되고 있다. 특히, 안드로이드 모바일 플랫폼 [2]과 같은 개방형 모바일 플랫폼은 소프트웨어 개발자들의 접근을 용이하게 한다. 안드로이드 스튜디오(Android Studio)와 같은 안드로이드 앱 개발을 위한 통합개발환경과 구글 플레이 스토어(Google Play Store)와 같은 온라인 저장소는 다양한 모바일 앱들을 유통할 수 있는 온라인 유통망을 제공한다. 이러한 보편화된 소프트웨어 개발 및 운영환경은 다양한 앱 개발을 촉진하고 있으며

*Corresponding Author : Dong Kwan Kim(dongkwan@mmu.ac.kr)
Received May 30, 2018 Revised July 10, 2018
Accepted Augutst 20, 2018 Published Augutst 28, 2018

그 중에서 게임 프로그래밍에 대한 관심은 증가하고 있다. 하지만, 일정 수준 이상의 게임 프로그램을 개발하는 일은 쉬운 일이 아니며 특히, 프로그래밍 경험이 많지 않은 경우는 완성된 게임 프로그램의 품질을 보장하기 힘들다.

소프트웨어 생산성과 품질을 함께 높이는 방법 중의 하나는 기존에 이미 검증된 개발 경험을 재사용하는 것이다. 소프트웨어공학에서는 이를 소프트웨어 재사용이라 정의하고 소프트웨어 프레임워크(Software Frameworks), 설계 패턴(Design Patterns), 소프트웨어 라이브러리(Software Libraries), 소프트웨어 컴포넌트(Software Components) 등 다양한 기법을 통해 소프트웨어 재사용에 대한 연구를 수행하고 있다. 모바일 컴퓨팅 환경에서 게임 프로그램을 개발하는 작업에도 소프트웨어 재사용 기법을 활용할 수 있으며 본 논문은 재사용 기법 중 설계 패턴의 재사용 방안을 제안한다.

설계 패턴은 소프트웨어 재사용 방식의 하나이며 소프트웨어 설계 시에 반복적으로 발생 가능한 문제들에 대한 체계적인 정의와 이에 대한 해결책을 형식화하여 표현한다. 소프트웨어 개발자들은 주어진 소프트웨어 개발 환경에서 설계 패턴에서 기술한 문제가 발생하면 이미 여러 차례에 걸쳐 검증된 해결책에 따라, 보다 효과적으로 설계 경험을 활용할 수 있다. 설계 패턴은 유사하게 반복되는 문제에 대한 전문가들의 지식이 반영된 해결책을 제공하므로 소프트웨어의 품질 향상을 기대할 수 있다. 설계 패턴을 적용함으로써 이미 설계된 모델의 구조가 변경될 수 있으며 이는 자연스럽게 구현 단계에도 영향을 미치게 된다. Erich Gamma외 3인은 총 23개의 대표적인 객체지향 패턴인 GoF 패턴들을 추출하고, 이들을 크게 객체생성패턴 (Creational Patterns), 행위 패턴 (Behavioral Patterns), 구조 패턴 (Structural Patterns) [3]으로 구분한다. 각 패턴을 형식화하여 표현하기 위해 패턴 별로 패턴 이름, 문제 및 배경, 해결책, 사례, 결과, 샘플 코드 등을 제시한다. 본 논문은 안드로이드 게임 프로그래밍에 GoF 패턴을 포함한 객체지향 설계 패턴을 적용한다.

본 논문은 다음과 같이 구성된다. 2장에서는 설계 패턴들을 적용한 안드로이드 기반 게임 프로그램 설계에 대해 기술한다. 3장에서는 2장에서 제시한 설계 패턴을 안드로이드 2D 게임 개발에 적용한 사례를 소개한다. 4장은 관련 연구를 소개하고, 5장은 결론과 향후 연구 방향을 제시한다.

2. 설계 패턴을 적용한 게임 프로그래밍

본 장에서는 설계 패턴을 고려한 일반적인 소프트웨어 개발 프로세스를 기술한다. 또한, 게임 프로그래밍 개발에 효과적인 구체적인 설계 패턴들을 제안하고 이를 기반으로 게임 프로그램 개발 프레임워크의 예를 소개한다.

