시스템분석설계
개인 포트폴리오

컴퓨터정보공학과 2-PB
20210957 김용민

목차

I. 유니티

i. 유니티란 무엇인가?
ii. 유니티의 장단점

I - i. 유니티란 무엇인가?

" 게임 개발을 위한 엔진 "

게임 엔진이란?

- 게임의 개발에 기반이 되는 그래픽 엔진, 물리 엔진, 오디오 엔진, UI 시스템, 게임 플레이 프레임워크 등의 핵심 기능들을 담은 소프트웨어.
- 쉽게 말하자면 게임 개발을 위해 여러 기능을 제공함으로써 **게임을 쉽게 제작할 수 있게 도와주는 프로그램**임.
- 여러 상용 엔진들이 있으나 개발사에서 주로 사용하는 엔진은 '**언리얼 엔진**'과 '**유니티**' 두 가지임.

유니티란?

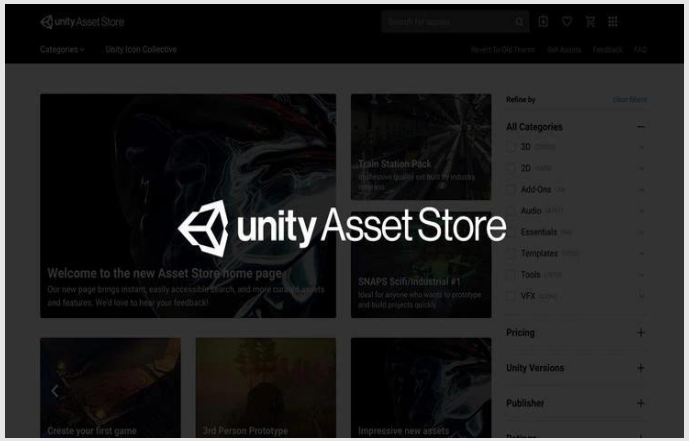
- 2004년 8월 Unity Technologies가 개발
- 3D 및 2D 비디오 게임의 개발 환경을 제공하는 게임 엔진이자, 3D 애니메이션과 건축 시각화, 가상 현실(VR) 등 인터랙티브 콘텐츠 제작을 위한 통합 제작 도구
- 개발에 사용하는 스크립트 언어는 C# (자바스크립트 기반의 'UnityScript'를 사용 가능 했으나 지원 중단됨)

I - i. 유니티란 무엇인가?

유니티란 무엇인가?

유니티의 장단점

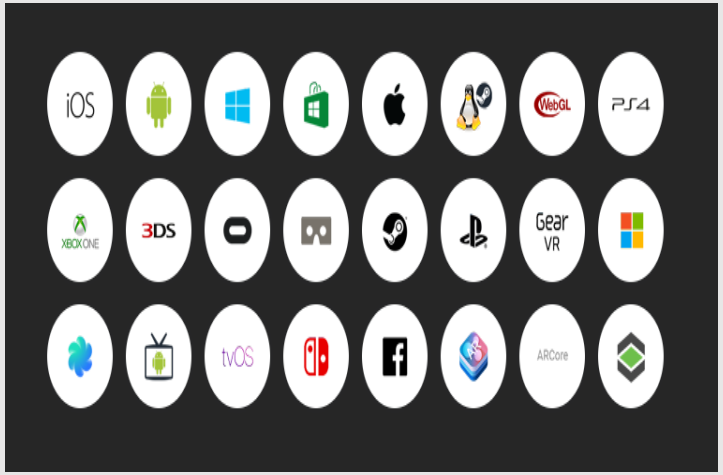
- **연 매출 및 자본금 10만 달러 미만일 경우 무료**로 이용 가능. 그 이상일 경우 Plus 플랜(월 $40), Pro 플랜(월 $150) 등을 구매해서 사용해야 함
- 유니티 개발사에서 학생 및 교육자를 위한 **교육 프로그램을 제공**하고 있으며, 4단계로 구성된 **자격증 시험**도 제공함
- '**에셋 스토어**'가 존재하여 각종 리소스부터 스크립트, 플러그인, 미리 만들어 놓은 AI 등을 이용자들끼리 무료로 공유하거나 사고 팔 수 있음

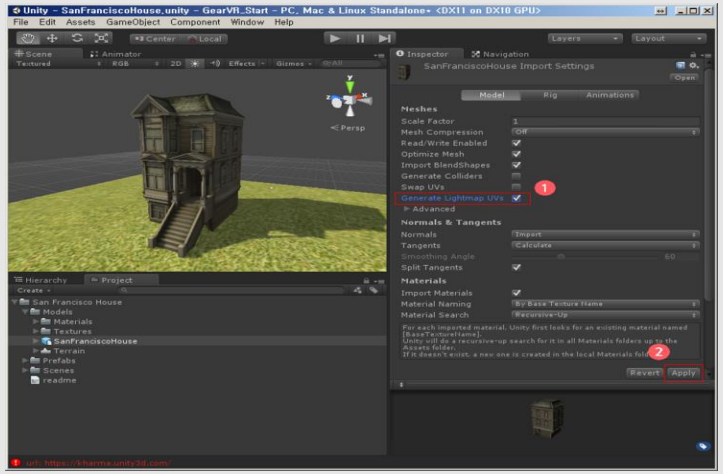


- 프로젝트를 씬과 오브젝트로 다루며, 물리적인 계산을 엔진이 해주기 때문에 직관적이고 편한 개발을 하기에 용이함
- 글로벌 사용도가 매우 높아 개발에 필요한 여러 정보를 손쉽게 획득할 수 있음
- 최신 모바일 게임 중 유니티 기반의 게임이 55%에 달하며, 최근 주목받고 있는 AR/VR 콘텐츠에서의 비율은 60%에 달함
- 유니티 대표 게임 : 하스스톤, 포켓몬 GO, 카트라이더 러쉬 플러스, 블루 아카이브, etc.

