Journal of The Korea Society of Computer and Information
Vol. 23 No. 8, pp. 25-30, August 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.08.025

Design of Runner Game using Overlap Circle in Unity3D

Hyun-Jun Kim*, Soo Kyun Kim*

Abstract

In this paper, we design a 2D one touch runner game which controls on smart device easily using a Unity3D game engine. Unity is the creator of the world's widely-used 2D and the 3D cross-platform game engine developed by Unity Technologies. Also, Unity builds high-quality 2D and 3D games on a various hardware device. This paper shows the familiar character as a cartoon and feels to enjoy using platformer with various game users. we show that the proposed platformer method improve controlling game path on the smart device.

▸Keyword: Sprite, Collision, Unity3D, 2D Runner Game, Platform

I. Introduction

2013년부터 '쿠키 런'[1] 및 '윈드 러너'[2] 같은 게임들이 다운로드 수에서 높은 순위권을 차지하면서 높은 매출을 기록하였다. 특히 2013년 러닝게임의 인기는 매우 컸고, 그 중 '쿠키 런'의 2013년 4분기 실적은 영업이익 82억 원, 매출 228억으로 나타났다. 많은 사용자들은 러너 게임을 즐기고 있으며, 그러한 사용자들을 목표로 하여 가볍고 중독성 있는 게임을 설계하는데 목적을 두고 있다.

플랫폼 게임[3]은 플레이하는 게임 속에 플랫폼을 설치하여 캐릭터가 플랫폼을 밟아가며 진행하는 게임을 말한다. 플랫폼이 등장하는 모바일 게임으로는 '쿠키 런', '지오메트리 대쉬', '윈드 러너' 등이 있으며, 플레이어는 플랫폼과 플랫폼을 쉴 새 없이 옮겨 다니며 장애물을 피하고 게임을 진행하게 된다. 장애물을 피하기 위해 필수적인 요소가 '점프'이다. 캐릭터가 점프에 실패하여 떨어지거나 장애물과 충돌하여 체력이 줄어들게 되어 캐릭터가 죽을 수도 있기 때문에 이러한 플랫폼 게임의 점프는 차지하는 비중은 비교적 높은 편이다. 점프는 플랫폼을 옮겨 다니거나 장애물을 피하는 것으로 플랫폼 게임의 핵심 요소라고 할 수 있다.

본 논문은 모바일 플래포머를 이용한 중독성 있는 2D 러너 게임 개발을 목표로 하며, 유니티 3D 엔진[4,5,6]을 사용하여 게임을 설계하는 방법에 대해 소개한다. 특히 캐릭터 디자인부터 게임 스테이지 구성 및 게임의 재미를 더해주는 게임 구조물 등에 대하여 구체적으로 설명한다.

II. System Architecture

2.1. System Architecture

본 절은 게임 디자인에 대해 설명한다. Fig. 1은 게임의 구성도를 나타내며, 게임은 메인화면에서 스테이지 선택, 옵션, 도움말 및 종료 메뉴를 선택할 수 있다. 스테이지 화면에서는 각 스테이지로 이동할 수 있도록 되어 있고, 총 5가지로 구성되어 있으며, 튜토리얼 스테이지 1개, 본 스테이지 4개로 구성되어 있다.

• First Author: Hyun-Jun Kim, Corresponding Author: Soo Kyun Kim
*Hyun-Jun Kim (nicesk@daum.net), Dept. of Game Engineering, Pai Chai University
*Soo Kyun Kim(kimsk@pcu.ac.kr), Dept. of Game Engineering, Pai Chai University
• Received: 2018. 07. 26, Revised: 2018. 08. 15, Accepted: 2018. 08. 17.
• This work was supported by the research grant of Pai Chai University in 2018.