2.1 설계 패턴을 고려한 소프트웨어 개발 과정

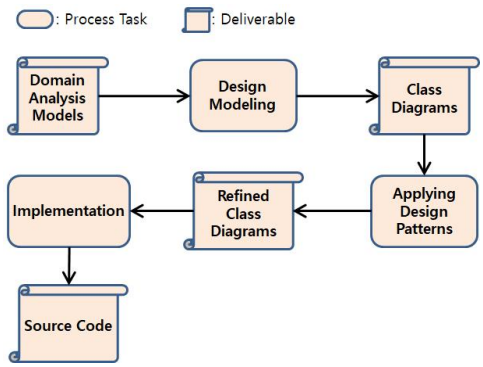

Fig. 1. Using Design Patterns in Software Development Process

Fig. 1은 소프트웨어 개발 프로세스 중 설계 패턴을 고려한 설계 및 구현 중심의 과정을 표현한다. 요구사항추출 과정을 통해 작성된 도메인 분석 모델을 입력으로 설계모델링 작업을 수행한다. 클래스다이어그램을 포함한 다양한 설계 다이어그램들이 산출된다. Fig. 1에서는 소프트웨어의 구조적 정보를 표현하는 클래스다이어그램만을 표현하고 있다. 설계 클래스다이어그램에 설계 패턴을 적용함으로써 보다 정제된 클래스다이어그램을 작성할 수 있다. 설계 패턴을 적용함으로써 소프트웨어 재사용성, 유지보수성, 확장성, 개발의 용이성, 가독성 등의 소프트웨어의 품질이 향상될 수 있다. 설계 패턴 적용 과정을 통해 기존 클래스다이어그램의 구조적 변경이 발생하므로 새로운 클래스나 인터페이스가 추가될 수 있고 삭제 또는 통합되는 클래스가 발생할 수 있다. 정제된 클래스다이어그램을 기반으로 구현 작업을 수행하며 프로그램 코드가 작성된다. 또한, 개발된 프로그램에는 소프트웨어의 실행속도 저하를 초래하거나 불필요하게 전력을 낭비하는 코드 패턴이 포함될 수 있으며 이를 제거하는 과정이 요구된다.

2.2 게임 프로그램을 위한 설계 패턴

본 절에서는 2D 기반의 일반적인 슈팅 게임에 설계 패턴을 적용한 사례들을 제공한다. GoF를 비롯한 설계 패턴이 슈팅 게임 프로그램에 적용되는 과정을 제안한다.

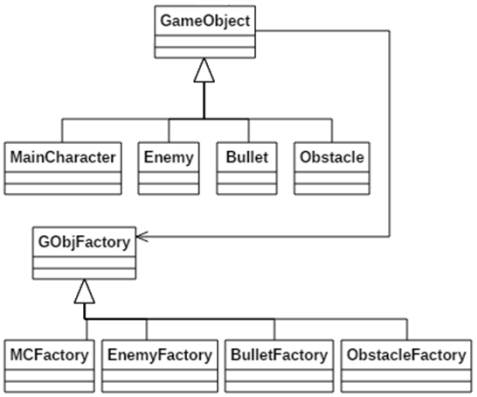

Fig. 2. Applying the Factory Method Design Pattern to Create the Instances of Game Objects

Fig. 2는 객체 생성 패턴 중 하나인 Factory Method 설계 패턴의 적용 사례이다. 슈팅 게임 클래스다이어그램에 포함된 GameObject 클래스를 상속한 MainCharacter, Enemy, Bullet, Obstacle 클래스의 객체를 생성하는 생성 메소드를 포함한 클래스들이 추가된다. MCFactory, EnemyFactory, BulletFactory, ObstacleFactory 클래스들은 각각 MainCharacter, Enemy, Bullet, Obstacle 클래스의 인스턴스를 생성한다. Factory Method 설계 패턴을 사용하면 클라이언트 코드에게 객체 생성과 관련된 내부 로직은 노출되지 않으며, 클라이언트는 GObjectFactory 클래스가 제공하는 공통된 인터페이스를 통해 각 클래스의 객체들을 생성할 수 있다. 또한, 차후에 새로운 게임 객체가 추가될 경우, 기 개발된 코드에 대한 수정을 최소화하면서 새로운 클래스의 객체 생성에 대한 로직 추가가 가능하다는 장점이 있다.