I - ii. 유니티의 장단점

유니티의 장점





I. 멀티 플랫폼 빌드를 지원

- 빌드가 간편하여, 플랫폼별 특성에 맞게 어느 정도의 조정작업과 최적화만 해주면 따로 개발할 필요 없이 다양한 플랫폼으로의 발매가 가능
- 현재 윈도우, 맥OS, iOS, 안드로이드, 플레이스테이션, 엑스박스, 닌텐도 스위치, 웹브라우저 등 27개의 플랫폼에서 사용 가능한 콘텐츠를 만들 수 있음

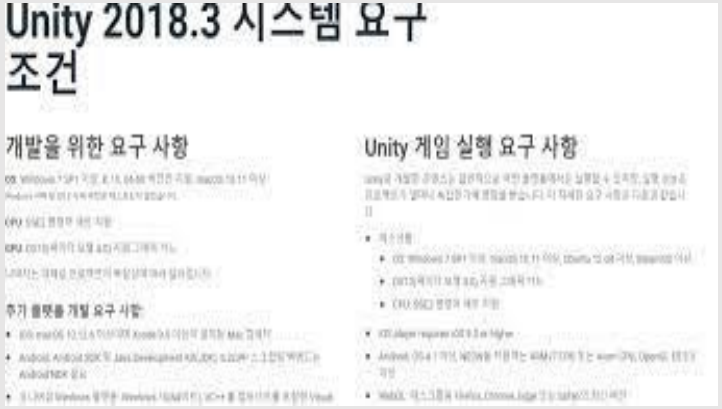
II. 비교적 쉬운 제작 환경

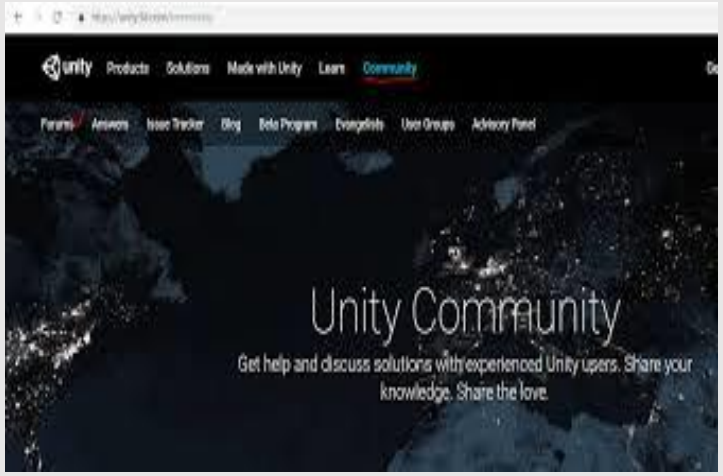
- WYSIWYG(What You See is What You Get) 방식의 GUI를 제공하여 사용자가 직관적으로 내부 에셋의 위치를 변경 및 적용하거나, 에셋의 임포트 등을 매우 손쉽게 할 수 있음
- 게임 제작에 대해 깊이 알지 못해도 비교적 간단하게 게임제작을 할 수 있음.
- 유튜브, 책 등을 보며 따라 해도 될 정도로 간단하게 학습 가능

유니티란 무엇인가?

유니티의 장단점

I - ii. 유니티의 장단점





III. 낮은 요구 사양

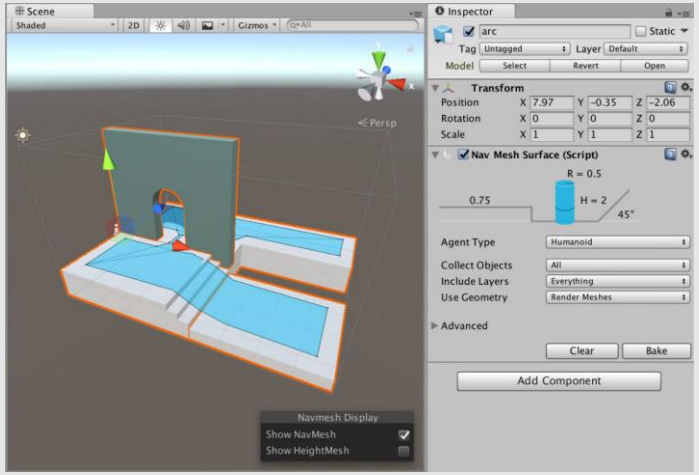
- ❖ 버전 업이 진행되면서 요구 사양이 점점 높아지고 있지만 AAA급 고퀄리티 게임을 개발할 수 있는 다른 메이저 게임 엔진에 비하면 비교적 가벼운 편이라 저사양 PC에서도 저사양 타깃의 간단한 게임 정도는 무리 없이 개발이 가능

IV. 넓은 사용자 층과 그로 인한 풍부한 관련자료

- ❖ 개발 초보자나 비 프로그래머에서부터 고급 개발자까지 사용자층이 다양하고 그 수가 많아 단계적으로 필요한 자료를 구하기가 용이함
- ❖ 사용자가 많은 만큼 에셋 스토어에서 얻을 수 있는 에셋의 종류가 다양하여 소규모 개발사도 적은 비용으로 고품질의 게임을 개발하기에 용이

