한국컴퓨터정보학회 동계학술대회 논문집 제23권 제1호 (2015. 1)

Unity 게임 엔진을 이용한 모바일 게임 개발

윤경섭°, 이현재*, 김락현*, 이한신*
°*인하공업전문대학 컴퓨터정보과
e-mail:ksyoon@inhatc.ac.kr°, dlguswo33@naver.com*, rlafkrguscjs@nate.com*, gkstlsgod@naver.com*

Mobile Game Development using Unity game engine

Kyung-Seob Yoon°, Hyeon-Jae Lee*, Rack-Hyeon Kim*, Han-Sin Lee*
°*Dept. of Computer Science, Inha Technical College

● 요 약 ●

모바일 게임 시장이 날이 갈수록 발전하고 성장하는 가운데 Unity 게임 엔진은 하나의 코드로 여러 플랫폼에서 동작하도록 하여 게임 개발이 전문가들만이 개발할 수 있는 영역이 아닌 누구나 쉽게 개발할 수 있도록 해주는 환경으로 인기를 받고 있다. 이에 따라 Unity 엔진을 이용하여 퍼즐 게임과 타워 디펜스 장르를 합친 퍼즐디펜스로 복합장르를 적용한 게임을 개발하였다.

키워드: 유니티 게임 엔진(Unity Game Engine), 모바일 게임(Mobile Game), 퍼즐게임(Puzzle Game), 타워 디펜스(Tower Defense)

I. 서 론

모바일 게임 시장이 2013년 이후 급성장하면서 많은 모바일 게임이 등장하였다. 이미 모바일 게임 비중이 PC방의 게임 비중보다 높게 나타나 있다.[1]

그러나 국내 게임 시장이 발전하지 못하고 있는 상황이다. 그 이유로는 첫째, 중국의 급성장으로 모바일 사업과 앱 시장이 점차 넘어감으로써 국내 게임 시장이 타격을 입은 것이고, 둘째, 국내 게임이 새로운 도전을 하기보다는 기존에 나와 있는 게임 장르를 기반으로 개발하거나 성공한 해외 게임을 모방하여 게임 개발을 하면서 좇아가기 바쁘기 때문이다.

이러한 문제점을 해결하고 이후 게임 시장의 발전을 위해서라도 새로운 도전과 미래의 게임 시장의 흐름을 읽는 능력이 필요하다고 생각한다. 모바일의 성장과 더불어 게임 개발의 민주화를 일구어낸 Unity 3D 게임 엔진을 이용하여 기존의 게임 장르를 뛰어넘어 복합적인 장르의 게임을 기획, 개발하였다.

II. 관련 연구

1. Unity 게임 엔진

현재 Unity 5.0 버전이 예약 판매되고 있으며 무료 버전으로 최근 4.6 버전이 나오면서 GUI 개발의 큰 변화가 이루어졌다. Unity 게임 엔진은 게임 개발이 주목적이지만 부가적으로 3D 모델링, 애니메이션, 최근 이슈가 되고 있는 가상현실 등에서도 이용되고 있다.

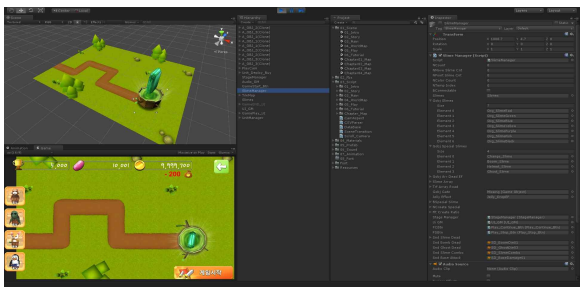

그림 1. Unity 게임 엔진
Fig. 1. Unity Game Engine

Unity 게임 엔진은 3D 모델링 구현 및 프로그래머가 C# 또는 자바 스크립트로 작성한 소스코드를 연결시켜준다. 또한 디자이너는 해당 게임 엔진에 2D 이미지와 3D 모델링 데이터를 연결시켜 눈으로 직접 확인할 수 있다. 뿐만 아니라 Drag and Drop 방식을 이용하여 객체 간의 연결이 보다 쉽고 프로그램 실행 도중에도 데이터의 변화를 쉽게 확인할 수 있다. 스크립트 작성은 MonoDevelop 개발 환경에서 하였고, 언어는 C#을 이용하여 게임 개발을 하였다.