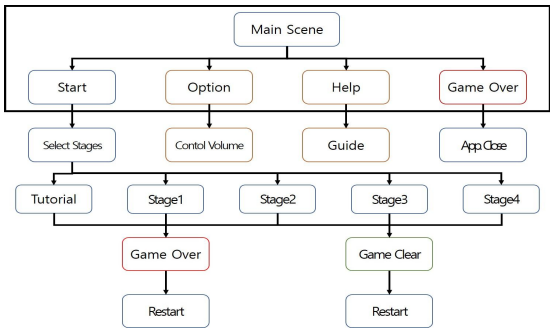

Fig. 1. System Architecture

Fig. 2는 스테이지1의 게임 구성도 보여주며, 게임 시작 할 때 3가지의 상황(게임 오버, 게임 클리어, 일시 정지)이 발생할 수 있고, 각각의 상황에 따라 게임종료 시 캐릭터 사망 애니메이션이 수행되며 다시 게임이 시작된다. 게임 클리어 시, 점수 화면이 나타나며 재시작 버튼과 메뉴 버튼이 생성된다. 게임을 다시 하는 기능과 메뉴 화면으로 이동하는 기능을 설정하고, 일시정지 버튼을 클릭하였을 때, 일시정지 메뉴가 나타나게 되고 계속하기, 다시시작, 메뉴화면, 게임종료의 과정을 수행하게 된다. 게임종료 버튼의 경우 어플리케이션을 완전히 종료시키게 한다.

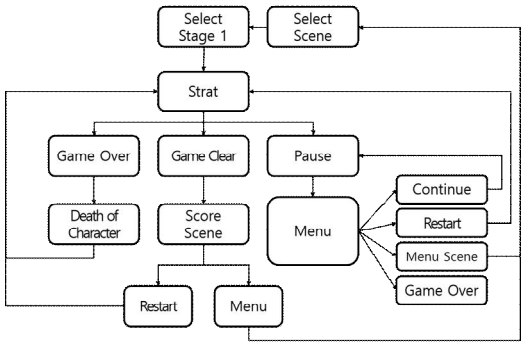

Fig. 2. Stage 1

스테이지 화면에서 스테이지 2, 3, 4는 잠겨 있도록 하였고, 스테이지1에서 코인과 클리어를 한 조건을 충족하였을 때 다음 스테이지로 이동하게 된다. 스테이지 1의 특징은 카메라의 회전이며, 모바일 플랫포머를 타깃으로 한 게임의 특성상 스마트폰을 회전 시킬 수 있다는 특징이 있기 때문에 게임이 회전함에 따라 스마트폰을 회전시키며 게임을 즐길 수 있도록 설정한다.

2.2. 2D Sprite

게임 캐릭터는 애니메이션을 구성하기 위한 필수 요소이며, 여기서는 스프라이트[7,8]을 사용한다. 2D 이미지들을 연속적으로 컴퓨터 화면에 보일 때 애니메이션 되며, 이렇게 애니메이션이 될 객체를 스프라이트[9]라고 한다. Fig. 3은 캐릭터의 애니메이션을 구성 할 스프라이트 이미지이다.

Fig. 3. Image using Photoshop

Fig. 3은 구글에서 다운로드 받은 이미지이며, 이미지를 가져와 포토샵CS6 프로그램을 이용하여 배경색을 투명하게 한 후 이미지를 일정한 크기로 분할하여 수정한다. Fig. 3은 분할되어 있는 스프라이트 이미지이며, 총 10장의 이미지로 캐릭터의 달리기 애니메이션을 생성한다.

2.3. Generation of Platfomer

플랫포머[10]는 발판이란 뜻으로 언덕, 층계 등을 포함하며, 플랫포머가 적용된 게임은 점프라는 구조물을 필수적으로 사용하게 된다.

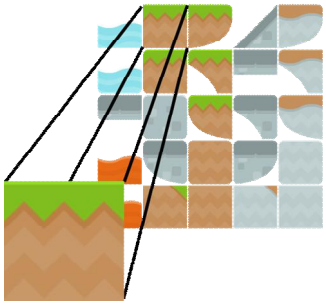

Fig. 4. Platformer Image

플랫포머는 게임 캐릭터 이미지와 다르게 움직임이 없기 때문에 스프라이트로 사용하지 않고, 2D 이미지로 사용한다. Fig. 4는 게임에 사용된 플랫포머 이미지이며 게임 설계 시 확대된 이미지를 이용하여 플랫포머를 배치하고, 캐릭터가 이미지를 발판으로 삼아 그 위에서 달릴 수 있도록 한다.

