유니티 3D 엔진을 이용한 모바일 아케이드 게임 개발

김수균*, 강지훈*, 송기섭*, 이희범*, 안성옥*

요약

지난 2009년 스마트폰이 도입되고 스마트폰을 이용한 모바일 콘텐츠 부문과 앱 스토어의 급성장으로 인하여 현재 국내 스마트폰 사용 인구는 3천만 명을 돌파하였다. 이러한 모바일 시장의 급성장과 스마트폰 하드웨어의 발달로 인해 PC 및 콘솔 기반 게임 사용자들이 모바일 게임으로 이동하고 있으며, 게임 엔진 개발사에서는 다양한 플랫폼에 대응하기 위한 엔진들이 선보이고 있다. 본 논문에서는 사전 준비를 요구하는 저 수준의 게임 개발 방식이 아닌 유니티 3D 엔진을 이용하여 복잡한 부분을 상대적으로 사용하기 쉽게 처리하고 최적화, G-센서(Gyro 센서), 셰이더 등과 같이 다루기 어려운 부분들을 쉽게 처리하는 방법에 대하여 설명한다. 또한, 10~20대 여성 사용자를 대상으로 하는 캐릭터 컨셉과 쉬운 인터페이스, 간편한 조작을 기반으로 개발된 캐주얼 아케이드 게임을 구현하는 방법에 대해 제안한다.

Development of Mobile Arcade Game using Unity 3D Engine

Soo-Kyun Kim*, Ji-Hoon Kang*, Gi-seob Song*, Hee-bum Lee*, Syung-og An*

Abstract

Ever since the introduction of Iphone in 2009, mobile contents and app store had been rapidly developing, and smartphone population in Korea has now reached 30 million consumers. During this period of rapid growth, a lot of game users have shifted from pc-based online game to mobile games, and new game engines are continuously coming out to counter against the wide variety of online game platforms. This paper will explain how the past method of game-making process that requires complex advance preparation can be replaced by an easier and simpler method through the usage of Unity 3D Engine. It will show how Unity 3D Engine aids in complicated processes like optimization, G-Senser, and Shader. Additionally, the paper will discuss how to actualize arcade games that are developed based on easy interface, simple controls, and character concepts targeted for teenage girls to women in twenties.

Key Words : G-sensor, Shader, Unity3D engine, Casual Game

---

* 배재대학교 게임공학과 (✉ kimsk@pcu.ac.kr)
· 제1저자(First Author) : 김수균 · 교신저자(Correspondent Author) : 안성옥
· 접수일(2013년 2월 5일), 수정일(1차 : 2013년 2월 28일), 게재확정일(2013년 3월 26일)

Ⅰ. 서 론

지난 2009년 스마트폰이 도입되고 스마트폰을 이용한 모바일 콘텐츠 부문과 앱 스토어의 급성장으로 인하여 현재 국내 스마트폰 사용 인구는 3천만 명을 돌파하였다. 이러한 모바일 시장의 급격한 변화로 인하여 핸드폰은 단순한 통신 장비가 아닌 하나의 플랫폼으로 자리를 잡았다. 그리고 스마트폰 하드웨어의 발전으로 인해 모바일 플랫폼에 맞는 고성능 게임들도 개발되고 있으며, 점점 고품질화 되는 모바일 게임 시장에서 사용자들의 눈높이도 점점 높아져 가는 추세이다. 본 논문에서는 이러한 환경에 맞춰나갈 수 있도록 유니티3D 게임 엔진을 이용하여 최적화 방법과 세이더 등 다루기 어려운 기술들을 쉽고 빠르게 구현하는 것에 대하여 설명한다. 또한, 구름 생성 및 G-센서 동작 등 게임의 구현에 있어 주요 코드에 대하여 알아본다.

Ⅱ. 게임 구성

본 논문은 유니티 엔진을 이용하여 고수준의 방식으로 게임을 개발함으로써 기술적인 부분을 DirectX와 같은 저수준 방식[1]에 비해 상대적으로 쉽고 빠르게 구현할 수 있다.