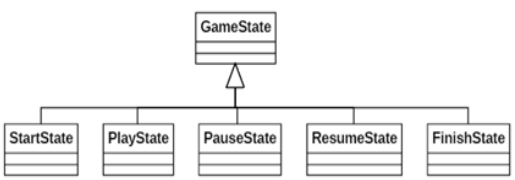

Fig. 3. Applying the State Design Pattern to Control Game States

Fig. 3은 슈팅 게임 클래스다이어그램에 State 설계 패턴을 적용한 사례이다. 슈팅 게임의 상태는 게임 시작, 게임 진행 중, 게임 일시 멈춤, 게임 재 시작, 게임 종료 등의 상태를 가질 수 있으며 GameState 클래스는 이러한 상태들을 단순하게 상수를 사용하여 관리할 수 있다. 또한, 게임 상태의 처리를 위해 다중 if-else문이나 switch 문 등의 다중 선택 구문을 사용해야 한다. 이러한 방법은 게임 상태의 수가 많아지면 코드가 복잡해지며 중복된 코드의 발생을 초래할 수 있어 유지보수 측면에서 바람직하지 않다. 하지만, State 설계 패턴을 사용하면 보다 효과적으로 각각의 게임 상태 변화에 따른 게임 진행을 제어할 수 있다. 또한 새로운 게임 상태의 추가도 비교적 용이하게 처리할 수 있다. Fig. 3에서 제시한 것처럼 각각의 상태를 개개의 클래스로 작성하고 각 상태 클래스들은 각각의 상태에 따라 고유의 상태 처리 메소드를 제공한다. 클라이언트는 GameState 상위 클래스가 제공하는 공통된 인터페이스를 통해 상태에 따른 처리 메소드를 호출할 수 있다.

```
private static GameMain gameInstance;

public static GameMain getInstance(){
    if(gameInstance == null){
        gameInstance = new GameMain();
    }
    return gameInstance;
}
```

Fig. 4. Code Example to Use the Singleton Design Pattern

Fig. 4는 슈팅 게임 프로그램 전체에 오직 하나의 인스턴스만 존재해야 하는 GameMain 클래스에 Singleton 설계 패턴을 적용한 코드 예제이다. GameMain 클래스는 게임 시작 시점에 게임을 위한 환경 설정, 게임 화면 객체 생성, 게임 참여자 등에 관한 정보를 초기화하므로 하나의 인스턴스만 생성하는 것이 일반적이다. Fig. 4에서 보듯이 클라이언트 코드는 getInstance 메소드를 통해 GameMain 인스턴스를 얻게 된다. GameMain 객체가 null인 경우만 새로운 인스턴스를 생성하며, 이미 객체를 생성한 경우에는 기 생성된 객체를 재사용할 수 있도록 반환함으로써, GameMain 인스턴스는 슈팅 게임이 실행되는 동안 오직 하나만 존재하게 된다. Singleton 설계 패턴을 사용함으로써 GameMain 인스턴스의 다중 생성으로 인해 발생할 수 있는 실행 오류를 사전에 방지할 수

있다.

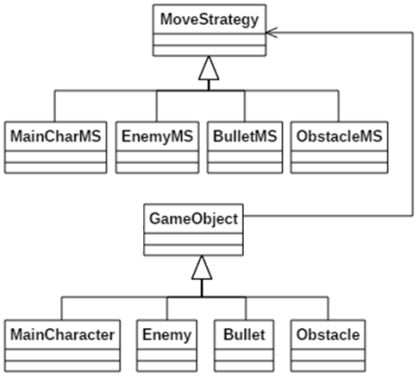

Fig. 5. Applying the Strategy Design Pattern to the Movement Algorithms of Game Objects