III. Game Design

본 절에서는 캐릭터를 비추는 카메라가 캐릭터를 부드럽게 따라가는 방법과 캐릭터를 비출 수 있는 범위에 대해 설명하고, 캐릭터 애니메이션, 오브젝트와의 충돌과 파괴, 플랫포머의 배치와 UI설계 등에 대해 설명한다.

3.1. Camera Movement and Bounce

조작키 없이 캐릭터는 일정한 속도로 앞으로 달려가고, 카메라는 캐릭터를 부드럽게 따라가도록 설정한다.

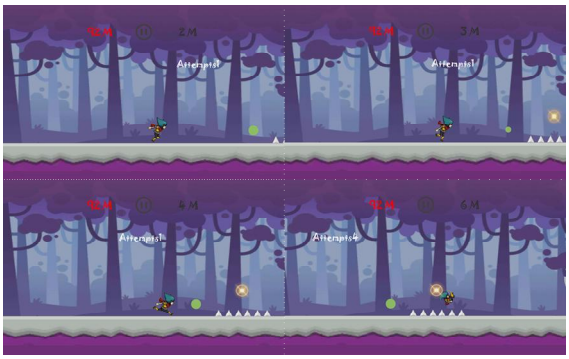

Fig. 5. Camera Movement

카메라가 캐릭터를 따라갈 때 이동 범위를 제한하여 맵 밖으로 너무 나가지 않도록 설정하는 것은 캐릭터를 따라갈 때의 움직임이 너무 단조로워 지루하지 않게 하기 위함이다. Fig. 6의 안쪽 선은 카메라 바운스[11]의 범위이며 카메라는 이 선의 바깥으로는 나갈 수 없도록 설정한다.

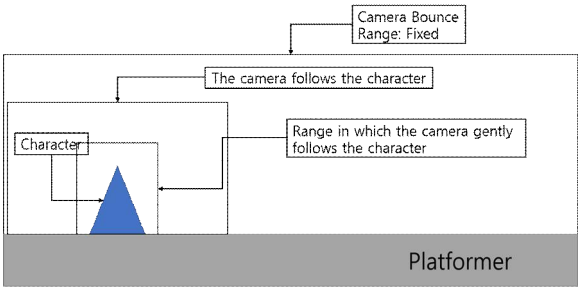

Fig. 6. Camera Bounce

3.2. Character Animation

본 절에서는 2D 게임을 제작하기 위하여 스프라이트를 이용하여 애니메이션을 수행하는 방법에 대해 설명한다. 3D 게임에서 캐릭터의 움직임을 표현하기 위하여 뼈대를 주축으로 애니메이션[12-14]을 수행한다. 그러나 러너 게임에서는 캐릭터의 달리기와 점프 애니메이션이 자연스럽고 부드럽게 연결 되어야하기 때문에 연속적인 스프라이트[15] 사용이 필요하다. Fig. 7은 애니메이션을 수행하기 위하여 스프라이트를 삽입한 결과이다.

Fig. 7. Sprite Animation

3.3. Collision Detection

게임에서 캐릭터가 적에게 공격을 받거나 장애물에 충돌하여 사망하고, 캐릭터가 사라지는 방식을 사용한다. Unity3D 엔진에서는 충돌에 자주 사용되는 함수가 있으며, 특히 OnCollision와 OnTrigger를 사용한다. OnCollision와 OnTrigger의 차이는 캐릭터가 충돌체를 통과할 수 있는가 이며 전자는 통과할 수 없지만 후자는 통과할 수 있다. 본 게임에서의 캐릭터와 장애물 간의 충돌은 OnCollsionEnter 함수를 사용하였고, 장애물의 Tag를 Trap으로 설정하여 BoxCollider2D 컴포넌트가 적용되어 있는 캐릭터 오브젝트가 Trap으로 Tag되어있는 오브젝트에 충돌하게 되면, 사망 애니메이션으로 전환되는 방식을 사용한다.

