멀티플랫폼 게임 엔진을 이용한 2D 게임 설계 및 구현

Design and Implementation for Multi-Platform Game Engine using the 2D Game

배재환[1)]

Jae-Hwan Bae[1]

요약

멀티 플랫폼(Multi-Platform)은 게임이 여러 종류의 플랫폼(Platform)에서 구동한다는 것을 말한다. 여기서 말하는 플랫폼이란 하드웨어(Hardware)가 될 수도 있고, OS(Operation System)가 될 수도 있다. 현재 출시된 멀티플랫폼게임엔진중 게임 개발사에서 많이 사용중인 엔진은 Unity3D 엔진이다. 본 논문에서는 Unity3D 엔진을 이용한 게임을 설계 개발 하고자 한다. 유니티 엔진은 C#, 자바스크립트, Boo로 코드를 작성할 수 있다는 이유로 C#과 모노 기반 코드로 개발되었다고 알려져 있었다. 실제로 엔진의 런타임 부분은 C++과 마이크로소프트 닷넷 API, 에디터 프로그램은 C#으로 개발되었다. 스크립트는 유니티 내에서 바로 수정은 하지 못하고 Mono Develop등 유니티를 지원하는 스크립트 에디터에서 수정할 수 있다.이에 본 논문에서는 멀티플랫폼 게임 엔진을 이용한 2D 게임 설계 및 구현을 제안 하고자 한다. 이를 통해서 다양한 멀티 플랫폼 기반의 게임 설계 및 개발에 도움이 되었으면 한다.

핵심어: 멀티 플랫폼 게임 엔진, 멀티 플랫폼 게임, 멀티 플랫폼, 2D 게임

Abstract

Multi-Platform means that the game runs on different kinds of platforms. The platform referred to here may be hardware or may be an operating system (OS). Unity3D engine is one of the most popular multiplatform game engines in game development. In this paper, we design and develop games using Unity3D engine. The Unity engine was known to have been developed in C # and mono-based code because of its ability to write code in C #, JavaScript, and Boo. In fact, the run-time part of the engine was developed with C ++ and Microsoft .NET APIs, and the editor program with C #. Scripts can not be modified directly in Unity and can be modified in Script editor supporting Unity such as Mono Develop. In this paper, we propose the design and implementation of 2D game using multi-platform game engine. We hope this will help you design and develop various multiplatform based games.

Keyword: Multi-Platform Game Engine, Multi-Platform Game, Multi-Platform, 2D Game

1) Department of Game Engineering, TongMyong Univ., Sinseon-ro, Nam-gu, Busan, 608-711, Korea
e-mail: bjhmail@tu.ac.kr
*"이 논문은 2017학년도 동명대학교 교내학술연구비 지원에 의하여 연구 되었음(과제번호 2017F004)"
Received(April 24.2017), Review (June 11.2017), Accepted(June 30.2017)

1. 서론

멀티 플랫폼(Multi-Platform)은 게임이 여러 종류의 플랫폼(Platform)에서 구동한다는 것을 말한다. 여기서 말하는 플랫폼이란 하드웨어(Hardware)가 될 수도 있고, OS(Operation System)가 될 수도 있다. 현재 출시된 멀티플랫폼게임엔진중 게임 개발사에서 많이 사용 중인 엔진은 Unity3D 엔진이다. 유니티 엔진은 C#, 자바스크립트, Boo로 코드를 작성할 수 있다는 이유로 C#과 모노 기반 코드로 개발되었다고 알려져 있었다. 실제로 엔진의 런타임 부분은 C++과 마이크로소프트 닷넷 API, 에디터 프로그램은 C#으로 개발되었다[1][2]. 스크립트는 유니티3D 내에서 바로 수정은 하지 못하고 Mono Develop등 유니티를 지원하는 스크립트 에디터에서 수정할 수 있다[3]. 이에 본 논문에서는 멀티플랫폼 게임 엔진을 이용한 2D 게임 설계 및 구현을 제안 하고자 한다. 논문의 주요 구성은 2장 게임엔진기술분석, 3장 멀티플랫폼게임엔진, 4장 게임기술 분석 및 게임 구현, 5장 결론으로 구성 된다.