Fig. 5는 Strategy 설계 패턴을 적용한 사례를 제시한다. 게임에 참여하는 객체들은 움직이는 패턴, 속도, 방향 등에 따라 다양한 이동 알고리즘이 적용될 수 있다. Fig. 5는 게임 객체들에 따른 이동 방식을 별도의 클래스로 작성한다. 상위 클래스인 MoveStrategy는 클라이언트에게 공통된 인터페이스를 제공하며 해당 인터페이스는 하위 클래스에서 개개의 클래스에 적절하게 구현된다. 클래스는 동일한 메소드를 호출하지만 실제로 어떤 객체의 메소드가 호출되는 지는 프로그램 실행 시간에 결정된다. 다시 말해, 각각의 알고리즘을 별도의 클래스로 캡슐화함으로써 알고리즘의 대체가 보다 용이해진다.

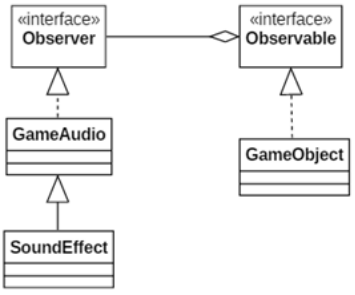

Fig. 6. Applying the Observer Design Pattern to the Game Sound Effects

일반적으로 슈팅 게임은 다양한 음향 효과를 사용하여 게임의 몰입도를 높이고자 한다. 게임 음향 효과의 종류는 게임의 초기 시작 오디오, 게임 중 배경 오디오, 물체 충돌 시 오디오, 게임 종료 시 오디오 등 다양하며 적절한 시점에 작동해야 한다. 이를 위해 Observer 설계 패턴을 활용할 수 있으며 Fig. 6은 Observer 설계 패턴을 적용한 사례를 제시한다. GameAudio 인스턴스는 관찰자(Observer)가 되며 GameObject 인스턴스는 관찰 대상(Subject)가 된다. Observer 설계 패턴은 관찰 대상이 되는 객체의 상태 변동을 실시간으로 모니터링할 수 있으며 변동이 발생하면 특정 기능을 수행한다. 특정 게임 객체가 게임 화면에 나타날 경우 특정 오디오가 재생되도록 할 수 있다. 또는, 화면상의 특정 영역에 게임 객체가 들어오면 지정된 오디오 파일을 재생할 수 있다. 특정 객체들끼리 충돌이 발생하는 경우도 실시간으로 모니터링할 수 있다.

게임 참여자의 흥미를 유발하기 위해 게임은 여러 단계의 레벨로 구성될 수 있다. 게임 레벨에 따라 게임에 등장하는 객체들, 배경들, 전략들이 변경될 수 있다. 일반적으로 레벨이 높아짐에 따라 게임의 난이도는 높아진다. 게임 난이도는 보다 강한 적군의 출현, 새로운 무기 등장, 기존 적군이나 무기들의 이동 패턴의 변화를 통해 조절될 수 있다.

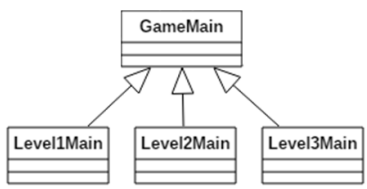

Fig. 7. Applying the State Design Pattern to Manage Game Levels

Fig. 7은 State 설계 패턴을 이용하여 게임 레벨을 표현하고 있다. 추상클래스인 GameMain을 상속 받아 각각의 게임 레벨에 따라 게임 레벨 관리자가 존재한다. 모든 게임 레벨에 공통되는 요소는 GameMain 클래스에 정의되며, 게임 레벨에 따른 차별화된 게임 객체 등은 각각의 게임 레벨 관리자가 관리한다. 새로운 게임 레벨을 추가하려면 해당 레벨을 관리하는 새로운 게임 레벨 관리자를 추가하면 된다.

