유니티 3D 엔진을 활용한 2.5D 어드벤처 게임 개발

이가영, 신병석
인하대학교 컴퓨터공학과
e-mail : elain202@gmail.com, bsshin@inha.ac.kr

Development of 2.5D Adventure Game using Unity3D™ Engine

Ka-young Lee, Byeong-seok Shin
Dept. of Computer Science, In-ha University

요 약

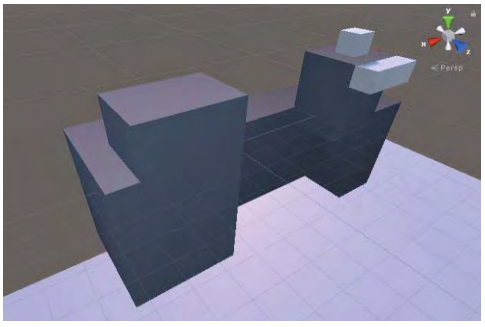
컴퓨터 게임에서 3D 가 아닌 2.5D 환경을 사용하면 사용자에게 신선한 상호작용적 즐거움을 줄 수 있다. 그러나 2.5D 환경의 게임을 만들려면 2D 와 3D 각각의 장점을 취하고 단점을 극복하기 위한 방법론이 필요하다. 본 논문에서는 유니티 3D 엔진을 이용하여 2.5D 어드벤처 게임을 개발하는 과정과 함께, 2.5D 의 단점을 극복하기 위해 적용한 그래픽 효과와 게임학(Game studies)적 요소를 제안한다.

1. 서론

2.5D 라는 용어는 쓰이는 분야에 따라 다양한 의미로 사용되지만, 게임 분야에서 2.5D 게임이란 가상카메라의 각도가 제한된 3 차원 게임을 의미한다[1]. 2.5D 게임을 만드는 개발자는 2D 와 3D 의 장점을 취함으로써 새로운 사용자 경험을 제공할 수 있으나, 2.5D 환경에서 게임에 대한 몰입(immersion)을 증진시키는 방법을 고안해야 한다.

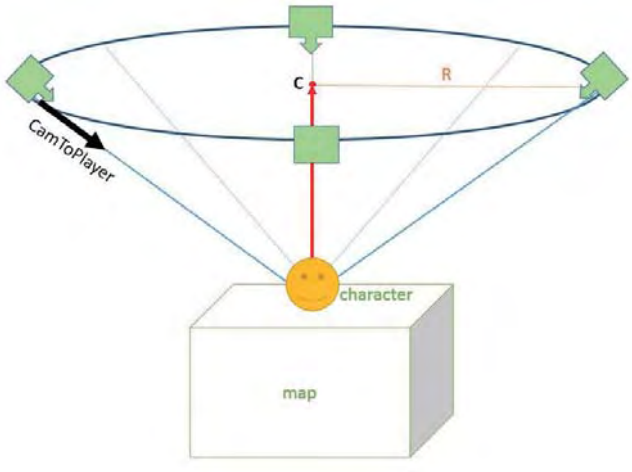
2. 유니티 3D 에서의 2.5D 게임 구현

여기에서는 (그림 1)과 같이 3D 큐브들을 이어 붙여 맵을 구성하였고, 이때 윗면, 아랫면을 제외한 큐브의 네 옆면은 각각 카메라에서 보이는 네 면이 된다. 이 면들에 직교투영(orthogonal projection)을 적용하여 2D 공간처럼 보이는 효과를 주었다.



(그림 1)큐브들로 이루어진 월드맵

3D 공간을 2D 공간처럼 보이도록 투영 시켰기 때문에, 실제로는 이동경로가 존재하지만 현재 시점에서는 보이지 않을 수 있다. 플레이어는 수시로 카메라 전환을 통해 맵의 네 면을 살펴보면서 이동할 경로를 정해야 한다.