유니티 엔진에서 게임은 하나의 3차원 공간을 의미하는 장면(Scene)들로 구성된다. 하나의 장면은 여러 가지의 게임 오브젝트(Game Object)로 구성되어 있으며, 하나의 게임 오브젝트는 여러 개의 컴포넌트(Component)들로 구성된다. 일반적으로 게임은 많은 장면으로 구성되지만, 장면이 지나치게 많으면 장면 간의 이동 시간이 길어 모바일 환경[2]에서는 부적합하다. 그렇기 때문에 본 논문에서 소개하는 게임은 최소한의 장면으로 구성된다. 총 4개의 장면으로 메인, 스테이지, 상점, 게임 플레이로 구성된다.

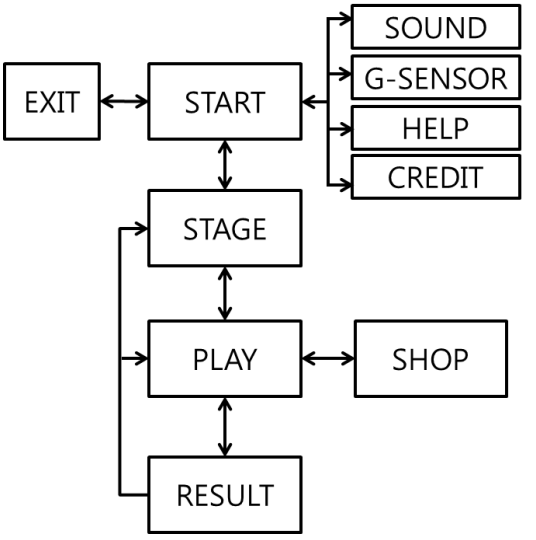

그림 1. 흐름도
Fig 1. Flowchart

게임의 전반적인 흐름은 다음의 <그림 1>과 같이 메인화면에서 시작하여 스테이지를 선택한 뒤, 게임을 진행하게 된다. 게임 진행 도중에 메뉴화면과 사운드 설정이 가능하고, 게임 결과에 따라서 메뉴 이동 가능 여부를 알 수 있다.

Ⅲ. 유니티 3D 게임 엔진

본 절에서는 에서는 유니티 3D 엔진을 이용하여 안드로이드 기반의 캐주얼 게임을 제작하고, 구현된 게임의 최적화 방법과 세이더에 대하여 설명한다.

3.1 유니티 엔진의 최적화 방법

<그림 2>는 유니티[3]에서 하나의 장면을 그릴 때, CPU와 GPU의 프로세스를 나타내는 그림이다. 최적화는 위와 같은 프로세스를 진행하는 과정에서 발생하는 병목현상을 파악하고 제거하는 작업이라고 할

수 있다. 유니티3D는 이를 쉽게 파악할 수 있도록 GUI 기반의 프로파일러를 제공하고 있다.

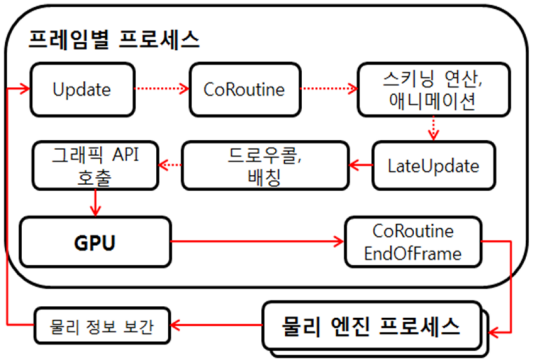

그림 2. 유니티의 렌더링 프로세스
Fig 2. Rendering Process of Unity 3D

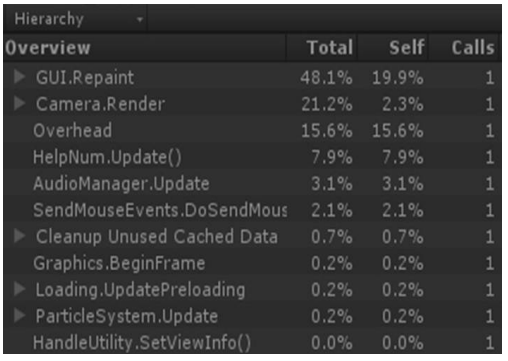

그림 3. 프로파일러 뷰
Fig 3. Profiler View

<그림 3>은 프로파일러 뷰[4]로 각각 CPU, GPU, 렌더링, 메모리, 오디오, 물리 영역으로 총 6개의 영역을 관리 할 수 있으며, 해당 영역의 타임라인에 분석된 정보가 표시되어 문제를 해결하는데 참고할 수 있다. 예를 들어, 캐릭터가 아이템을 먹는 순간 약간의 끊김 현상이 발생한다면, Overview에 아이템의 Update함수 내에서 처리 속도가 늦어진다는 것을 알 수 있다.