2. 게임엔진 기술분석

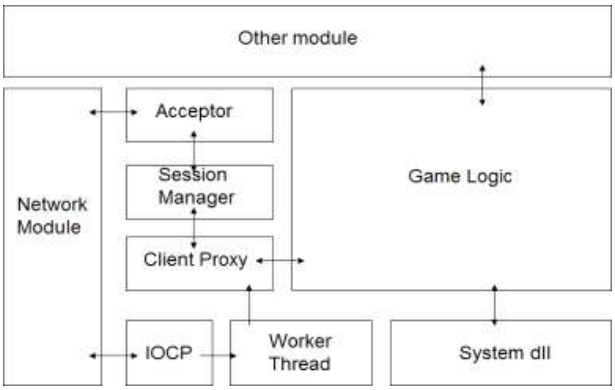
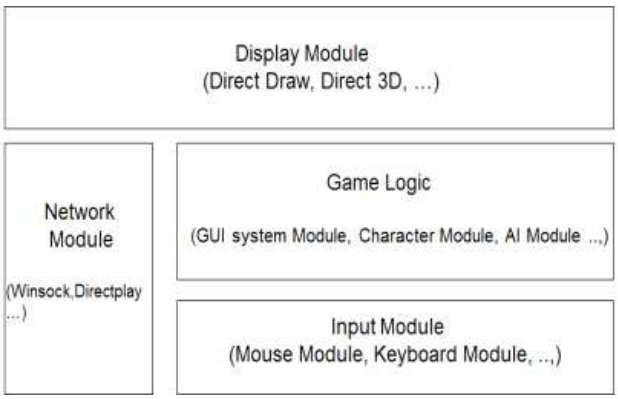
2.1 게임엔진 기술 분석

게임엔진의 개념이 본격적으로 자리를 잡은 것은 1996년 발매된 FPS(First Person Shooter)게임 퀘이크(Quake)이후 3D게임이 보편화 되면서부터였다. 퀘이크의 개발자 존 카멕이 퀘이크 시리즈의 소스코드를 공개하며 게임엔진이라는 개념이 자리 잡은 것이다. 상용 게임엔진에는 3D 게임의 구동을 위한 렌더링(Rendering) 등의 기능이 미리 구현되어있기 때문에, 이를 이용함으로써 게임 개발 기간을 단축할 수 있다. 게임엔진이 가지는 주요 기능으로는 3D 그래픽의 표현을 위한 렌더링 엔진(Rendering Engine), 3D 공간의 충돌 감지 및 현실적인 물리 효과(Physics Effect)를 내기 위한 물리엔진(Physics Engine), 각종 개발 도구(Development Tool) 등이 있다. 돈을 받고 라이선스(Licence)를 판매하는 상용 게임엔진의 경우, 개발자의 편의와 개발 속도 향상을 위한 스크립트 에디터(Script Editor), 맵 에디터(Map Editor) 등의 다양한 기능들을 제공하고 있다. 대표적인 상용 게임엔진으로는 언리얼 엔진(Unreal Engine), 크라이 엔진(Cry Engine), 소스 엔진(Source Engine)등이 있다. 언리얼 엔진과 크라이 엔진, 소스 엔진은 FPS 게임 개발을 위한

엔진인 만큼, FPS나 TPS(Third Person Shooter) 게임을 개발하는 데 용이하나, 각종 부가 기능들을 이용해 MMORPG(Massively Multiplayer Online Role-Playing Game)나 액션 게임(Action Game) 등을 개발하기도 한다. 일반적인 게임 서버 & 클라이언트 엔진 모델은 아래와 같다. 게임 엔진은 게임 소프트웨어의 구성에 필요한 소프트웨어 구성 요소를 재사용할 수 있게 만든 것이다. 엔진은 개발에 필요한 기능을 즉시 사용할 수 있도록 제공하여 개발의 단가와 복잡도를 줄여주고, 제 일정에 복잡한 게임을 출시할 수 있게 해 준다. 이것은 게임 업계에서 매우 중요한 일이다.

게임 엔진은 유연하고 재사용 가능한 소프트웨어 환경을 제공해주는 점 때문에 '게임 미들웨어'라고 불리기도 한다. 게임 미들웨어라고 불리는 시스템들은 구성 요소(컴포넌트) 기반의 구조를 가지는 것이 많다[4][5].