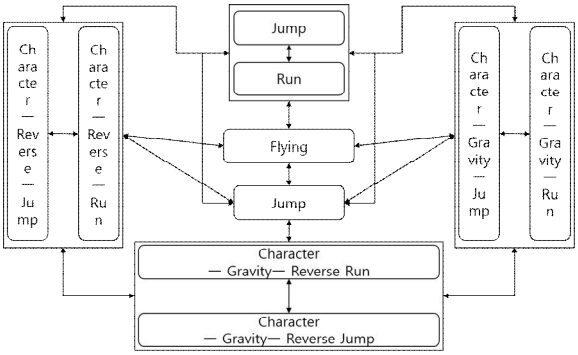

Fig. 8. Character Animation

Fig. 8은 캐릭터 애니메이션 내의 애니메이터를 간략하게 나타낸 도표이다. 애니메이터에서는 오브젝트의 애니메이션 사이의 수행을 파라미터를 통해 수행하도록 설정할 수 있다. Fig. 8의 캐릭터 달리기 항목에 해당되며, 캐릭터 점프, 역중력, 반대 진행 방향의 애니메이션을 생성한 후 애니메이터에 적용 시키면 캐릭터는 충돌 또는 사망 했을 때의 애니메이션을 수행할 수 있다.

```
void OnCollisionEnter2D
(Collision2D col) {
  if("Trap" 오브젝트에 충돌하였을

 때){
    진동발생 ();
    재시작 코루틴 함수 실행 ();
  }
}
```

Fig. 9. Collision Detection Pseudocode

Fig. 9의 OnCollsionEnter2D 함수는 2D 물리에만 적용되는데 오브젝트의 자식 충돌체가 다른 충돌체에 충돌했을 때 호출된다. 관련된 해당 오브젝트에 대한 추가 정보는 호출 동안에 전달된 Collision2D 파리미터를 통해 보고된다. 이 정보가 필요없다면, 해당 파라미터 없이 OnCollisionEnter2D를 선언할 수 있다. 장애물 오브젝트에는 PolygonCollision2D 컴포넌트를 적용하여 세밀한 충돌을 수행하도록 설정하였고, 캐릭터가 "Trap"으로 태그 되어 있는 오브젝트에 충돌 하였을 때, 재시작 코루틴이 실행되어 캐릭터는 처음 위치로 돌아가게 되며 바로 게임이 재시작 된다.

```
public IEnumerator RestartGameCo(){
  스코어 증가 = false;
  배경음 정지 ();
  캐릭터 애니메이션 설정 (“Die”, true);
  yield return new WaitForSeconds
  (0.5f);

  카메라 위치 = 처음 카메라 위치;
  캐릭터 위치 = 처음 캐릭터 위치;
  캐릭터 애니메이션 설정 (“Die”, false)
  배경음 재생 ();
  스코어 = 0;
  스코어 증가 = true;
}
```

Fig. 10. Restart Pseudocode

Fig. 10의 스크립트는 캐릭터 오브젝트가 장애물 오브젝트와 충돌하였을 때, 발생하는 코루틴 함수에 대한 내용이며, 장애물 오브젝트와 충돌하는 즉시 스코어 증가와 배경음을 멈추고, 캐릭터에게는 사망 애니메이션을 수행하도록 설정한다. 이는 0.5초 동안 수행되며, 0.5초가 지난 후에는 카메라 위치와 캐릭터 위치를 처음 위치로 이동시키며, 캐릭터 사망애니메이션을 멈추고, 배경음과 스코어 증가를 시작하도록 한다. 또한 스코어는 다시 0으로 초기화시켜 다시 스코어를 획득할 수 있도록 설정한다.

3.4. Platformer

본 절에서는 플랫포머의 배치와 캐릭터의 플랫포머 인식을 구체적으로 설명한다.

Fig. 11. Stage Platformer

Fig. 11은 스테이지4의 플랫포머 배치도이다. 플랫포머는 플레이어의 진행 경로를 설정해주고 플랫포머 이외의 길은 숨겨진 경로 또는 사망으로 이어지는 방식으로 설정한다.

```
// bool값 grounded의 유무 상태 확인
public bool grounded;
// Layer의 이름
public LayerMask whatIsGround;
// 플랫포머의 범위를 체크할 위치
public Transform groundCheck;
// 범위를 체크할 지름
public float groundCheckRadius;

void Update () {
  grounded =
  Physics2D.OverlapCircle(groundCheck.position,
groundCheckRadius, whatIsGround);
}
```

Fig. 12. Pseudocode of Character's Location

Fig. 12는 캐릭터가 플랫포머를 더욱 섬세하고 세밀하게 인식을 하도록 캐릭터 달리기 스크립트에 Physics2D. OverlapCircle 함수를 사용한다. OverlapCircle 함수는 충돌체를 체크할 위치에서 지름 값을 설정하여 지름 크기만큼의 원형의 영역 내의 레이어 이름이 있는지 확인하는 함수이다.