2. 복합장르

이 게임은 기존 게임 시장에 나와 있는 하나의 장르를 기반으로 한 게임과 다르게 2개의 장르를 복합적으로 적용하여 개발한 게임이다. 출/퇴근 시간 많은 이들이 접하고, 짧은 시간에 몰입할 수 있는 퍼즐 게임과 마찬가지로 몰입성이 있는 게임이지만 어느 정도 숙련된 게이머들은 타워만 설치하면 보고 있지 않아도 알아서 게임이 진행되

는 타워 디펜스 장르를 합쳐 퍼즐디펜스 게임이라는 복합장르로 기획하였다.

3. 2D 타일 기반의 맵 제작기

게임의 핵심 부분의 개발은 Unity 엔진만으로도 충분하지만 부수적인 데이터의 생성 및 관리는 Unity 엔진으로는 한계가 있다. 특히 해당 게임은 스테이지를 클리어하면 다음 스테이지가 개방되는 형식의 게임이므로 형태가 비슷한 스테이지를 많이 만들어야 하므로 따로 스테이지만을 제작할 수 있는 툴이 필요하여 그림 2와 같이 DFD 설계를 한 뒤 맵 제작기를 구현하였다.

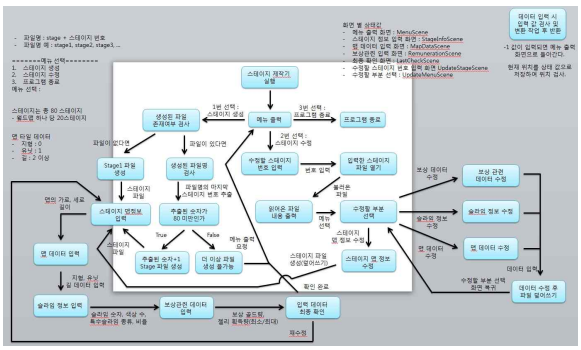

그림 2. 맵 제작기 설계를 위한 DFD
Fig. 2. DFD for Map Creater Design

맵 제작기는 Visual C++ MFC를 이용하여 간단한 2D 타일 기반 방식으로 그림 3과 같이 제작하였고, 해당 프로그램을 이용하여 스테이지 정보를 Binary Text 파일 형태로 출력한다. 이 파일은 다시 Unity 게임 엔진에서 스크립트로 해당 파일을 불러오도록 코드를 작성하여 실제 게임 화면에 나타날 오브젝트들이 생성되도록 구현하였다.

그림 3. 맵 제작기 화면
Fig. 3. Map Creater Screen

III. 설 계

1. 게임 설계

게임을 설계하기에 앞서 팀원들과 게임 세부기획을 하고 게임의 방향을 설정 하였으며, 각자 역할 분담을 하였다. 하나는 게임의 UI 화면 설계 및 관련 스크립트 작성이고, 다른 하나는 타워 유닛에 관한 설계 및 구현이다. 마지막으로 게임의 핵심인 플레이 화면의 동작, 자료구조와 알고리즘을 설계 및 구현하는 작업이다. 게임 세부 설계는 그림 4와 같다.