[표 1] 게임 서버 및 클라이언트 구조
[Table 1] Game server and client structure

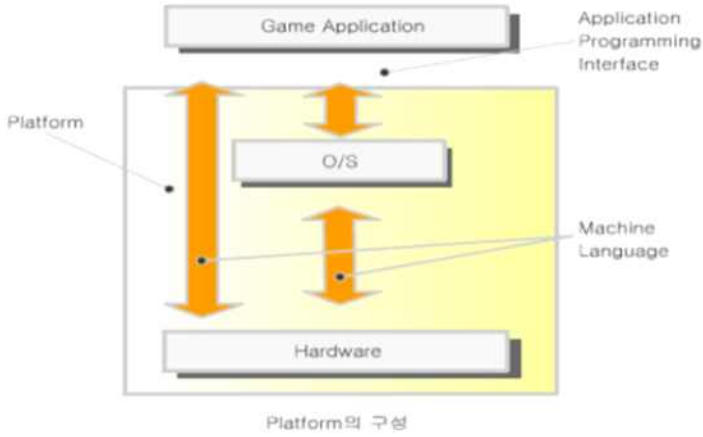
구분	Game Server & Client 구조
서버	
클라이언트	

3. 게임기술 분석 및 게임 구현

멀티 플랫폼 게임 개발을 위해서는 멀티 플랫폼 개발이 가능한 언어를 사용하거나 멀티 플랫폼을 지원하는 게임 엔진을 구매해 게임을 개발하는 게 일반적이다. 특히 직접 멀티 플랫폼을 지원하는 게임 엔진(Game Engine)을 만들 경우, 각 플랫폼이나 기기에 따른 대응이 어려워지기 때문에 개발 효율과 위험성 관리를 위해 멀티 플랫폼을 지원하는 게임 엔진을 구매하는 경우가 많다. 보통 멀티 플랫폼이라 하면 여러 기기등에서 구동이 된다는 의미로 사용하지만, 아예 다양한 플랫폼에서 연동되는 멀티 플레이(Multi-Play)를 지원하는 경우도 있다. 세가가 개발한 액션 MORPG(Multiplayer Online Role-Playing Game) 판타시 스타 온라인 2(Phantasy Star Online 2)에서는 플레이스테이션 3와 PS Vita, PC 사용 유저가 함께 즐길 수 있는 멀티플레이를 선보인 바 있다.

멀티 플랫폼(Multi-Platform)은 게임이 여러 종류의 플랫폼(Platform)에서 구동한다는 것을 말한다. 여기서 말하는 플랫폼이란 하드웨어(Hardware)가 될 수도 있고, OS(Operation System)가 될 수도 있다. 현재 출시된 멀티플랫폼게임엔진중 게임 개발사에서 많이 사용중인 엔진은 Unity3D 엔진이다.[6-8] 본 논문에서는 Unity3D 엔진을 이용한 게임을 설계 개발 하고자 한다. 유니티 엔진은 C#, 자바스크립트, Boo로 코드를 작성할 수 있다는 이유로 C#과 모노 기반 코드로 개발되었다고 알려져 있었다. 실제로 엔진의 런타임 부분은 C++과 마이크로소프트 닷넷 API, 에디터 프로그램은 C#으로 개발되었다. 스크립트는 유니티 내에서 바로 수정은 하지 못하고 Mono Develop등 유니티를 지원하는 스크립트 에디터에서 수정할 수 있다.

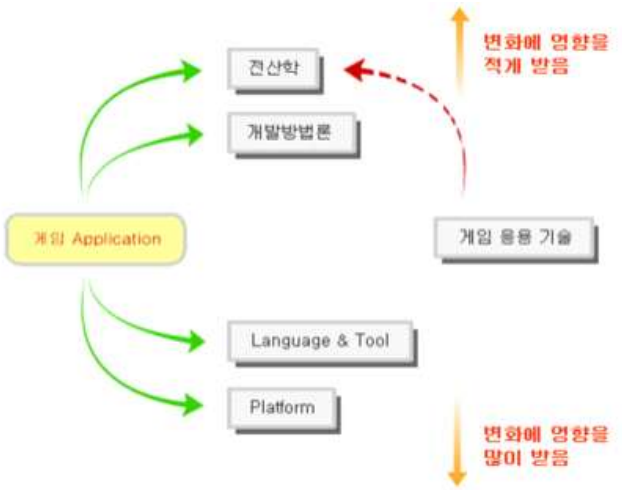
Visual Studio Tools for Unity[2] 플러그인을 이용해서 Visual Studio를 통해 유니티 C# 스크립트를 개발 및 디버깅할 수 있으며, 해당 플러그인은 비주얼 스튜디오 커뮤니티(Visual Studio Community) 버전 이상에서 사용 가능하다. 유니티는 Premium Support 솔루션을 가지고 있다. 이 솔루션을 이용하면 유니티3D에 질문을 하고 하루 내로 답변을 받을 수 있다.