그 뒤에 Update함수 내에서 문제점을 찾아 처리할 수 있다. 코드 최적화에 대해서는 <그림 4>를 예로 들 수 있다.

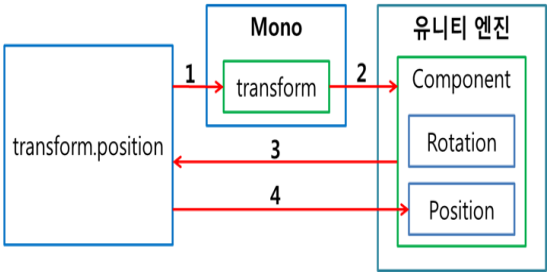

그림 4. 함수 처리 방식
Fig 4. Function processing

<그림 4>는 소스 코드의 함수 처리 방식[4]으로 그림과 같이 오브젝트의 위치, 회전, 크기 값을 저장하고 조작하기 위한 transform에서 위치를 변환하는 position을 사용하면 해당 오브젝트를 이동시킬 수 있다. 이 코드를 Update함수에 넣었을 경우 1~4의 과정을 프레임마다 반복해야 하므로 처리속도가 느려지게 된다.

하지만, <그림 5>의 코드처럼 <그림 4>의 3번 결과 값을 로컬 변수에 저장하고 이를 재활용 하게 되면, <그림 4>의 1~2의 과정은 생략된다. 이러한 방식으로 불필요한 연산을 크게 줄일 수 있다.

```
Transform refTransform;
Start () {
    refTransform = transform;
}
Update () {
    refTransform.position = Vector3.zero;
}
```

그림 5. 최적화의 예
Fig 5. Example of optimization

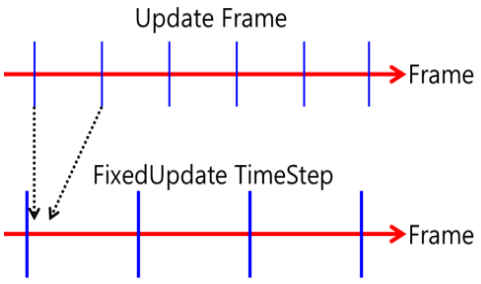

그림 6. 프레임간의 부조화
Fig 6. Mismatch between two frames

<그림 6>을 보면 렌더링 프레임[6]과 물리 프레임이 서로 초당 프레임 수가 다른 것을 볼 수 있다. 이 현상으로 인해 프레임의 지연 현상이 발생하는 경우가 있다. 주로 카메라가 물리 효과가 적용된 오브젝트를 따라갈 때 발생하는데, 기존에 있던 Update 함수를 사용할 경우에는 초당 프레임 수가 PC 성능이나 환경 등에 따라 유동적으로 변하게 된다. 이와 같은 현상은 초당 프레임 수를 고정시키면 쉽게 해결할 수 있다. Update 함수의 이름을 FixedUpdate로 변경한 뒤, 초당 프레임 수를 고정시키면 위와 같은 프레임간의 부조화를 없앨 수 있다.

3.2 셰이더

유니티는 사용자의 편의를 위하여 아래의 <그림 7>과 같이 다양한 효과의 셰이더 시스템을 제공한다. 고사양의 데스크탑이나 콘솔기기 뿐만 아니라 스마트폰 앱 제작에 최적화된 가벼운 모바일 셰이더[7]까지 100여 가지의 셰이더를 제공해주고 있다.

본 논문에서는 모바일용 내장 셰이더를 사용하여 간단하게 구현하였지만, 셰이더랩이라는 간편한 문법을 통해 멀티 플랫폼에서 동작하는 셰이더를 쉽게 제작할 수 있고, 정점/픽셀 셰이더 코드를 Cg와 GLSL로 직접 제작할 수 있도록 인터페이스가 제공되어 고품질 셰이더[8]를 제작할 수 있다.