I - ii. 유니티의 장단점

유니티의 단점



UNREAL ENGINE

unity

I. 빈약한 고급 기능

- 개발하기에 직관적이고 쉽다는 장점 때문에 초보자들이 이용하기 좋지만, 역설적으로 간편하게 적용할 수 있는 고급 기능들이 부족함.
- 그래서 웬만한 고급 기능들은 개발자가 직접 구현해야 되기 때문에 현업에서 사용하려면 번거로움을 감수해야함

II. 소스 코드 비공개

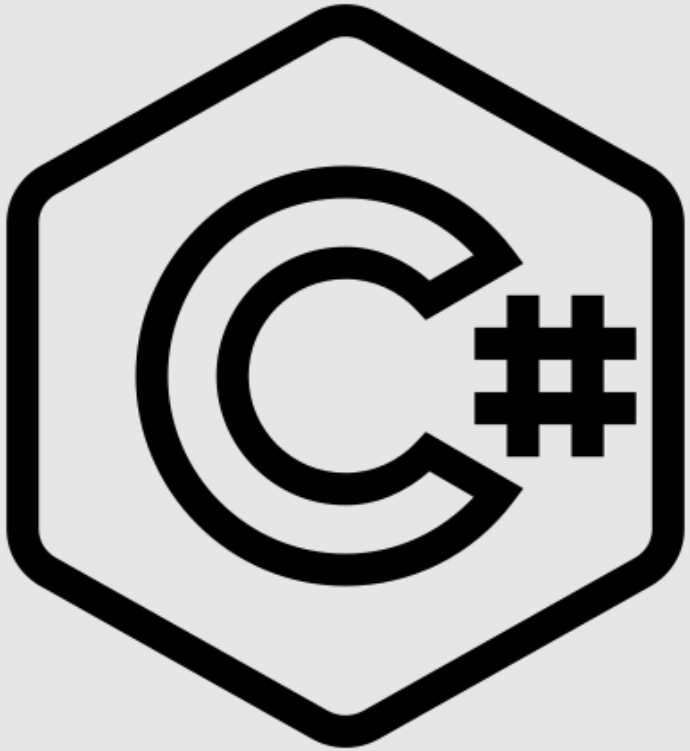
- 유니티는 기본적으로 월간 비용을 지속 지불하더라도 전체 소스코드는 여전히 비공개이기 때문에, 소스코드가 포함된 라이선스를 원할 경우 그 비용이 지나치게 올라 상황에 따라서 언리얼 엔진의 사용을 고려해야 될 수준임

III. 커뮤니티의 질적인 문제

- 진입 장벽이 낮은 탓에 어린 학생이나 이상한 사람이 많아서 잘못된 지식을 알려주거나 분쟁이 생길 수 있어 사용자의 자체적인 사리분별 필요

II. C#

i. C#이란 무엇인가?
ii. C# 기초 문법



Ⅱ - i. C#이란 무엇인가?

스크립트 작성을 위해 필요한 C#

‘ C# ‘

- ❖ 마이크로소프트에서 개발한 **객체 지향 프로그래밍 언어**로, 닷넷 프레임워크의 한 부분으로 만들어졌으며 나중에 ECMA (ECMA-334)와 ISO (ISO/IEC/23270)의 표준으로 자리 잡았음
- ❖ C++와 Java의 문법과 비슷한 문법을 가지고 있음

C#의 특징

- ❖ Java가 자바 가상 머신(JVM)이 필요하듯 C#은 **닷넷**이 필요함.
- ❖ 일단은 Java보다 차세대 언어이므로 성능적으로 우위인 부분이 많긴 하지만 여전히 C/C++보다 다소 느림 (가상머신 언어의 태생적인 한계점)
- ❖ 윈도우, 웹, 게임 및 모바일 프로그래밍 등 모든 영역에서 사용되는 범용 프로그래밍 언어임
- ❖ 개발도구는 일반적으로 Visual Studio 를 사용함
- ❖ 확장자로 ‘.cs’를 사용

Ⅱ – ii. C# 기초 문법

아주 간단한 C# 프로그램

예 제

```
namespace Intro_Ex1
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Console.WriteLine("Hello World...");
        }
    }
}
```

Hello World 라는 텍스트를 콘솔 화면에 출력하는 프로그램

- ❖ 모든 C# 프로그램은 Main()이라는 시작 함수(메소드)를 가져야 함. Main() 메소드는 임의의 클래스 안에서 존재하며, 프로그램 상에 단 1개만 있어야 함.
- ❖ Main()은 static으로 선언되며, 메소드 인자는 string[] 문자열임.
- ❖ System.Console은 .NET Framework 클래스이며, WriteLine은 화면에 데이터를 Console클래스에 출력하는 메소드임.