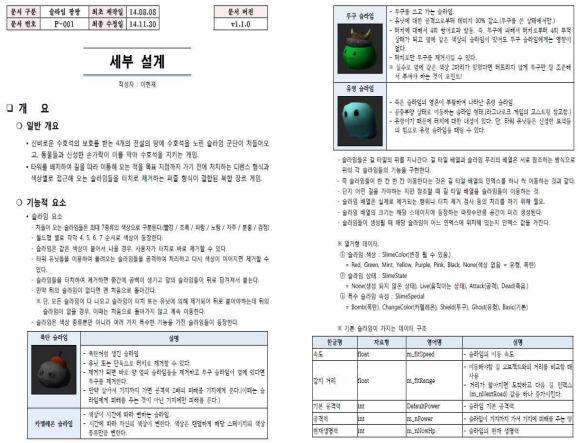

그림 4. 게임 세부 설계
Fig. 4. Game Details Design

게임 세부 설계 자료를 바탕으로 프로토타입을 제작하여 게임을 개발 및 테스트를 해보고 게임의 흥미 여부, 개발 가능성, 디자인 검토 등을 업체와 함께 의논하여 개발을 결정하였다. 이후 디자인 기획서를 바탕으로 그림 5와 같이 3D 모델링으로 디자이너들이 작업을 진행하였다.

그림 5. 3D 모델링
Fig. 5. 3D Modeling

게임 플레이 화면을 구현하기 위해 표 1과 표2와 같이 필요한 데이터 정리 및 자료구조 설계와 메소드 설계를 하였다.

표 1. 자료구조 설계
Table 1. Data Structure Design

※ 슬라임 매니저 데이터 구조

한글명	자료형	영어명	설명
타일 크기	const float	TILE_SIZE	- 타일의 크기 - 초기 값 : 5.0f
슬라임 생성 시간	float	TimerTick	- 슬라임들이 생성되는 주기 - 초기 값 : 1.0f
슬라임 숫자	int	m_nCount	- 해당 스테이지에 등장하는 슬라임 숫자.
현재 이동 중인 슬라임 숫자	int	m_nMoveSlimeCnt	- 현재 활성화 상태에서 화면에 보여진 슬라임의 숫자를 나타낸다. - 최종적으로 이 숫자가 0이 되어야 게임이 끝난다. ※ 물론 0이 되기 전에 기지 HP가 0이되면 마찬가지로 끝난다.(패배)
슬라임 생성 카운트 숫자.	int	m_nPointSlimeCnt	- 슬라임들이 활성화될 때, 슬라임 배열을 끝에서부터 하나씩 감소하면서 참조하기 위한 point 변수.
생성되는 슬라임 종류(색상) 개수	int	m_nColorCount	- 스테이지 정보에서 불러온 색상 종류의 개수를 저장.
임시 인덱스 변수	int	m_nTempIndex	- 새로 생성한 슬라임과 연결을 위해 임시로 저장하는 변수. - 즉, 이동 중인 슬라임들 중 가장 뒷부분에 위치한 슬라임과 방금 활성화된 슬라임을 연결할 때 필요하다. - 이때 가장 뒷부분의 슬라임 index 값을 저장함. - 초기 값 : -1
연결 중인지 여부	bool	m_bConnectable	- 슬라임 연결 중이면 true이고, 그렇지 않으면 false이다. - 초기 값 : false
슬라임 상위 오브젝트	GameObject	Slimes	- 생성된 슬라임들이 들어갈 상위 오브젝트. - 단순히 관리하기 용도로 사용.

