유니티 엔진을 이용한 모바일 디펜스 게임 제작

김수균*, 박상훈O, 신의성*, 한창민*, 임원규*, 안성옥*
O*배재대학교 게임공학과
e-mail: kimsk@pcu.ac.kr*

Development of Mobile Defense Game using Unity Engine

Sanghoon ParkO, Euiseong Shin*, Changmin Han*, Wongyu Lim*, Soo Kyun Kim*, Syungog An*
O*Dept. of Game Engineering, Paichai University

● 요 약 ●

모바일 게임 시장은 국내외를 막론하고 상승세를 보이는데 그 중 국내 모바일 게임회사 또한 크게 성장하고 있다. 국내 회사들은 유니티 게임 엔진의 장점인 멀티 플랫폼을 활용하여 안드로이드, iOS 등 플랫폼에 구애 받지 않고 최적화 기법을 사용하여 플랫폼에 맞는 게임을 제작하고 있다. 이러한 유니티 엔진의 장점들을 활용해 쉽고 빠르게 디펜스 게임을 제작 해 본다. 본 논문에서는 모바일 게임에서의 오브젝트 풀 기법, 애니메이터 개념과 구현 방법에 대해 설명한다.

키워드: Mobile Game, Unity 3D, Game Engine, Object Pool

I. 서론

멀티 플랫폼을 지원하며 다양한 함수들을 제공하는 유니티 엔진의 장점들을 사용하여 게임을 간편하게 개발할 수 있는 요즘, 기업과 회사가 아니더라도 개개인이 게임 개발에 참여하는 사례가 늘고 있다. 지난 16년 게임물 관리 위원회에서는 사업자가 아닌 개인도 게임 심의를 진행할 수 있도록 절차를 간소화 한 것이 방증이다. 또한 디자이너도 에셋 스토어 등을 활용한 프리랜서가 늘고 있는데 이처럼 개인이 프로젝트를 혼자 진행하는 개발자가 모바일 게임 개발에서 최적화 기법 등에 대한 지식 없이 개발한다면 좋은 아이디어의 게임을 모바일 환경에서 원활히 실행되도록 제작 할 수 없다. 본 논문에서는 개인 게임 개발을 위한 유니티 내의 기능들과 모바일 최적화 기법에 대해 설명한다.

II. 본절

1. 애니메이터

가상 환경인 게임 속 3차원 캐릭터의 애니메이션은 캐릭터를 조금 더 살아있는 것처럼 느끼게 만들어 주며 유저들의 몰입도를 높여주는 역할을 한다. 따라서 이 애니메이션들은 캐릭터의 상태에 따라 재생되어야 하고 관리되어야 하는데 어떠한 상태에 따라 자동으로 변경되어 재생 될 수 있도록 구현하는 방법 중 하나가 FSM(유한 상태 기계)이다. FSM은 유한한 상태를 가진 기계를 구현하는 것이며 상태 패턴과는 다르다. FSM은 상태 전이에 대한 매커니즘이 있는 틀이며 FSM이 상태 패턴이라는 디자인 패턴으로 만들어 진 것이라 할 수 있다. 최신 유니티 버전에서는 이를 직접 스크립트로 구현하지 않고 사용할 수 있게 하는 애니메이터를 지원한다. 애니메이터도 일종의 FSM인데 코드가 아닌 눈으로 볼 수 있는 노드로 구성되어 있다.[1]

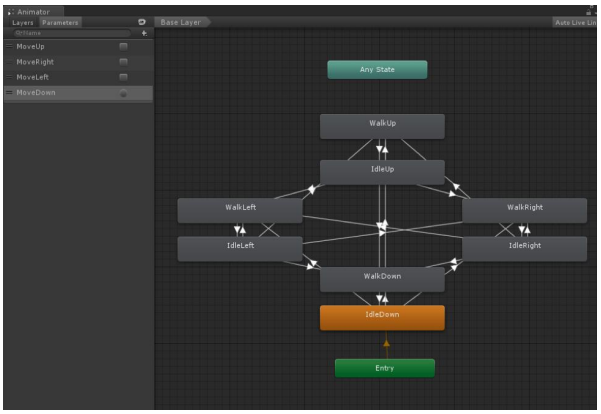

Fig. 1. 유니티 애니메이터

그림1에서와 같이 각 애니메이션 상태는 특정한 조건에 의해 전이되며 이는 개발자가 코딩을 통해 구현하지 않아도 가능하게 해주며 한눈에 알아볼 수 있게 해준다. 사용 할 애니메이션들을 마우스로 드래그 해 애니메이터에 끌어 놓으면 노드가 생기는데 이 노드들의 Transition을 연결해 주기만 하면 되어 간편하고 쉽게 사용이 가능하

다. 각 트랜지션의 Inspector창에 트랜지션의 조건을 붙여 지정 할 수 있으며 그 조건은 변화시킬 변수는 왼쪽 Parameters에서 만들고 스크립트에서 값을 조절할 수 있다. 또한 유니티 애니메이터는 그림2처럼 아바타 마스크 기능을 통해 여러 애니메이션의 블렌딩도 제어할 수 있어 디자이너가 아니더라도 애니메이션을 쉽고 원하는 대로 자유롭게 다룰 수 있게 해준다.