Ⅱ – ii. C# 기초 문법

C# 주석과 자료형 및 변수 선언

예 제

```
namespace Intro_Ex2
{
    class Program
    {
        static void Main(string[] args)
        {
            // 코멘트: 한 라인 코멘트는 두개의 슬래시 사용함
            int a = 1;

            int b = 1;  // 코멘트: 하나의 문장 뒤에 코멘트를 달 수 있음

            /*
                복수 라인에 대한 코멘트
                int c;
                int d;
            */
        }
    }
}
```

- C# 에서 주석은 한 라인에 대해 쓸 경우 // 을 사용하고 복수 라인에 대해 쓸 경우 /* */ 를 사용함
- 정수형의 기본 타입은 8가지가 존재하나 주로 int 를 사용함
- 실수형은 float, double, decimal 등이 있으나 주로 double 을 사용함
- 문자형은 한 글자를 표현하는 char형과 문자열을 표현하는 string형이 있음.
- char은 **작은 따옴표**를 사용하고 string은 **큰 따옴표**를 사용함
- Boolean 형은 오직 true / false 값만으로 담을 수 있음

Ⅱ – ii. C# 기초 문법

C# enum (열거형)

예제

```
class Program
{
    enum City
    {
        Seoul,   // 0
        Daejun,  // 1
        Busan = 5,  // 5
        Jeju = 10   // 10
    }

    static void Main(string[] args)
    {
        City myCity;

        // enum 타입에 값을 대입하는 방법
        myCity = City.Seoul;

        // enum을 int로 변환(Casting)하는 방법.
        // (int)를 앞에 지정.
        int cityValue = (int) myCity;

        if (myCity == City.Seoul) // enum 값을 비교하는 방법
        {
            Console.WriteLine("Welcome to Seoul");
        }
    }
}
```

- C#의 키워드 enum 은 열거형 상수를 표현하기 위한 것으로, 이를 이용하면 **상수 숫자들을 보다 의미 있는 단어들로 표현**할 수 있어서 프로그램을 읽기 쉽게 해줌
- enum 문은 클래스 안이나 네임스페이스 안에서만 선언될 수 있음. 즉, 메소드 안이나 속성 안에서는 선언되지 않음
- enum 타입은 **숫자형 타입과 호환이 가능**함. 만약 enum 타입의 변수를 int로 캐스팅하면 해당 enum값의 숫자 값을 얻게 됨. 또한 enum 타입의 변수는 enum 리터럴값과 서로 비교할 수 있음

Ⅱ – ii. C# 기초 문법

C# foreach 반복 구문

예제

```
static void Main(string[] args)
{
    // 3차배열 선언
    string[,,] arr = new string[,,] {
            { {"1", "2"}, {"11","22"} },
            { {"3", "4"}, {"33", "44"} }
    };

    //for 루프 : 3번 루프를 만들어 돌림
    for (int i = 0; i < arr.GetLength(0); i++)
    {
        for (int j = 0; j < arr.GetLength(1); j++)
        {
            for (int k = 0; k < arr.GetLength(2); k++)
            {
                Debug.WriteLine(arr[i, j, k]);
            }
        }
    }

    //foreach 루프 : 한번에 3차배열 모두 처리
    foreach (var s in arr)
    {
        Debug.WriteLine(s);
    }
}
```

- C# foreach 문은 배열이나 컬렉션에 주로 사용하는데, 컬렉션의 각 요소를 하나씩 꺼내 와서 foreach 루프 내의 블록을 실행할 때 사용됨.
- C#에서 for 와 foreach는 성능적 측면에서 for 가 경우에 따라 약간 빠를 수 있지만 대부분의 경우 성능적 차이는 크지 않으며, foreach 는 for 보다 훨씬 간결한 코드를 제공한다는 장점이 있음.
- 따라서, 다중 배열을 처리할 경우 foreach 문을 사용하는 게 더 편리함

Ⅱ – ii. C# 기초 문법

C# 네임스페이스

예 제

```
namespace MyNamespace
{
    class A
    {
    }

    class B
    {
    }
}
```

예 제

```
//첫번째 방식
static void Main(string[] args)
{
    System.Console.WriteLine();
}

//두번째 방식
using System; //System 네임스페이스 지정

static void Main(string[] args)
{
    Console.WriteLine();
}
```

- .NET Framework는 무수하게 많은 클래스들을 가지고 있는데, 그것들을 충돌없이 보다 편리하게 관리/사용하기 위해, .NET에서 네임스페이스를 사용함. C#에서도 이러한 개념을 적용하여 **클래스들이 대개 네임스페이스 안에서 정의**됨.
- 네임스페이스는 같은 그룹의 클래스를 묶는 도구임(Java의 package와 유사)
- 네임스페이스를 사용하기 위해서는 두가지 방식이 있음. 첫째는 **클래스명 앞에 네임스페이스를 전부 적는** 경우, 둘째는 프로그램 맨 윗단에 **해당 using을 사용하여 C# (.cs) 파일에서 사용하고자 하는 네임스페이스를 한번 설정해주고, 이후 해당 파일 내에서 네임스페이스 없이 직접 클래스를 사용**하는 경우임. 실무에서는 주로 두번째 방식을 사용함.