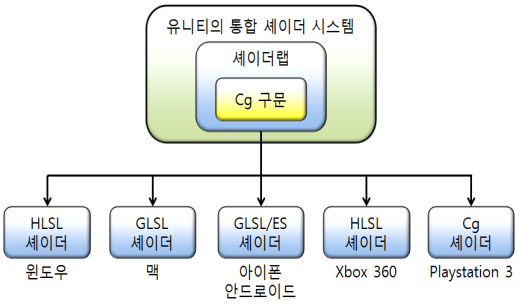

그림 7. 통합 셰이더 시스템
Fig 7. Integrated shader system

Ⅳ. 게임의 구현

본 절에서는 게임에 사용된 주요 소스 코드에 대한 내용으로 구성된다. 본 논문에서 제작한 'Flying Chick' 이라는 게임은 안드로이드 기반의 캐주얼 아케이드 장르로 주인공 캐릭터가 구름에서 떨어지면서 독수리의 공격을 피하고, 구슬을 모아가는 단순한 형태의 게임이다. 여기서 가장 핵심적인 요소인 구름 생성 방식과 G-센서를 이용한 캐릭터 이동, 상점 시스템의 구현에 대하여 설명한다.

4.1 구름 생성

<그림 8>은 구름 생성[9] 시 초기 값으로 설정된 슈도코드로, 구름간의 높이를 설정하고 구름 번호에 따라 구름과 구름 사이에 간격을 두어 구름과 구름이 일정한 간격으로 생성되도록 하였다.

CloudTrans[i].position은 구름의 생성 위치로 높이는 Cloud[i].Start가 되고, xz축은 일정 반경 이내에 무작위로 생성된다.

```
for ( 구름 개수 )
{
    Cloud[i] = 각 구름의 컴포넌트;
    Cloud[i].Start = 구름 사이 거리 * 구름 개수;
    CloudTrans[i] = 각 구름의 transform클래스;
    CloudTrans[i].position = 구름의 생성 위치;
    구름 번호 증가;
}
```

그림 8. 구름 생성 슈도코드
Fig 8. Pseudocode for Cloud creation

<그림 9>는 게임이 시작할 때의 화면이다. 캐릭터의 아래 방향에 구름들이 일정한 높이 간격으로 생성되는 것을 볼 수 있다. 게임이 시작되면, 캐릭터를 제외한 모든 오브젝트가 위쪽으로 이동하여 사용자 시점에서는 캐릭터가 구름을 밟고 내려가는 것처럼 보이게 된다.

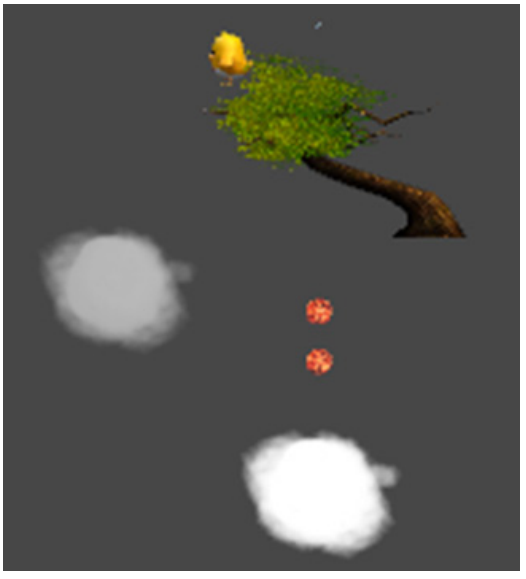

그림 9. 구름 생성 결과
Fig 9. Result of cloud creation

4.2 G-센서를 이용한 조작

<그림 10>은 G-센서(Gyro Sensor)[10]를 이용한 슈도코드이다. G-센서의 상태에 따라서 센서의 기능을 켜고 끌 수 있도록 설계하였다. 이 부분은 크게 독수리의 이동과 캐릭터의 이동 및 회전으로 구성되어있다. 기기의 G-센서 값을 가져오는 Input.acceleration을 사용하여 값에 따라 해당 독수리 오브젝트를 움직인다. 캐릭터도 동일한 방식으로 구현하며, 방향을 바라보도록 설정한다.

```
if ( GSensor 실행 )
{
    EagleAccel.x = Input.acceleration.y;
    EagleAccel.y = -Input.acceleration.x;
    if ( 타겟 오브젝트 )
        EagleAccel * Speed만큼 이동;

    ChickAccel.x = Input.acceleration.y;
    ChickAccel.z = -Input.acceleration.x;
    if ( 게임 진행 중 )
        캐릭터 회전 =
        Quaternion.LookRotation(-ChickAccel);
}
```