(그림 2)카메라의 네 방향

카메라는 (그림 2)의 점 C 를 중심으로 하는 원의 둘레를 따라서만 움직인다. 점 C 는 캐릭터의 y 좌표에 일정한 상수를 더한 위치이다. 키보드의 왼쪽 화살표를 누르면 점 C 를 중심으로 -90°만큼, 오른쪽 화살표를 누르면 +90°만큼 원의 둘레를 따라 카메라가 이동한다. 또한, 플레이어에 초점을 맞추기 위하여 카메라가 캐릭터를 향하도록 회전(rotation)시킨다. 이를 구현한 유니티 C# 코드는 (그림 3)과 같다. 여기에서 변수 degree 는 점 C 를 중심으로 하는 원의 둘레에서 카메라의 위치를 나타낸다. 자연스러운 애니메이션 효과를 위하여 Vector3.Lerp()함수를 통해 시간에 따른 위치를 보간(interpolation)했다.

3. 게임학적 관점에서 2.5D 의 장단점 분석

게임에 3D 그래픽을 적용하는 것은 현실감을 높이는 효과가 있다. 그러나 어드벤처 게임의 경우 3D 그

래픽이 오히려 게임에 대한 도전감을 떨어뜨릴 수 있는데, 이는 3D 그래픽이 주는 현실감이 플레이어로 하여금 게임 세계에서 해결하기 어려운 일을 현실의 업무로 받아들이게 하고, 이것이 플레이어에게 스트레스 요소로 작용하기 때문이다[2]. 이를 보완하기 위해 개발된 것이 3D 그래픽에 2D 그래픽의 요소를 적용한 2.5D 게임이다. 그러나 2.5D 게임은 3D 게임에 비해 단순해 보이고 흥미가 떨어진다는 지적을 받을 수 있다. 따라서 이를 보안해 줄 게임적 요소를 탐구해 보았다.

```
Vector3 CamToPlayer = Vector3.Normalize
(Camera.main.transform.position - player.transform.position);
float angle = Vector3.Angle(Vector3.up, CamToPlayer);
tempquat.eulerAngles = new Vector3(90-angle, -degree, 0);

Camera.main.transform.rotation = Quaternion.Lerp
(Camera.main.transform.rotation, tempquat, lerpSpeed *
Time.deltaTime);
Camera.main.transform.position = Vector3.Lerp
(Camera.main.transform.position, new Vector3(player.position.x +
f_Radius * Mathf.Sin(degree * Mathf.Deg2Rad), player.position.y
+ CameraHeight, player.position.z - f_Radius * Mathf.Cos(degree
* Mathf.Deg2Rad)), lerpSpeed * Time.deltaTime);
```

(그림 3)카메라 회전을 위한 스크립트

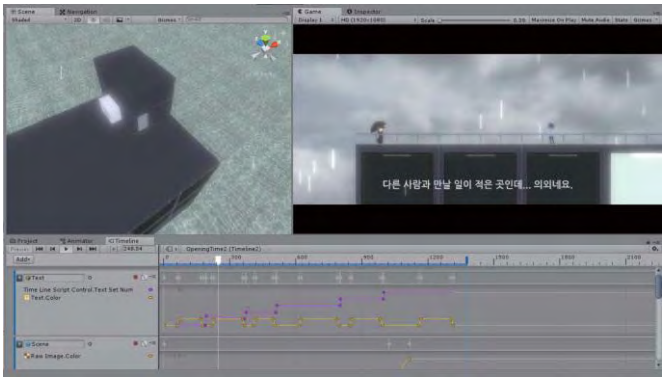
게임의 구성요소는 컨트롤, 커뮤니케이션을 포함하는 인터랙션(interaction), 레이아웃, 컬러, 타이포그래피, 메타포를 의미하는 인터페이스(interface), 배경스토리와 게임의 구조를 다루는 콘텐츠(contents)로 분류할 수 있다. 이 세 구성요소가 유기적으로 조화를 이룰 때, 사용자는 초고조의 몰입 상태를 경험하게 된다[3].

인터랙션에 의한 물리적 몰입감은 사용자의 적극적인 자세와 기술적 지원을 바탕으로 이루어진다. 이를 위해 유니티 3D 게임 엔진이 수많은 게임 개발에 사용되어 왔으며, 그 효율성이 인정되었다[4]. 따라서 이후의 과정에서는 게임의 구조 중, 인터페이스와 콘텐츠를 개발하는 데에 집중하였다.