3.5. U.I.

구조물은 플레이어가 게임을 클리어하기 위해 도움을 주거나 방해하기 위해 맵 곳곳에 설치해 놓았다. 각 구조물들은 스프라이트 이미지를 사용하여 동적인 애니메이션을 수행하도록 설정해 놓았다.

```
void OnTriggerStay2D(Collider2D
other){
  if(“DJump” 오브젝트에 충돌하였을 때)
  {
    if(마우스 왼쪽 버튼을 클릭했을 때){
      리지드바디의 높이 =
      new Vector2 (myRigidbody.velocity.x ,
      jumpForce);
    }
    grounded = false;
  }
}
```

Fig. 13. Double Jump Pseudocode

더블 점프 오브젝트(Fig. 13)는 플레이어의 점프를 한 번 더 가능하게 해주는 구조물이다. 더블 점프에 사용된 OnTriggerStay2D 함수는 오브젝트의 콜라이더와 다른 오브젝트의 콜라이더가 충돌하고 있는 동안 매 프레임 계속 호출한다. 충돌 관련 오브젝트에 대한 자세한 정보는 호출시 전달되는 Collision2D 인수로 보고된다. 더블 점프 오브젝트는 캐릭터와 충돌하는 동안 계속 수행되며, 충돌하고 있는 동안에 타이밍에 맞춰 터치하면 수행되도록 설정되어 있다.

하이점프 오브젝트는 플레이어의 의지와 상관없이 충돌하였을 때 강제적으로 캐릭터 점프력의 1.5배로 캐릭터를 점프시킨다. 더블점프 및 하이점프 오브젝트 외에도 카메라 회전 오브젝트(카메라를 90도 회전시킴), 상하좌우 반전 오브젝트 등 다양한 구조물을 삽입하여 게임에 재미를 더했다. 게임 UI[16]는 게임 진행에 있어서 플레이 진행상황을 보여주고, 버튼을 사용하여 일시정지 메뉴를 불러오는 등 게임 진행을 도와준다.

```
if(스코어가 증가하는가?) {
  스코어 += 초당 증가 스코어 * Time.deltaTime;
}
if(스코어 > 최고 스코어 일 때){
  최고스코어 점수 = 스코어 점수;
  PlayerPrefs.SetFloat(“S2HighScore”,
  최고 스코어 점수);
}
스코어.text = Mathf.Round (스코어) + “ M”;
최고 스코어.text = Mathf.Round (최고 스코어) + “
M”;
```

Fig. 14. Score Pseudocode

Fig. 14에서와 같이 스코어는 초당 포인트를 증가시키도록

설정하였고, 증가하는 스코어를 텍스트 UI를 이용하여 화면에 표시한다. 증가한 스코어의 최고 스코어는 최고점수 텍스트 UI에 저장하여 고정 시켰다.

```
void OnTriggerEnter2D(Collider2D
_other){
  if("S3Coin1" object에 충돌했을 때){
    오브젝트 활성화 (false);
    PlayerPrefs.SetInt("S3Coin1", 1);
  }
  if("S3Coin2" object에 충돌했을 때){
    오브젝트 활성화 (false);
    PlayerPrefs.SetInt("S3Coin2", 1);
  }
  if("S3Coin3" object에 충돌했을 때){
    오브젝트 활성화 (false);
    PlayerPrefs.SetInt("S3Coin3", 1);
  }
}
```

Fig. 15. Coin Pseudocode

Fig. 15는 캐릭터가 코인 오브젝트에 충돌하게 되면 코인 오브젝트와 충돌함과 동시에 PlayerPrefs에 값을 저장하여 같은 이름의 변수를 스테이지 선택화면의 코인 이미지에 적용 시킨 후 얻게 된 값이 1이 되었을 때 코인 이미지가 활성화 되도록 표시한다.