그림 10. G-센서 슈도코드
Fig 10. Pseudocode for G-Sensor

<그림 11>은 G-센서를 이용하여 캐릭터가 이동하고 회전하는 모습을 보여준다. 그림에서 보는 바와 같이 모바일 기기를 지면과 평행한 상태를 기준으로 상하좌우로 기울이면 캐릭터가 해당 방향을 바라보면서 이동하는 것을 볼 수 있다.

4.3 상점 시스템

<그림 12>는 아이템[11]을 구매하고 기록하는 슈도코드이다. 아이템이 선택되면 코인에서 아이템 가격을 뺀 뒤, 레지스트리에 값을 저장하는 PlayerPrefs 함수를 사용하여 코인과 아이템의 개수를 저장한다.

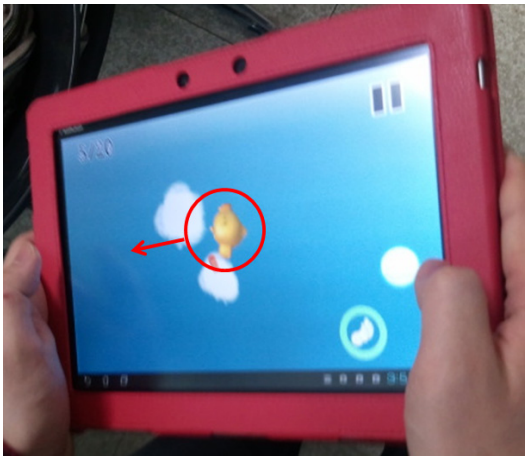

그림 11. 캐릭터 이동에 G-센서 적용
Fig 11. Apply the G-sensor to move the game character

```
if ( 아이템 선택 )
{
    if ( 코인이 아이템 가격보다 적음 )
        코인 부족 메시지 출력;
    else if ( 아이템 구매 가능 )
    {
        코인 -= 아이템 가격;
        레지스트리에 코인 저장;
        아이템 개수 증가;
        레지스트리에 아이템 개수 저장;
    }else
        보유 아이템 초과 메시지 출력;
}
```

그림 12. 아이템 구매 슈도코드
Fig 12. Pseudocode for Item purchase

<그림 13>은 상점 메뉴의 구현 화면으로, 사용자가 게임에서 얻은 코인으로 각종 아이템을 구매할 수 있다. 화면 우측에는 보유중인 아이템이 표시되며, 원하는 아이템을 선택하고 상점에서 나가면 게임 도중에 사용할 수 있도록 구현한다.

그림 13. 상점 메뉴
Fig 13. Menu

V. 실행 결과

본 게임은 안드로이드 4.0 버전과 nVidia 테그라3 칩셋을 사용하고 있는 아수스 트랜스포머 패드 TF300에서 테스트 하였으며 , 윈도우 7 환경에서 유니티 3D 3.5.5버전에서 자바스크립트와 C#을 이용하였으며, 어도비 포토샵 CS3, 오토데스크 3D맥스 2010를 이용하여 그래픽 리소스를 제작했다.
<그림 14>는 게임 메인화면을 나타낸다. 화면 중앙에는 게임시작, G-센서, 사운드 설정, 도움말이 있고, 우측 하단에는 크레딧, 좌측 하단에는 게임 종료 버튼으로 구성된다.

그림 14. 게임 메인화면
Fig 14. Main Screen

그림 15. 스테이지 메뉴
Fig 15. Stage Menu

메인화면에서 게임시작 버튼을 누르게 되면 <그림 15>와 같이 스테이지 메뉴를 보여준다. 화면 좌측에 있는 큐브를 터치 드래그하여 스테이지를 선택할 수 있으며, 우측에는 각 스테이지들의 최고기록을 보여준다. 화면 하단을 보면 좌측부터 뒤로 가기, 상점, 플레이 버튼으로 구성된다.