기 언급한 GoF 설계 패턴 외에 안드로이드 아키텍처는 적절한 Model-View-Presenter(MVP) 설계 패턴[4]의 사용을 통해 프로그램의 모듈화를 높일 수 있을 것으로 기대한다. MVP 패턴은 잘 알려진 Model-View-Controller(MVC) 설계 패턴에 기반을 둔다. 안드로이드 아키텍처에서 MVC 설계 패턴을 사용할 경우, 일반적으

로 뷰, 컨트롤러, 데이터 소스 접근을 위한 코드가 액티비티(Activity)나 프래그먼트(Fragment)에 함께 포함될 수 있지만, MVP 패턴을 사용하면 로직 부분과 표현 부분을 용이하게 분리할 수 있다. 다시 말해, 관심사의 분리(Separation of Concerns)에 기반을 둔 코드 분리를 통해 프로그램 모듈화를 높여 소스 코드에 대한 이해, 기능 확장, 모듈 테스트, 유지보수를 용이하게 할 수 있다. MVP 패턴에서 표현(Presenter) 계층은 뷰(View) 계층과 모델(Model) 계층 사이에 위치하며, 모델 계층을 통해 데이터 소스로부터 데이터를 전달받고, 주어진 형식에 따라 뷰 계층으로 전달한다.

2.3 게임 프레임워크

본 절에서는 기 기술된 설계 패턴을 기반으로 한 안드로이드 게임 프레임워크와 이를 확장하여 슈팅 게임에 적용한다.

Fig. 8은 안드로이드 플랫폼 기반의 게임 프로그램 개발에 공통적으로 재사용될 수 있는 안드로이드 게임 프레임워크를 나타낸다. GameMain 클래스는 안드로이드 컴포넌트 중의 하나인 Activity를 상속하며 게임 프로그램을 시작하는 메소드를 포함하는 클래스로 특정 게임 프로그램이 실행 중일 경우 하나의 인스턴스만 생성하는 것이 효과적이다. GameMain 클래스는 GameInput 클래스를 통해 키보드 입력, 터치 입력, 자이로 센서 입력 등의 게임 진행을 위한 사용자 인터페이스를 정의한다. 또한, 다양한 게임 설정을 위해 GameConfig 클래스를 사용한다. GameView 클래스는 게임 화면 구성 등과 같이 게임 디스플레이를 위한 클래스로 그래픽 속도를 고려하여 안드로이드 플랫폼에 내장된 SurfaceView 클래스를 상속한다. 게임에는 다양한 개체들이 사용되며 경우에 따라, 독립적인 실행이 보장이 필요할 수 있으므로 쓰레드를 확장한 GameThread 객체를 사용한다. 그리고 게임은 다양한 게임 장면으로 구성될 수 있으므로 GameView 클래스는 GameScene 클래스를 통해 이를 구현한다. 또한 각 게임 장면은 다양한 게임 객체들을 포함할 수 있으며 GameObject가 사용된다. 또한, 게임의 상태 즉, 게임 시작, 게임 중, 게임 일시 멈춤, 게임 재 시작, 게임 종료 등을 GameState 클래스를 통해 관리한다. GameAudio 클래스는 게임의 초기 시작 오디오, 게임 중 배경 오디오, 물체 충돌 시 오디오, 게임 종료 시 오디오 등을 표현한다.

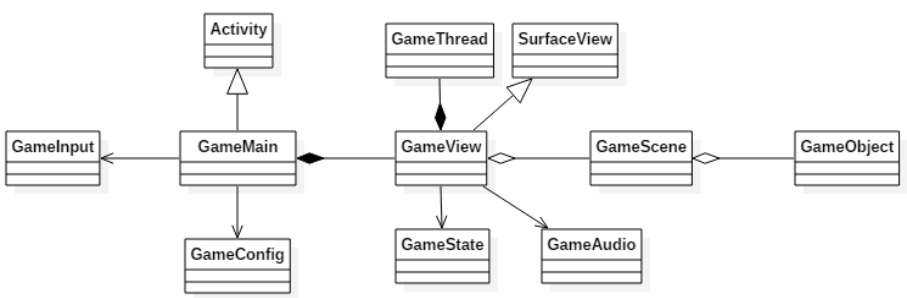

Fig. 8. An Example of the 2D-based Android Game Framework Including Common Game Classes