표 2. 메소드 설계
Table 2. Method Design

한글명	영어명	반환 타입	파라미터
슬라임 오브젝트 삭제	DestroySlime	void	int index
필요한 변수			
변수명	자료형	설명	
BackIndex	int	- 슬라임의 뒤쪽 index를 저장하는 변수.	
FrontIndex	int	- 슬라임의 앞쪽 index를 저장하는 변수.	
effect	GameObject	- 죽는 이펙트를 저장할 임시변수	
position_Eff	Vector3	- 이펙트가 나타날 position 값을 저장할 변수.	
설명	① 슬라임을 제거하기 전에 먼저 해당 index 슬라임의 앞과 뒤의 index를 가져와 각각 FrontIndex와 BackIndex에 저장한다. ② postion_Eff에 현재 슬라임(Index)의 position을 세팅하는데, z 값만 -1.5f 하여 지나가는 슬라임에 가려져 안 보이지 않도록, 즉 이펙트가 보이도록 한다. ③ 폭탄 슬라임이면 m_gobjArrDeadEF[7]에 들어있는 파티클 프리팹을 동적으로 생성하고, 폭탄이 터지는 사운드를 재생한다. ④ 만약 유령 슬라임이였다면, m_gobjArrDeadEF[8]에 들어있는 파티클 프리팹을 동적으로 생성하고 유령이 죽는 사운드를 재생한다. ⑤ 나머지 슬라임들은 현재 슬라임의 색상 값을 가지고 m_gobjArrDeadEF[색상값(0-6)]에 들어있는 파티클 프리팹을 동적으로 생성한다.(사운드는 여기서 재생하지 않는다.) = 나머지 일반 슬라임과 카멜레온 슬라임은 보통 터치나 유닛에 의해서 죽게 되는데, 유닛에게 죽게되면 Slime 스크립트의 Hurt 함수에서 사운드가 재생되고, 터치로 죽게되면 죽는 사운드가 아닌 Combo 사운드가 재생된다. ⑥ LinkedSlime() 함수를 호출하여 앞과 뒤의 슬라임을 연결한다. ⑦ index 슬라임을 Destroy하여 제거하고 이동 중인 슬라임 count를 하나 감소시킨 뒤 슬라임 배열의 index 요소를 null로 변경한다.		

IV. 구 현

구현은 C# 스크립트로 작성하였다. 먼저 각각의 오브젝트가 수행될 스크립트를 작성하고 이들을 통합 관리하고 연결시키기 위해 Manager 스크립트를 작성하였다.

각각의 오브젝트에 스크립트를 추가하여 내부 자료구조와 어떤 일들을 처리할지 설계 내용을 바탕으로 구현을 하였다. 또한 데이터베이스와 구글 플러스 클라우드 데이터 연동을 위해 PlayerPrefs라는 Unity만의 DataBase에 데이터를 저장하고 불러올 수 있도록 데이터 설계를 하였다.

```
[RequireComponent(typeof(AudioSource))]
public class SlimeManager : MonoBehaviour {
    private const float TILE_SIZE = 5.0f;
    private float TimerTick = 1.0f;
    public int m_nCount;
    public int m_nMoveSlimeCnt;
    public int m_nPointSlimeCnt;
    public int m_nColorCount;
    public int m_nTempIndex;
    public bool m_bConnectable = false;
    public static int m_getJelly;

    public GameObject Slimes;

    public GameObject[] m_gobjSlimes;
    public GameObject[] m_gobjSpecialSlimes;
    public GameObject[] m_gobjArrDeadEF;
    public Slime[] m_slimeArray;
    public Transform[] m_trfArrayRoad;
    public GameObject m_gobjGate;
    public GameObject m_JellyEffect;

    public bool[] m_bSpecialSlime;
    public int[] m_nCreateSpecial;
    public float[] m_fltCreateRatio;

    public StageManager m_stageManager;
    public UI_GM m_uiGM;
    public Play_Continue_Btn _PCBtn;
    public Play_Stop_Btn _PSBtn;
    public AudioClip[] m_sndSlimeDead;
    public AudioClip m_sndBombDead;
    public AudioClip m_sndGhostDead;
    public AudioClip m_sndSlimeCombo;
    public AudioClip m_sndBaseAttack;
```