Ⅱ – ii. C# 기초 문법

C# yield

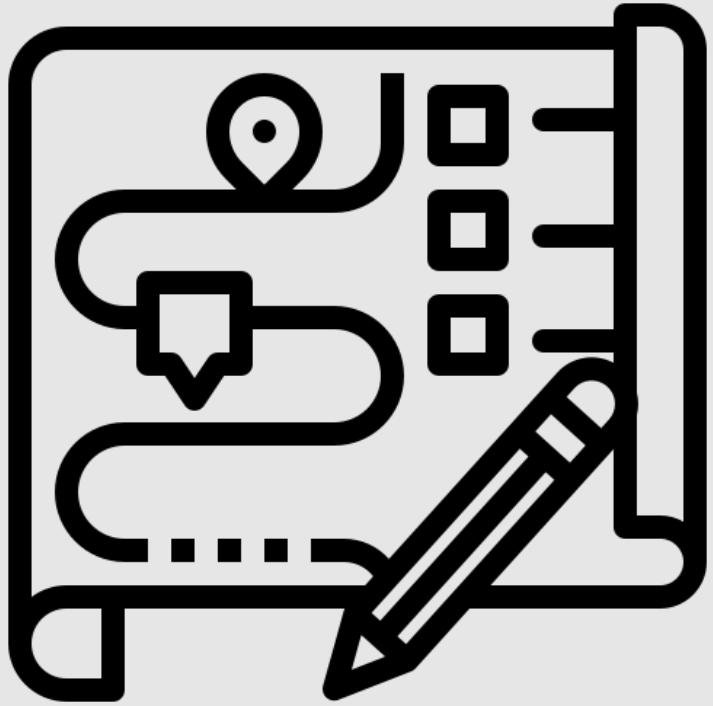
예제

```
using System;
using System.Collections.Generic;

class Program
{
    static IEnumerable<int> GetNumber()
    {
        yield return 10;  // 첫번째 루프에서 리턴되는 값
        yield return 20;  // 두번째 루프에서 리턴되는 값
        yield return 30;  // 세번째 루프에서 리턴되는 값
    }

    static void Main(string[] args)
    {
        foreach (int num in GetNumber())
        {
            Console.WriteLine(num);
        }
    }
}
```

- Yield 키워드는 호출자(Caller)에게 컬렉션 데이터를 하나씩 리턴할 때 사용함. 흔히 Enumerator(Iterator)라고 불리우는 이러한 기능은 집합적인 데이터셋으로부터 데이터를 하나씩 호출자에게 보내주는 역할을 함
- yield는 2가지 방식으로 사용되는데,
- yield return은 컬렉션 데이터를 하나씩 리턴하는데 사용되고,
- yield break는 리턴을 중지하고 Iteration 루프를 빠져 나올 때 사용함

III. 게임 기획

i. 기획 시작 단계
ii. 플로우 차트 작성하기



Ⅲ – i. 기획 시작 단계

기획 시작 단계

플로우 차트 작성하기

게임을 만들 때 가장 먼저 필요한 과정

" 어떤 게임을 만들까? "

❖ 게임 기획의 시작은 '**어떤 게임을 만들까?**'라는 질문에서 시작된다. 회사나 팀에 따라 다르겠지만 만들어야 할 게임을 기획자가 모두 정하는 것은 아니며, 팀원과의 아이디어 회의를 통해 혹은 만약 회사에 소속되어 있다면 회사의 지시에 의해 어느 정도 방향이 정해지는 경우도 있다.

❖ 어떤 게임을 만들 것인지 정하는 과정은 단순히 툭하고 아이디어가 떠오르는 것으로 끝나지 않는다. 게임 기획의 모든 과정이 그러하겠지만 어떤 게임을 만들 것인지 정하는 과정은 **수많은 질문들을 통해 이루어진다.** 여러가지 질문을 던져보고 그 질문에 대해 답을 내려가는 과정이 이루어져야 기반이 탄탄해져서 향후 기획을 할 때 갈팡질팡하지 않는다.

❖ 시작 단계에서 던져야 할 질문이 여러가지 있겠지만, 대표적으로 3가지를 꼽을 수 있다.

Ⅲ – i. 기획 시작 단계

기획 시작 단계

플로우 차트 작성하기

첫째. 어떤 재미를 줄까?

게임에서 가장 중요한 '재미'

❖ 사람마다 지향하는 바가 다르겠지만 일반적으로 게임에서 가장 중요한 것은 **'재미가 있는가?'**이다. 재미있는 게임을 만들기 위해서는 게임을 기획하는데 있어 유저에게 어떤 재미를 줄 것인지 명확하게 하고 넘어가야 한다. 기획자 스스로가 자신의 게임이 어떤 재미를 줄 수 있을 지 파악하지 못한다면 당연히 유저에게도 제대로 전달 될 리가 없다.

그렇다면 재미라는 것은 어떻게 만들어 낼 수 있을까?

❖ 재미라는 것은 지극히 주관적이기 때문에, 누군가는 재밌는 게임이 누군가에게는 재미가 없을 수 있다. 시장의 트렌드에 따라, 이용하는 유저의 연령이나 성별에 따라 다양한 결과가 나올 수 있다.

❖ 그렇기에 기획자는 먼저 **다양한 게임을 즐겨보는 것**이 좋다. 특히 스마트폰 게임의 경우 변화에 민감하기 때문에 성공한 게임들이 왜 성공했고, 왜 재밌어 하는지 파악하는 것이 큰 도움이 된다.

❖ 이러한 경험들을 바탕으로 내 **게임의 타겟층**을 어떻게 할 지 정하고, **어떤 부분에서 재미를 줄 지** 구상해 본다. 남자와 여자, 3040과 1020 등 사용자 계층 마다 게임에 대한 경험이나 가치관이 다르기 때문에 타겟층을 확실히 하는 것이 재미 요소를 결정하는 데 큰 영향을 끼치기 때문이다. 또한 조작감에서 재미를 줄 지, 스토리나 디자인 적인 부분에서 재미를 줄 지 기획 단계에서 생각을 해 놓아야 제작 단계에서 게임의 방향이 흐트러지지 않는다.