[그림 1] 플랫폼 구성
[Fig. 1] Platform Configuration

4. 게임기술 분석 및 게임 구현

게임 기술이란 부분은 포괄적인 응용 부분이라서 정의하기가 쉽지 않다. 만약 기반기술이라면 비교적 명확한 선과 경계선이 있으나 게임 기술은 기반기술들을 이용하는 응용기술이므로 그 경계가 명확하지 않다.



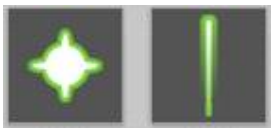
[그림 2] 게임 프로그래밍 다이어그램
[Fig. 2] Game Programming Diagram

그래서 게임 기술은 게임을 제작하는데 있어서 필요한 기술로 정의 할 수 있을 것이다. 즉 기술의 영역을 사용상의 범위 혹은 필요상의 범위로 한정할 수 있을 것이다. 게임에 사용하는 기술은 여려가지 상황에 따라 다를 수 있으나 대략 게임을 직접적으로 개발하는 프로그래밍 기술과 게임에 필요한 그래픽이나 사운드 등을 제작하는 제작기술 그리고 그 공정을 관리하는 관리기술이라고 정의할 수 있다[9].

4.1 게임 오브젝트 구현

게임오브젝트란, 게임에 등장하는 캐릭터, 건물, 아이템등과 같은 객체들을 의미한다. 게임 오브젝트 시스템 설정은 변수(Parameter), 작용장치(System), 명령체계(Command System), 상호작용(Interaction System)으로 나눌 수 있다.

[표 2] 게임 오프젝트
[Table 2] Game Object

게임 오브젝트(Game Object)	
아군	
적군	
장애물	
파티클	

4.2 게임 프로그래밍 구현

게임 프로그래밍 기술은 기반학문인 전산학과 이를 기반으로 한 응용기술, 언어와 Platform 기술 등이 있다. 게임 시스템이 복잡해짐에 따라 Platform은 크게 H/W 와 S/W로 구성되어 있으며, 각 단계를 이어주는 여러 인터페이스로 구성 된다고 할 수 있다. 아래에 논문에서 설계 개발하고자 하는 프로그래밍 소스 일부를 추가 하였다.

- **Player_Script**

```
using UnityEngine;
using System.Collections;

[System.Serializable]
public class Boundary
{
public float xMin, xMax, yMin, yMax; //Screen Boundary dimentions
}

public class Player_Script : MonoBehaviour
{
//Public Var
public float speed; //Player Ship Speed
public Boundary boundary; //make an Object from Class Boundary
public GameObject shot;//Fire Prefab
public Transform shotSpawn;//Where the Fire Spawn
public float fireRate = 0.5F;//Fire Rate between Shots
public GameObject Explosion;//Explosion Prefab

//Private Var
private float nextFire = 0.0F;//First fire & Next fire Time


// Update is called once per frame
void Update ()
{
//Excute When the Current Time is bigger than the nextFire time
if (Time.time > nextFire)
{
nextFire = Time.time + fireRate; //Increment nextFire time with the current system time + fireRate
Instantiate (shot , shotSpawn.position ,shotSpawn.rotation); //Instantiate fire shot
```

```
GetComponent<AudioSource>().Play (); //Play Fire sound
}
}

// FixedUpdate is called one per specific time
void FixedUpdate ()
{
float moveHorizontal = Input.GetAxis ("Horizontal"); //Get if Any Horizontal Keys pressed
float moveVertical = Input.GetAxis ("Vertical");//Get if Any Vertical Keys pressed

Vector2 movement = new Vector2 (moveHorizontal, moveVertical); //Put them in a Vector2 Variable (x,y)

GetComponent<Rigidbody2D>().velocity = movement * speed; //Add Velocity to the player ship rigidbody

//Lock the position in the screen by putting a boundaries
GetComponent<Rigidbody2D>().position = new Vector2
(
Mathf.Clamp (GetComponent<Rigidbody2D>().position.x, boundary.xMin, boundary.xMax),   //X
Mathf.Clamp (GetComponent<Rigidbody2D>().position.y, boundary.yMin, boundary.yMax)   //Y
);
}

//Called when the Trigger entered
void OnTriggerEnter2D(Collider2D other)
{
//Excute if the object tag was equal to one of these
if(other.tag == "Enemy" || other.tag == "Asteroid" || other.tag == "EnemyShot")
{
Instantiate (Explosion, transform.position , transform.rotation); //Instantiate Explosion
SharedValues_Script.gameover = true //Trigger That its a GameOver
Destroy(gameObject); //Destroy Player Ship Object
}
}
}
```