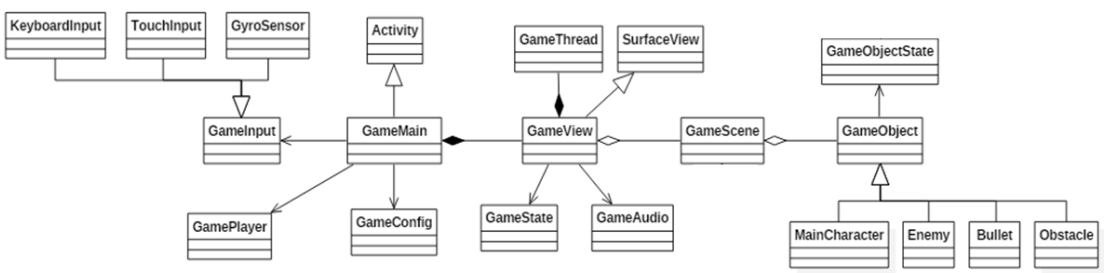

Fig. 9. Class Diagram for Shooting Games based on the Proposed Android Game Framework

Fig. 9는 제시한 안드로이드 게임 프레임워크를 기반으로 한, 2D 슈팅 게임을 위한 클래스다이어그램을 나타낸다. 해당 클래스다이어그램은 안드로이드 게임 프레임워크를 뼈대로 하고 있으며 GameInput 클래스를 확장한 KeyboardInput, TouchInput, GyroSensor 클래스를 포함한다. 이 외에도 제공되는 게임 인터페이스에 따라 원하는 클래스를 추가할 수 있다. GameObject 클래스는 게임에 등장하는 객체들을 표현한 것이므로 슈팅 게임에서는 GameObject 클래스를 확장하여 MainCharacter(주인공), Enemy(적군), Bullet(총알), Obstacle(장애물) 클래스를 제공한다. GamePlayer 클래스는 게임 참여자에 관한 데이터 즉 참여자 아이디, 득점, 로그 기록 등을 관리한다. GameObjectState 클래스는 게임에 사용되는 다양한 게임 객체들의 상태 즉, 활성화상태, 잠시 멈춤 상태, 종료 상태 등을 관리한다.

3. 사례연구

본 장에서는 앞에서 기술한 설계 패턴들을 5개의 안드로이드 2D 게임 개발에 적용한 사례를 제공한다. 사격 게임(Shooting Game), 스네이크 게임(Snake Game), 테트리스 게임(Tetris Game), 리듬 게임(Rhythm Game), 틱택토 게임(Tic-Tac-Toe Game)이 사례연구를 위해 사용되었으며, Table 1은 게임 종류별 요구사항에 따라 활용 가능한 설계 패턴을 요약 정리한다. Table 1에서 두 번째 열은 각 게임별 요구사항을 기술한다. 해당 요구사항을 효과적으로 해결할 수 있는 설계 패턴이 세 번째 열에 제공된다.

Table 1. Design Patterns Applied for Five Android 2D Games

Game Types	A List of Requirements	Design Patterns
Shooting Game	To provide interfaces for the creation of game object instances	Factory Method
	To build multiple game states as individual classes which can hold proper fields or methods to manage each game state effectively	State
	To create only one instance of the GameMain class which may be shared by multiple objects in the shooting game	Singleton
	To provide the same interface for the movement of game objects and to hide the internal implementation details of the interface	Strategy
	To support multiple game sound effects according to game contexts	Observer
	To represent the degree of difficulty of the shooting game	State
	To improve the modularity and testability of the architecture of the game	MVP
Snake Game	To create only one instance of the GameBoard class when playing the game	Singleton
	To express game states such as StartState, PauseState, RunningState, and FinishState in the Presenter layer of the MVP	State
	To make the snake and its targets move concurrently when playing the game	Game Thread
	To implement different game levels-as the game level increases, the target of the snake can change its location periodically	State
	To play game sound effects such as intro, background, and collision sounds	Observer
	To improve the modularity and testability of the architecture of the snake game	MVP
Tetris Game	To make only one instance of the TetrisView class in the Tetris game	Singleton
	To instantiate the block of the different shape flexibly and independently	Factory Method
	To implement the concrete algorithm of determining the block's position or shape	Strategy
	To generate the block and move it downwards after a certain amount of time	Game Thread
	To increase the degree of coupling of an Android activity or fragment by placing correlated code into one place	MVP
Rhythm Game	To manage multiple game states and changes from one state to another	State
	To create the instances of the game screens such as InitialScreen, ScoreScreen, LoadingScreen, and GameScreen	Factory Method
	To support various rhythm game modes with changeable movement patterns of notes and hitboxes	Strategy
	To run a game loop for creating and starting the next note concurrently	Game Thread
	To make the views of the rhythm game independent from the models for better modularity and testability	MVP
Tic-Tac-Toe Game	To make sure there is only one model of the tic-tac-toe game at any time	Singleton
	To create various game boards with different properties such as board size, color, and player	Factory Method
	To implement movement algorithms of the computer player	Strategy
	To implement the states of the tic-tac-toe game such as RunningState and FinishState	State
	To support the separation of concerns-TicTacToeModel(Board, BoardCell, Player), TicTacToeView, and TicTacToePresenter	MVP