Ⅲ – i. 기획 시작 단계

기획 시작 단계

플로우 차트 작성하기

둘째. 무엇을 소재로 만들어야 할까?

신선하고 특이한 소재가 무조건 좋은 것은 아니다.

- 왕자와 공주, 앨리스, 히어로, 좀비 등 스마트폰 게임에는 다양한, 생각지도 못한 소재들로 구성된 게임들이 등장하곤 한다. 새들을 날려 돼지들을 혼내 주는 게임이나, 식물을 심어서 좀비들의 공격을 방어하는 게임, 동물들이 갑옷을 입고 좀비들과 싸우는 게임 등 **신선한 소재는 유저들의 관심을 끄는 주된 역할을 하는 부분**이다.

- 그렇다고 **신선하고 특이한 소재가 무조건 좋다는 것은 아니다**. 오히려 흔히 접할 수 있는 것들이 식상해 보일 수는 있어도 유저들에게 익숙함을 줄 수 있다. 단순히 소재만 이야기한다면 신선하고 특별한 것이 좋을 수도 있겠으나 게임은 단순히 소재 뿐만 아니라 다양한 요소들이 조합되어 만들어지는 컨텐츠이기 때문에 전략에 따라 소재를 선택하는 것이 중요하다.
- 소재를 정하는 방법은 다양하다. 굳이 게임이 아니더라도 다양한 창작물에서 어떤 소재들을 사용했는지 찾아보고 결정해도 된다. 하지만 소재에 따라 다소 매니악해지거나, 거부감이 들거나 혹은 너무 식상해 보일 수 있기 때문에 신중히 결정해야 한다. 따라서 소재를 정할 때에는 앞서 **정했던 타겟층을 생각**하여 구상해야 한다. 만약 타겟층을 여성으로 잡은 게임에서 좀비나 오크, 노출이 심한 여성 캐릭터를 소재로 사용한다면 심한 거부감을 불러일으킬 수 있다.

Ⅲ – i. 기획 시작 단계

기획 시작 단계

플로우 차트 작성하기

셋째. 무슨 차별성으로 어필해야 할까?

하늘 아래 새로운 것은 없다

❖ '하늘 아래 새로운 것은 없다'라는 말이 있다. 하지만 **새롭다는 것과 차별화가 된다는 말은 개념이 다르다.** 기존에 애니팡과 같은 게임들이 무더기로 나왔을 때 실시간 대전이라는 기능을 이용해 유저들 간에 퍼즐 대결을 벌이게 한다면 기존 게임들과 비교해 차별화가 될 수 있다. 차별성이라는 것은 게임의 소재로도, 컨텐츠로도 풀어나갈 수 있는 부분이다.

❖ 모든 부분이 중요하지만 특히 차별화되는 부분에 대해서는 더욱 신경을 쓰는 것이 좋다. 사실 게임이라는 것이 기획서를 아무리 잘 쓰고 문서를 잘 쓴다고 하더라도 재미있는지, 없는지는 개발이 되어봐야 아는 것이다. 하지만 그 전에 어떤 게임을 만들지 정하는 과정에서 이 게임이 다른 게임에 비해 어떤 부분이 차별화되는지 뚜렷하게 전달할 수 있다면 그 게임은 상당히 매력적인 게임이 될 수 있다. 특히 회사에 소속되어 있다면 회사의 관리자들에게 컨펌을 받아야 하는데 차별성은 아주 중요한 역할을 한다.

❖ 위에서 설명했듯 차별성을 정하는 데에는 꼭 새로운 것을 만들어야 한다는 강박을 가질 필요는 없다. 만약 TCG 게임을 기획한다고 할 때 새로운 효과로 차별성을 둔다고 한다면 아마 힘든 여정이 될 것이다. 이미 시장에는 셀 수 없을 정도의 카드들이 나와 있기 때문이다. 그렇기에 UI/UX 나 이펙트에서 차별을 두든지, 스토리 라인에서 차별을 두든지 **다양한 방면에서 생각**을 해봐야 한다.

Ⅲ – ii. 플로우 차트 작성하기

기획 시작 단계

플로우 차트 작성하기

자신의 생각을 팀원에게 공유하는 방법

게임 개발은 다양한 사람들이 모여 이루어진다.

- 게임 개발이라는 것은 디자이너, 프로그래머, 기획자, 더 나아가 출시까지 고려하면 마케터 등 다양한 사람들이 모여 이루어진다. 그렇기에 기획자는 각 분야의 사람들과 협업하기 위하여 생각한 것들을 정리하여 자료로 만들어 **팀원들에게 공유**해야 한다. 이럴 때 필요한 것이 바로 플로우 차트이다.

플로우 차트의 필요성

- **플로우 차트**(Flow Chart)란 한국어로 말하면 '흐름도'다. 쉽게 말하자면 내가 표현하고자 하는 것을 하나의 흐름으로 표현하는 것이라고 생각하면 된다. 그렇다면 플로우 차트는 왜 필요한 것일까?
- 보통 게임 기획서는 대개 수 십장으로 구성이 된다. 그래서 디자이너, 프로그래머가 전달 받고 꼼꼼히 보고 제대로 이해하여 구현하기란 쉽지가 않다. 또한 기획자 본인도 자신의 기획서가 완벽하다고 자신하기란 쉽지 않다. 만약 팀원들이 기획서를 잘못 보거나 제대로 보지 않아 문제가 생긴다면 기획자에게도 책임이 있다고 볼 수 있다.
- 복잡한 시스템이나 컨텐츠를 기획하다 보면 언제나 예외사항이 생길 수 있고 고려하지 못하는 부분이 생길 수 있다. 그렇기 때문에 플로우 차트는 팀원 뿐만 아니라 자신이 기획서를 작성할 때 참고할 수 있는 **네비게이션과 같은 존재**가 된다.