4.2 게임 플레이

논문에서 프로토타입으로 설계 및 개발한 멀티플랫폼 게임 엔진을 이용한 2D의 실행 화면은 아래와 같다.

[그림 3] 게임 플레이 화면
[Fig. 3] Game play screen

5. 결론

멀티 플랫폼(Multi-Platform)은 게임이 여러 종류의 플랫폼(Platform)에서 구동한다는 것을 말한다. 여기서 말하는 플랫폼이란 하드웨어(Hardware)가 될 수도 있고, OS(Operation System)가 될 수도 있다. 현재 출시된 멀티플랫폼게임엔진중 게임 개발사에서 많이 사용중인 엔진은 Unity3D 엔진이다. 본 논문에서는 Unity3D 엔진을 이용한 게임을 설계 개발 하고자 한다. 유니티 엔진은 C#, 자바스크립트, Boo로 코드를 작성할 수 있다는 이유로 C#과 모노 기반 코드로 개발되었다고 알려져 있었다. 실제로 엔진의 런타임 부분은 C++과 마이크로소프트 닷넷 API, 에디터 프로그램은 C#으로 개발되었다. 스크립트는 유니티 내에서 바로 수정은 하지 못하고 Mono Develop 등 유니티를 지원하는 스크립트 에디터에서 수정할 수 있다.이에 본 논문에서는 멀티플랫폼 게임 엔진을 이용한 2D 게임 설계 및 구현을 제안 하고자 한다. 이를 통해서 다양한 멀티 플랫폼 기반의 게임 설계 및 개발에 도움이 되었으면 한다.

ISSN: 2508—9099 JDAEM

References

[1] Bae, Jae-Hwan. "Design and Analysis of Creativity Based Infants Learning Game." Korea Computer Game Association Spring Conference, (2009):118-125.

[2] Lee, Jeong-Gwan. "Theological basis of the education ministry in light of Calvin's doctrine of Education." Theological Criticism 21 (2008): 387-406.

[3] Lee, Eun-Kyoo. "Research on Westminster Shorter Catechism as Curriculum for Christian Youth." The Korean Society for Practical Theology 26.2 (2011): 247-275.

[4] Kwon, Ho. "Preaching the Heidelberg Catechism for the Reformed Church Education", The Society of Reformed Theology 28 (2013): 215-246.

[5] You, Gi-Won and Yoon, Seon-Jeong. "Serious Game Design and Implementation for Kids." Journal of Korea Game Society 15.4 (2015): 19-28.

[6] Park, Jong-Seok "Review of Christian educational computer games." Christian and Education 10 (2002): 46-68.

[7] Park, Joong-Soo. "Design and Prototyping of a Bible Game in Smart Platform." Korean Society For Computer Game 26.4 (2013): 99-104.

[8] Kwon, Yong-Man. "A Study on the Game Definition & Its Attributions." Korean Society For Computer Game, 27.4 (2014): 221-227.

[9] Kim, Na-Young. "A research on good game factors for designing educational game at an early development stage." Korean Society For Computer Game 28.2 (2015): 53-61.