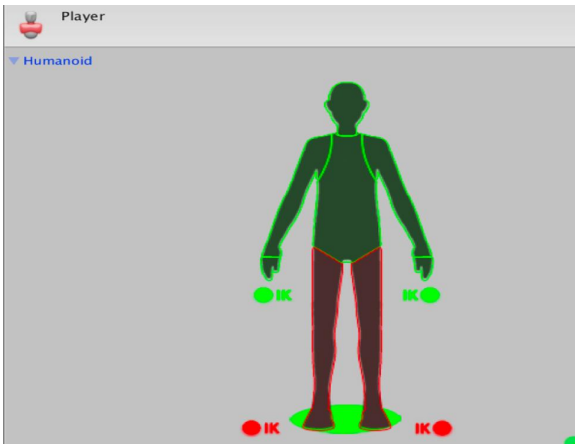

Fig. 2. 휴머노이드 아바타 마스크

아바타 마스크를 생성하고 블렌딩 되지 않게 할 부분을 지정해 준 후 레이어를 통해 블렌딩에 직접 관여할 수 있다. 휴머노이드와 제네릭 타입 모두 가능한 기능이기 때문에 휴머노이드 모델이 아닌 다른 캐릭터 모델이나 사물 모델에도 사용이 가능하여 확장성이 높다.

2. 오브젝트 풀

성능이 좋지 않은 모바일 게임 개발을 위해서는 메모리와 퍼포먼스를 조금 더 빠르게 유지해 줄 필요가 있다.

자주 생성되고 파괴되는 오브젝트들에 대해 미리 생성해두고 화면에 표시되지 않도록 하여 관리하는 방법을 오브젝트 풀 기법이라고 한다.

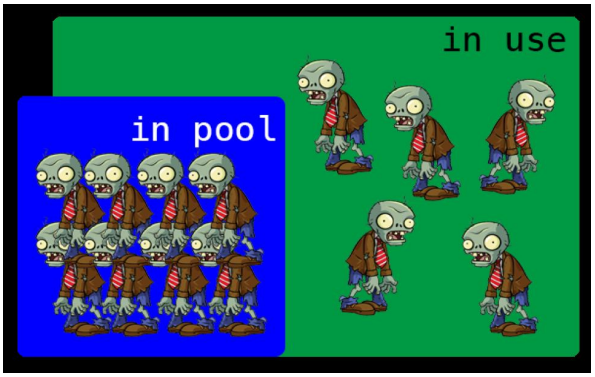

Fig. 3. 오브젝트 풀 기법

그림 3과 같이 사용 중인 오브젝트들을 제외한 다른 오브젝트들은 화면에 표시하지 않으며 메모리상에 놔둔 채로 관리하면 오브젝트 생성, 파괴 시에 발생하는 성능 저하를 막을 수 있다. 그 예로 탄막 슈팅 게임에서 무수히 많은 총알이 개별로 생성되고 파괴된다면 엄청난 리소스를 잡아먹게 된다. 이를 예방하고 퍼포먼스를 유지하는 일종의 방법론이다. 게임 매니저 혹은 리소스 매니저에서 오브젝트에 대한 레퍼런스를 유지하고 있기 때문에 메모리를 많이 잡아먹는 단점이 있으나 성능이 좋지 못하고 실시간 플레이가 중요한 모바일 게임 환경에서 반드시 필요한 방법이다.[2]

Fig. 4. 실제 게임에 적용한 오브젝트 풀

적들이 많이 생성되고 파괴되는 디펜스 게임에서는 적들을 반드시 오브젝트 풀로 관리해 줄 필요가 있다.

그림 4와 같이 오브젝트풀을 묶어 관리할 게임오브젝트아래에 적들이 담겨있고 그 적들을 Enable/Disable하여 관리함으로서 부하가 큰 Instantiate/Distroy함수 사용을 줄일 수 있다.

III. 결론

본 논문에서는 유니티를 이용하여 모바일 플랫폼 게임을 개발하는데 필요한 기능들을 설명하였다. 이러한 기능들은 간단하고 효율적으로 사용할 수 있으며 모바일 플랫폼의 단점을 극복할 수 있다. 때문에 모바일 게임 개발에서 유니티 엔진의 사용은 적은 노력으로도 만족스러운 성과를 기대할 수 있다. 또한 복잡한 스크립트를 작성하지 않고도 기능을 구현할 수 있게 돕는다.

본 논문에서 설명한 애니메이터와 오브젝트 풀은 개인이 게임을 개발하게 될 경우에 효율적인 방법으로 활용될 것이며 어떤 게임에도 적용될 수 있도록 설명하였다.

REFERENCES

[1] Unity Manual : https://docs.unity3d.com/Manual/index.html

[2] Unity Tutorial : https://unity3d.com/kr/learn/tutorials/topics/scripting/object-pooling