그림 6. SlimeManager 스크립트의 자료구조
Fig. 6. Data Structure Of SlimeManager Script

```
33 public class Slime : MonoBehaviour
34 {
35     private float m_fltSpeed;                // 슬라임 속도.
36     private float m_fltRange = 1.25f;        // 감지 범위.
37     private int DefaultPower = 10;           // 슬라임 기본 공격력.
38     private int m_nPower;                    // 공격력.
39     private int m_nNowHp;                    // 슬라임 현재 생명력.
40
41     public int m_nNextRoad = 0;              // 다음 길 인덱스.
42     public int m_nMyIndex;                   // 슬라임 인덱스.
43     public int m_nFrontIndex;                // 앞 슬라임 인덱스.
44     public int m_nBackIndex = -1;            // 뒤 슬라임 인덱스.
45
46     private Transform[] m_trfArrayRoad;      // 길 배열.
47     private HP_Gauge m_targetBase;           // 기지
48
49     public SlimeColor m_slimeColor;          // 슬라임 색상.
50     public SlimeState m_slimeState;          // 슬라임 상태.
51     public SlimeSpecial m_slimeSpecial;      // 특수 슬라임 속성.
52     public bool m_bJelly = false;            // 젤리를 가지고 있는지 여부.
53
54     public SlimeAni m_slimeAni;
55     // 통신용 매니저들.
56     private StageManager m_stageManager;
57     private SlimeManager m_slimeManager;
58
59     void Awake(){
60         m_stageManager = GameObject.FindGameObjectWithTag("StageManager").GetComponent<StageManager>();
61         m_slimeManager = GameObject.FindGameObjectWithTag("SlimeManager").GetComponent<SlimeManager>();
62         m_targetBase = GameObject.FindGameObjectWithTag("Gauge").GetComponent<HP_Gauge>();
63     }
64
65     // HP, 속도, 공격력, 이동할 길 세팅.
66     void Start(){
67         if(m_slimeSpecial == SlimeSpecial.Bomb){
68             m_nPower = DefaultPower * 2;
69         }
70         else{
71             m_nPower = DefaultPower;
72         }
73         m_trfArrayRoad = m_stageManager.GetArrayRoad();
74         m_nNowHp = m_stageManager.sData.slimeHp;
75         m_fltSpeed = m_stageManager.sData.slimeSpeed;
76     }
77
```

그림 7. 슬라임 스크립트의 구현
Fig. 7. Implementation Of Slime Script

스크립트에서 public으로 선언한 변수는 Unity 환경에서 객체를 Drag and Drop 방식으로 끌어다 놓으면 자동으로 변수 값이 들어가며 기본 자료형(int, float 등)은 직접 값을 입력하여 프로그램 도중에도 실시간으로 테스트해볼 수 있다.

V. 결 론

해당 게임을 개발하면서 Unity 게임 엔진의 개발 편의성을 경험해 볼 수 있었고, 기획-설계(내부/디자인)-구현-테스트까지 전체적인 절차를 경험해 볼 수 있어서 좋은 기회가 되었다.

또한 단일 장르는 이미 많은 게임 개발자가 개발한 경험이 있고, 자료가 많은 반면에 복합장르는 서로의 교집합이 적으면 적을수록 연관성을 찾기도 어렵고 구현이 어렵다.

이 게임에서는 하나의 장르를 기반으로 한 게임과 다르게 퍼즐 게임과 타워 디펜스 장르를 합쳐 퍼즐디펜스 게임이라는 복합장르로 개발함으로써 모바일 게임의 흥미를 극대화 시킬 수 있는데 기여할 수 있다고 사료된다.

이 모바일 게임 개발은 현장 실습 업체((주)아인픽춰스)와 함께 공동 개발하였으며, 현재 구글과 연동 작업을 마무리 하고 1월 말 ~ 2월 초에 마켓에 출시할 예정이다.

참고문헌

[1] 한국콘텐츠진흥원, “2014 대한민국 게임 백서 요약본”, 한국콘텐츠진흥원, 23-24, 2014

[2] 이득우, “(개정판)유니티 4 게임 개발의 정석”, 2013

[3] 지국환, “C# 초보자를 위한 유니티 게임 개발 스타트업”, 2014

[4] 데브코리아, http://devkorea.co.kr

[5] 데브코리아, 질문답변 내용, http://devkorea.co.kr/bbs/board.php?bo_table=m03_qna&wr_id=54914&sfl=wr_name%2C1&stx=%ED%81%B4%EB%A6%B0%ED%81%B4%EB%A6%AC%EC%96%B4&sop=and