4. 구현된 게임의 상세 설명

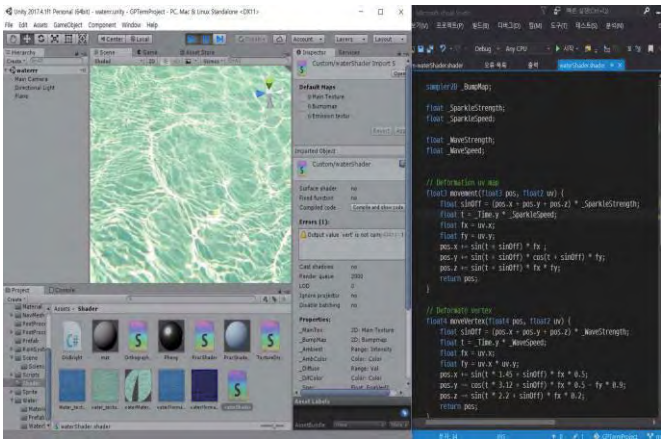
본 프로젝트에서는 의문의 여성 캐릭터를 추가함으로써 게임을 플레이 하기 위한 목적성을 부여하였다. 비가 끊임 없이 오는 세계에서 눈을 뜬 주인공은 의문의 여성을 만난다. 의문의 여성으로부터 주인공은 건물을 따라 높은 곳으로 간다면, 답을 찾을 수 있다는 말을 듣는다.

의문의 여성과의 대화 화면과 게임 세계의 전체적인 분위기를 보여주기 위하여 유니티의 Timeline 에디터를 사용하였다. Timeline 에디터는 유니티 2017부터 지원하는 컷신(cut-scene) 에디터이다. 오브젝트의 스크립트에 구애받지 않고 애니메이팅이 가능하고, Public 변수도 애니메이팅이 가능하기 때문에 대사 출력 등의 많은 응용을 할 수 있다[5].



(그림 4)Unity Timeline Editor

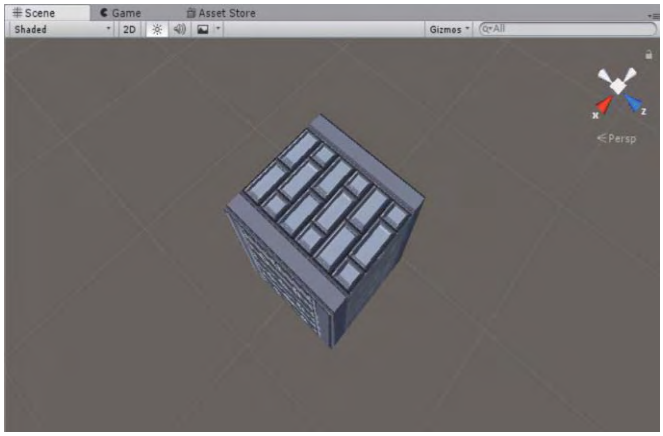
그리고 스토리에 맞는 그래픽 효과를 구현하여 게임의 전체적인 분위기를 만들어 주었다. 게임 속의 세계는 비가 끊임 없이 내려 건물이 잠길 정도가 되었다. 건물 주변에 차오른 물을 표현하기 위하여 water material 을 만들고, 커스텀 셰이더 코드를 작성하였다. 작성된 셰이더 코드에서는 수면이 출렁이는 효과를 위하여 plane 의 표면 좌표를 삼각함수에 따라 변형시킨다. 또한 ambient, diffuse, specular 항목을 넣어 빛에 대한 질감 처리를 해 주었으며, 물 텍스쳐에 normal map 을 입혀서 물이 출렁임에 따라 물 표면이 반짝반짝 빛나게 하였다.



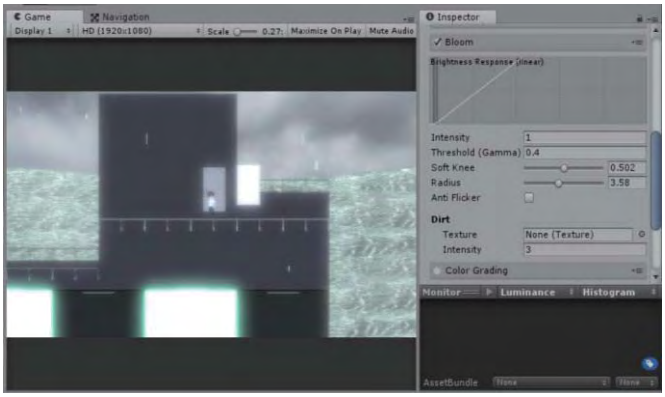
(그림 5)water material 과 Shader 소스코드

건물의 벽돌 같은 경우, 오브젝트에 텍스쳐를 입히고 좀 더 현실적인 렌더링을 위하여 추가로 bump map 을 입혔다.