IV. Experimental Results

게임은 다음과 같은 사양에서 개발되었다. 윈도우7 64bit, AMD FX(tm)(8300)Core Processor 3.30GHz 환경에서 개발한다. 특히 이미지 제작은 어도비 포토샵을 이용하였고, 프로그램 코드는 C#을 이용하였으며, 유니티 엔진 4.8.6버전으로 개발하였다.

Fig. 16. Game Play Scene

Fig. 16은 리버스 오브젝트, 비행 조작 및 게임 구조물을 이용하여 게임 플레이를 하고 있는 장면이다.

Fig. 17은 모바일을 이용하여 실제 게임을 실행하여 스테이지를 진행하는 상황이다. 스테이지는 이름과 배경음은 게임 스토리에 연관 있도록 설정한다. 본 게임은 PC로 개발을 하여 테스트 하였고, 모바일에서도 정상적으로 작동하는 것을 확인 하였다.

Fig. 17. Execution on a Mobile

V. Conclusions

본 논문은 남녀노소 모두가 쉽게 즐길 수 있는 2D 원터치 러너 게임을 유니티3D 엔진을 이용하여 모바일에서 즐길 수 있도록 설계하였다. 특히 본 게임은 중독성 있는 게임 방식과 배경음에 맞춘 플랫포머의 배치가 큰 특징이다. 스테이지를 잠그고 선택하는 방식과 사망하면 처음으로 돌아가는 방식을 사용하여 플레이어에게 클리어 하겠다는 도전의식을 발휘하게 하고, 중독성을 이끌어 낼 수 있도록 게임의 완성도를 높인 것이 특징이다.

REFERENCES

[1] Hangyuong GameTok Newshttp://gametoc.hankyung.com/news/articleView.html?idxno=15322, 2013
[2] Windrunner: https://ko.wikipedia.org/wiki/%EC%9C%88%EB%93%9C%EB%9F%AC%EB%84%88
[3] Seo Dong Min, IT DongA http://it.donga.com/6838/
[4] Creighton, "Unity 3D Game Development by Example Beginner's Guide", 2010
[5] Charles Bernardoff , NGUI for Unity, PACKT, 2014
[6] Will Goldstone, "Unity 3.x Game Development Essentials", Packt Publishing; 2 edition, December 20, 2011
[7] Szijarto, G. and K. Jozsef. High Resolution Folaige Rendering for Real-time Applications. Budmerice, Slovak Republic: SCCG, 2003.
[8] ThinkEquity LLC(2011), "Multi-Channel Game-As-A-Service II: Ubiquitous Game In The Cloud".
[9] Sprite: http://www.thisisgame.com/webzine/series/nboard/212/?series=99&n=57612

[10] Platformgame: https://ko.wikipedia.org/wiki/%ED%94%8C%EB%9E%AB%ED%8F%BC_%EA%B2%8C%EC%9E%84
[11] 2D Foundations,http://2dfoundations.wikifoundry.com/page/Foreground,+Middleground,+and+Background
[12] Lander, Jeff, "skin them bones game programming for the web generation:, Game Developer Magazine pp. 11-16. May 1998.
[13] Carl Granberg, "Character Animation With Direct3D," CharlesRiverMedia, pp. 47-49, April 2009.
[14] Doug L, James and ChristopHer D, "Skinning mesh animations," ACM Transactions on Graphics (ACM SIGGRAPH 2005), pp. 399-407, August 2005.
[15] Mark DeLoura, "Game Programming Gems", Charles River Media, August 2000.
[16] Bahn Kyoungjin, Hyo Kim, Kyungwon Lee, Hyunhee Kim, "A Study on Effect on Flow of Customized User Interface in Game", Society of Design Convergence, pp. 1-12, May 2007.

Authors

Hyun-Jun Kim received B.S. degrees in Game Engineering from Paichai University.

Soo Kyun Kim received Ph.D. in Computer Science & Engineering Department of Korea University, Seoul, Korea, in 2006. He joined Telecommunication R&D center at Samsung Electronics Co., Ltd., from 2006 and 2008. He is now a professor at Department of Game Engineering at Paichai University, Korea. His research interests include multimedia, pattern recognition, image processing, mobile graphics, geometric modeling, and interactive computer graphics. He is a member of ACM, IEEE, IEEE CS, KACE, KMMS, KKITS and KIIT.