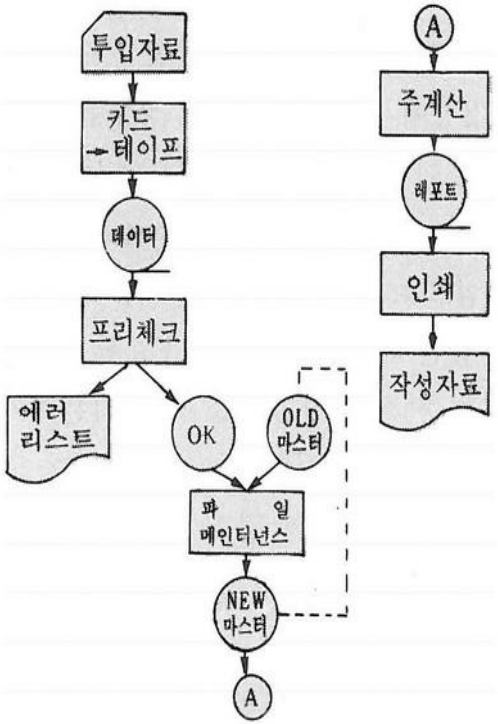
Ⅲ – ii. 플로우 차트 작성하기

기획 시작 단계

플로우 차트 작성하기

플로우 차트를 구성하는 방법

흐름도를 말한다. 문제의 범위를 정하며 분석하고, 그 해법을 명확하게 하기 위해서 필요한 작업이나 사무처리의 순서를 통일된 기호와 도형을 사용해서 도식적으로 표시한 것을 말한다. 프로그램에 관해서는 논리의 흐름을 특정한 기호를 사용해서 도식적으로 표현한 다이어그램 또는 블록 다이어그램이라고도 한다.



컨텐츠와 시스템을 나누어서

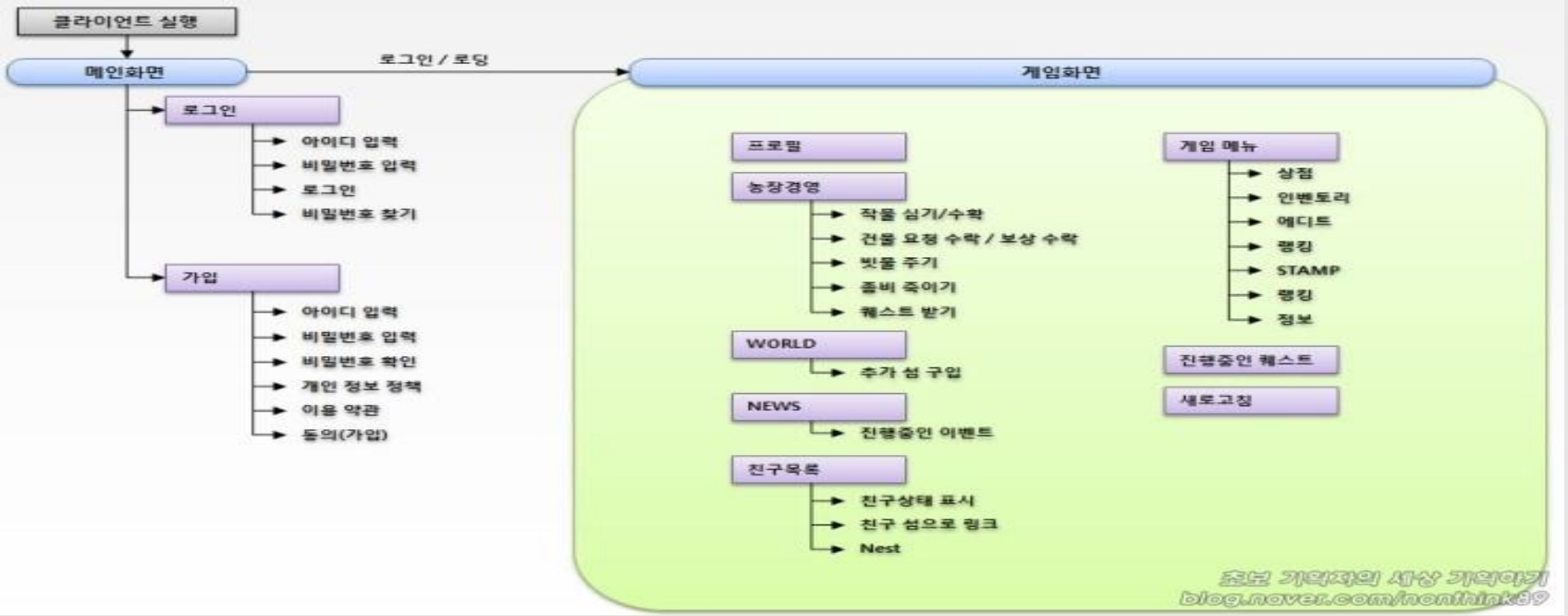
- 플로우 차트를 어떻게 구성해야 할까? 이는 기획자들마다 사용하는 방법이 다르겠지만, 대개 **컨텐츠**와 **시스템**을 나누어서 작성한다.
- 컨텐츠는 **화면 단위**로, 시스템은 마찬가지로 **화면이나 버튼 입력 단위**로 구성을 한다.