4. 관련연구

설계 패턴을 이용한 소프트웨어 재사용 기술은 설계 패턴 정의, 설계 패턴 자동 추출, 설계 패턴 검증 등 다양한 분야에서 연구되고 있다. 또한, 프로그래밍 언어나 소프트웨어 플랫폼 개발 시 내부적으로 기존 설계 패턴이나 이를 확장한 패턴들을 이용하고 있다. 최근에는 설계 패턴을 모바일 게임 프로그래밍에 적용하여 게임 프로그램의 생산성과 품질을 높이려는 연구들이 소개 되고 있다. 이미 다양한 응용 도메인에서 활용되고 있는 GoF 패턴들을 게임 프로그램 개발에 적용하기 위한 방법, 사례, 지침 등이 제시된다. [5]에서는 추상화된 GoF 패턴들을 소스 코드 수준에서 활용할 수 있도록 게임 규칙과 게임 로직에 적용한 사례를 제공하며, [6, 7]에서는 슈팅 게임의 구성요소인 적군, 무기, 통신, 제어 등을 위한 설계 패턴을 분석하고 이를 위해 기존 GoF 패턴을 활용한 적용 사례를 제시한다. 기존의 GoF 패턴 외의 다른 설계 패턴들을 식별, 정의, 적용하여 게임 프로그램에 대한 개발 생산성 및 유지보수성을 높이려는 연구들도 소개되고 있다. [8]은 Document-View 패턴을 적용하여 초보 프로그래머가 보다 쉽게 게임 제작을 할 수 있는 안드로이드 게임 프레임워크를 제시하며, [9]에서는 Linked Observer 패턴을 중심으로 안드로이드 게임 프레임워크를 설계하고 이를 적용한 사례를 기술한다. 또한, [10]에서는 멀티 플레이어 카드 게임을 위한 설계 패턴들을 추출 및 정의하고 적용 사례를 제시한다. [11]에서는 대표적인 모바일 플랫폼 중의 하나인 J2ME 기반의 게임 프로그래밍을 위한 설계 패턴을 정의하고, MVC 패턴에 기반을 둔 숫자 퍼즐게임 제작에 적용한 사례를 제공한다. [12]에서는 안드로이드 플랫폼 기반의 게임 콘텐츠 개발을 위한 개발 도구, 게임 제작 과정, 콘텐츠 개발 사례를 제시한다.

5. 결론 및 향후연구방향

본 논문은 게임 도메인이 아닌 다른 응용 도메인에서 이미 다양한 장점을 입증한 설계 패턴을 게임 도메인에 적용하였다. 초보 프로그래머도 보다 용이하게 게임 프로그램 개발의 생산성 및 유지보수성을 높일 수 있도록, GoF 설계 패턴을 포함한 객체지향 설계 패턴을 안드로이드 게임 개발에 적용하기위한 지침을 제공하였다. 슈팅, 스네이크 게임 등과 같은 2D 기반의 게임을 개발할 때, 공통적으로 발생할 수 있는 게임 요구사항을 선정하고 이를 해결하기 위해 적절한 설계 패턴을 적용하였다.