게임 속 세상은 비가 많이 내려서 전체적인 분위기가 우울하고, 축축하고, 신비로운 느낌을 준다. 이를 표현하기 위해서, 커스텀 셰이더를 통해 건물의 창에만 같은 색의 텍스쳐를 가산해 주어 빛이 나는 효과를 넣었다. 또한, 습기 때문에 흐릿하게 보이는 효과를 표현하기 위하여 카메라에 후처리(post-processing) 기능을 넣었다. 포스트 프로세싱은 화면에 이미지를 표시하기 전에, 카메라의 이미지 버퍼에 전체 화면의 필터 및 효과를 적용하는 과정이다[6]. 여기에서는 습기를 표현하기 위하여 안개(fog)와 bloom 효과를 적용하였다.

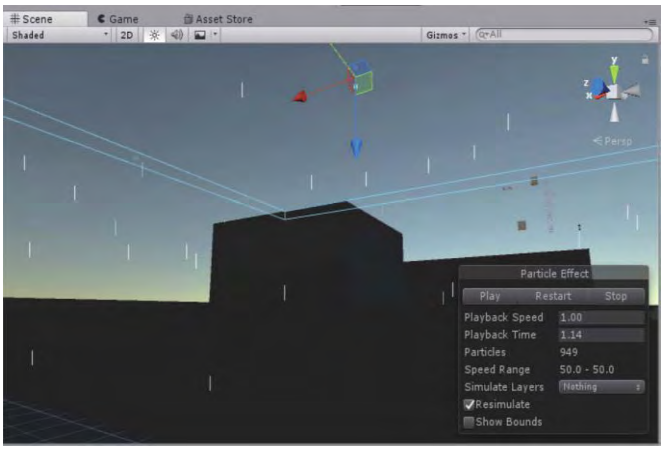


(그림 6)bump mapping 을 적용한 영상



(그림 7)그래픽 효과가 적용된 게임장면과 bloom inspector

게임 세상의 멈추지 않는 비는 유니티의 입자시스템(Particle System)으로 표현하였다[7]. 각 입자는 -Y 방향의 속도를 갖는다. 실제 비는 지상에 닿을수록 공기와의 마찰 때문에 빗줄기가 가늘어진다. 따라서 입자마다 수명을 랜덤으로 할당했으며, 수명이 닳으면서 입자의 길이가 짧아지고 입자의 색은 투명해진다.



(그림 8)입자시스템으로 표현한 비 효과

5. 결론

본 프로젝트에서는 유니티 3D 를 사용하여 2.5D 게임을 만들었으며, 게임학적 접근을 통해 게임성을 보안하였다. 이 과정에서 게임의 스토리와 분위기에 맞는 그래픽 효과를 통해 사용자의 경험 흥미도를 증진시켰다.

참고문헌

[1] https://en.wikipedia.org/wiki/2.5D
[2]권혁인, 이현정, 박진완. (2011). 2D 영상과 3D 입체 영상에서의 액션 어드벤처 게임 몰입도 비교. 한국콘텐츠학회논문지, Vol. 11, No. 1, 157-164, doi: 10.5392/JKCA.2011.11.1.157
[3]박상진. (2006). 게임구성요소와 몰입과의 상관관계에 대한 연구. 한국콘텐츠학회 추계종합학술대회 논문집, Vol. 4, No. 2, 819-823.
[4] https://unity.com/solutions
[5]https://docs.unity3d.com/kr/2018.1/Manual/TimelineSection.html
[6]https://docs.unity3d.com/kr/2017.4/Manual/PostProcessingOverview.html
[7]https://docs.unity3d.com/kr/2017.4/Manual/class-ParticleSystem.html