Ⅲ – ii. 플로우 차트 작성하기

기획 시작 단계

플로우 차트 작성하기

컨텐츠 플로우 차트



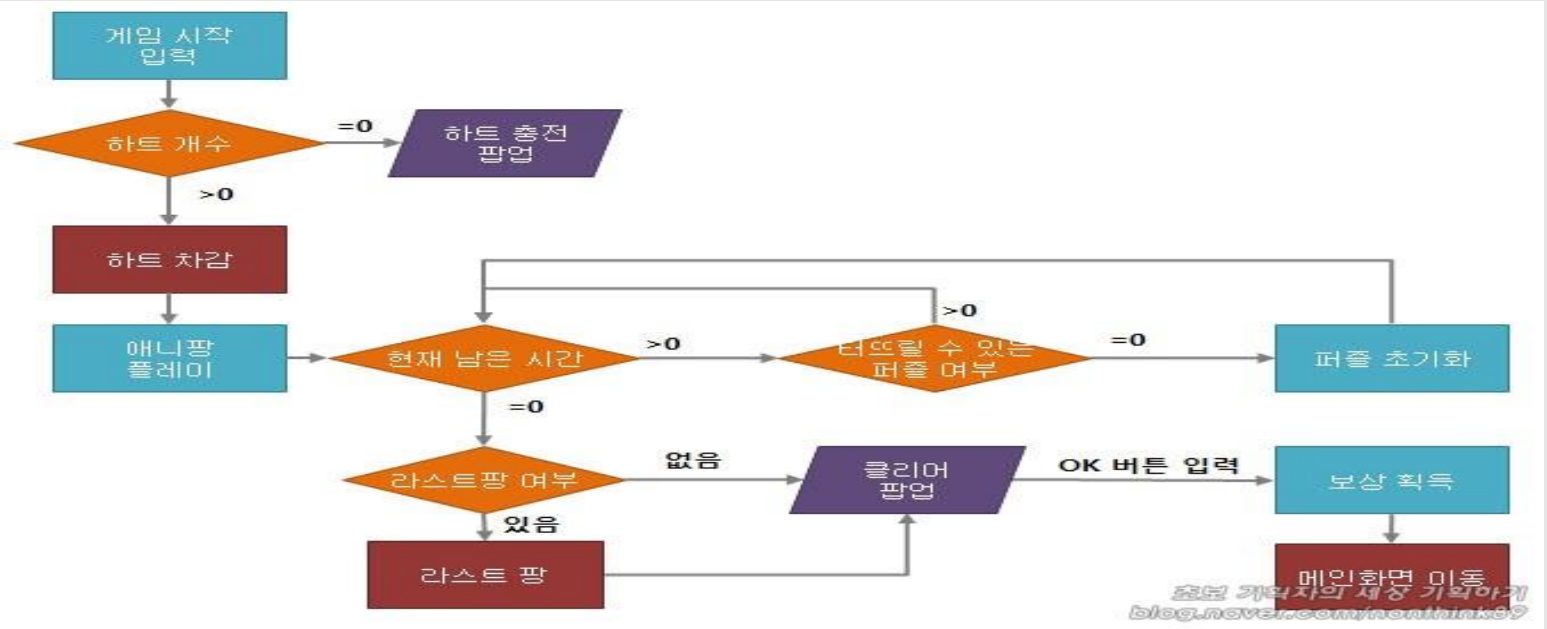
- 위의 이미지는 '룰 더 스카이'라는 스마트폰 게임의 분석 보고서 중 플로우 차트 부분을 가져온 것이다.
- 화면은 파란색, 버튼은 보라색, 내부 시스템에 대해서는 텍스트로만 표기하고 있다.
- 위와 같이 **게임 진행이 어떤 화면을 거쳐서 진행되는지** 알도록 작성해야 한다.
- 또한 각 화면에는 **어떤 컨텐츠들이 있는지**에 대해 기획자가 생각한 것들을 팀원들이 한 눈에 보고 이해하여 구현하기 쉽게 끔 작성해야 한다.

Ⅲ – ii. 플로우 차트 작성하기

기획 시작 단계

플로우 차트 작성하기

시스템 플로우 차트



- 위의 이미지는 '애니팡'이라는 스마트폰 게임의 시스템 플로우 차트를 작성해본 예시이다.
- 게임의 **전반적인 시스템 구성**과 **그 흐름**에 대해 이해할 수 있도록 작성해야 한다. 애니팡을 예로 들면, 게임 시작을 누르면 발생하는 재화의 변화, 그리고 게임 진행 시 시간 흐름에 따라 발생하는 것들, 끝날 때 까지 발생하는 것들을 흐름에 따라 표현해야 한다.
- 특히 시스템 플로우 차트의 경우 세부적인 시스템에 대해 예외 사항, 구현 사항을 하나하나 정리할 수 있어 제대로 작성해 놓아야 이후 세부 기획을 진행할 때 실수도 적어 지고, 프로그래머가 고민해야 할 부분이 적어진다.
- 프로그램 처리나 유저의 버튼 입력, 팝업, 분기 등에 따라 도형과 색을 다르게 써서 표현하는 등 최대한 팀원이 이해하기 쉽도록 작성해야 한다.


thank
you