향후 연구주제로 설계 패턴을 유형화하여 게임 유형에 무관하게 공통적으로 적용할 수 있는 설계 패턴과 게임 유형에 따라 특화된 설계 패턴을 제안할 수 있을 것이다. 또한, 설계 패턴 적용 효과를 실험적 또는 수학적으로 제시함으로써 소프트웨어 개발 생산성이나 유지보수에 미치는 영향을 구체화할 수 있을 것이다. 효과적인 프로그램 개발을 위해 설계 패턴뿐만 아니라 안티 패턴(Anti-Patterns)을 고려해야 한다[13]. 안티 패턴은 소프트웨어 개발 시에 반복해서 발생할 수 있는 오류를 표현한다. 모바일 프로그램의 경우, 안티 패턴으로 인해 성능 저하, 메모리 손실, 전력 손실, 네트워크 속도 저하, 보안 취약 등의 문제점[14, 15]이 초래될 수 있으므로 안티 패턴 제거를 위한 연구가 필요하다.

REFERENCES

[1] MIT App Inventor, http://appinventor.mit.edu/explore/.

[2] *Android Developer Website*. Available: https://developer.android.com/index.html.

[3] E. Gamma, R. Helm, R. Johnson & J. Vlissides. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

[4] Vogella GmbH, *Android Architecture with MVP or MVVM, Tutorial,* http://www.vogella.com/tutorials/AndroidArchitecture/article.html.

[5] X. Kounoukla, A. Ampatzoglou & K. Anagnostopoulos. (2016). Implementing Game Mechanics with GoF Design Patterns, *in Proceeding of the 20th Pan-Hellenic Conference on Informatics* (pp. 1-4). Patras.

[6] J. Qu, Y. Wei & Y. Song. (2014). Design Patterns Applied for Networked First Person Shooting Game Programming, *in Proceeding of the Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* (pp. 1-6). Las Vegas.

[7] J. Qu, Y. Song & Y. Wei. (2016). Design Patterns Applied for Game Design Patterns, *in Proceeding of the Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing* (pp. 351-356). Las Vegas.

[8] Y. Choi. (2016). Study on Android Game Framework Using Document-View Pattern (DVAGF), *The Journal of Korea Multimedia Society, 19(4),* 789-795.

[9] M. Seo. (2013). Design and Implementation for Android Game Framework Using the Linked Observer Pattern, *The Journal of Information Technology Services, 12(3),* 421-432.

[10] D. V. Ranganathan, R. Vishal, V. Krishnamurthy & R. (2017). Devarajan, Design Patterns for Multiplayer Card Games, *in Proceeding of the IEEE International Conference on Computer, Communication, and Signal Processing* (pp. 1-3), Chennai.

[11] J. Narsoo, M. S. Sunhaloo & R. Thomas. (2009). The Application of Design Patterns to Develop Games for Mobile Devices using Java 2 Micro Edition, *The Journal of Object Technology, 8(5)*, 153-175.

[12] T. E. Kim. (2016). Study on development of the game content based on the Android, *The Journal of Digital Contents Society, 17(2)*, 105-109.

[13] G. Hecht, R. Rouvoy, N. Moha & L. Duchien. (2015). Detecting Antipatterns in Android Apps, *in Proceeding of the 2nd ACM Mobile Software Engineering and Systems*(pp. 148-149). Florence.

[14] L. Cruz & R. Abreu. (2017). Performance-based Guidelines for Energy Efficient Mobile Applications, *in Proceeding of the IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems*(pp. 46-57). Buenos Aires.

[15] M. Brylski. *Android Code Smells Catalogue.* http://www.modelrefactoring.org/smell_catalog/.

김 동 관(Kim, Dong Kwan) [정회원]

- 1993년 2월 : 숭실대학교 전산학과(공학사)
- 1998년 2월 : 숭실대학교 전산학과(공학석사)
- 2009년 7월 : Virginia Tech 컴퓨터과학과(공학박사)
- 2013년 3월 ~ 현재 : 목포해양대학교 해양컴퓨터공학과 교수
- 관심분야 : 소프트웨어공학, 모바일 프로그래밍, 런타임 시스템, 안드로이드 모바일 시스템
- E-Mail : dongkwan@mmu.ac.kr