그림 16. 게임 플레이
Fig 16. Game Play Screen

<그림 16>은 게임 플레이 화면으로, 인터페이스는 왼쪽 상단부터 시계방향으로 목표 구슬 개수, 일시 정지, 상점아이템, 날기, 방향키로 구성된다.

그림 17. 게임 플레이
Fig 17. Game play Screen

<그림 17>은 독수리 타깃, 경고메시지, 바람 가르는 효과 등의 이펙트와 관련된 화면으로, 2D 이미지를 변환하여 효과를 나타낸다.

그림 18. 게임 결과
Fig 18. Result Screen

<그림 18>은 게임 결과 화면으로, 플레이 시간으로 랭크가 결정된다. 또한, 구슬 획득 개수와 랭크 점수, 보너스 점수들로 총 점수가 계산되어 총 점수에서 일정량의 코인을 얻을 수 있다. 그리고 게임 결과 상태에 따라서 다음 스테이지의 플레이 가능 여부를 알 수 있다.

게임 플레이는 안드로이드 2.1버전이상부터 호환 가능하며, 1280*800(16:10)의 해상도를 가지고 있다. 본 논문에서 구현한 게임은 테스트에 사용한 기기에 최적화되어 있지만, 모든 인터페이스가 화면 비율로 맞

춰져 있어 사실상 거의 모든 안드로이드 기반 휴대폰에서 구동이 가능하다.

Ⅵ. 결론

본 논문은 빠르게 성장하는 모바일 게임 시장의 변화에 맞춰갈 수 있도록 빠르게 개발할 수 있는 유니티 3D 게임 엔진을 이용한 게임 개발에 대하여 설명하였다. 저 수준의 게임 개발 방식은 우선 프로그래밍 언어를 이해해야 하기 때문에 사전 준비를 요구한다. 하지만 유니티 3D 게임 엔진은 이러한 복잡한 부분을 상대적으로 구현하기 쉽게 구성되어 있어서 개발 기간을 크게 단축시킬 수 있다.

주요 사용 연령층인 10~20대 여성을 대상으로 한 귀여운 컨셉의 케릭터 사용 했고, 알아보기 쉬운 인터페이스와 간편한 조작을 기본으로 하였다. 이러한 게임의 컨셉을 기반으로 완성도를 좀 더 높여 상용화 수준으로 끌어들일 후속 연구를 진행할 계획이다.

참고문헌

[1] Soo-Kyun Kim. *Design and Development of 3D Online Arcade Tank Game*, Jorunal of KKITS, Vol. 6, No. 1, pp. 45~51, 2011.

[2] Park Dalgyeong. *Unity Bible,* Rainbow Ground, 2011.

[3] Lee Dustin. *The Principle of Unity Game Development,* Acorn, 2012.

[4] Wes Mcdermott. *Creating 3D game art for the iPhone with unity,* Acorn, 2011.

[5] Menard. *Game Development with Unity*, Course Technology, 2011.

[6] Craig Stevenson. *Unity 3 Blueprint,* Acorn, 2011.

[7] Jate Wittayabundit. *Unity 3 Game Development*, Acorn, 2012.

[8] Jate Wittayabundit. *Unity 3 Game Development Hotshot*, Acorn, 2012.

[9] Sue Blackman. *Beginning 3D Game Development with Unity*, Apress, 2011.

[10] Takahashi Keiziro. *Unity Introduction*, Acorn, 2012.

[11] Creighton. *Unity 3D Game Development by Example*, Acorn, 2011.

저자소개

김수균(SooKyun Kim)

2006년 고려대학교 컴퓨터학과(이학박사)
2006년~2008년 삼성전자 통신연구소 책임 연구원
2008년~현재 배재대학교 게임공학과 교수

안성옥(Sung-Og An)

1983 고려대학교 수학교육과(이학사)
1985 고려대학교 컴퓨터학과(이학석사)
1989 고려대학교 컴퓨터학과(이학박사)
1991년~현재 배재대학교 게임공학과 교수

강지훈(Ji-Hoon Kang)

현재 배재대학교 게임공학과 석사과정

송기섭(Gi-seob Song)

현재 배재대학교 게임공학과 학부과정

이희범(Hee-bum Lee)

현재 배재대학교 게임공학과 학부과정