캡스톤디자인(종합설계) 결과보고서

소속학부(과)	디지털콘텐츠공학과	팀명	사마일영
개설 연도 및 학기	2022학년도 ■1학기 □2학기	교과목명	캡스톤디자인2 (기업연계프로젝트)
과제명	메타버스 플랫폼		
과제유형	■기업연계형 캡스톤디자인	□기술이전형 캡스톤디자인	
희망금액		(기술이전금액)천원	

참여기업현황					
	기업	기업명		소재지	전주
		사업자번호		주요생산품목	응용 소프트웨어 개발 및 공급업
	담당자	성명		소속부서	
		H.P		E-mail	

기업연계 담당교수

참여 학생 현황	이름	학부(과)	학년	학번	H.P	E-mail
구분	이름	학부(과)	학년	학번	H.P	E-mail
팀장		디지털콘텐츠공학과	3	여		
팀원1		디지털콘텐츠공학과	4	여		
팀원2		디지털콘텐츠공학과	4	여		
팀원3		디지털콘텐츠공학과	4	여		
팀원4						
팀원5						
팀원6						

집행경비내역	비목	집행내역	금액
	재료비		0천원
	인쇄비		0천원
	학생여비	자세히 작성	
	학생회의비	(0)천원 × (4)인 × (0)회	0천원
			천원
	총액		0천원

위와 같이 캡스톤디자인(종합설계) 결과보고서를 제출합니다.

첨부 : 캡스톤디자인(종합설계) 과제 상세 결과보고서[별첨 1호]

2022년 6월 16일

지원학생(팀장) (인)

사업책임자(지도교수) (인)

참여기업 담당자 (인)

원광대학교 LINC 3.0 사업단장 귀하

캡스톤디자인(종합설계) 상세 결과보고서

서론

1-1 과제설계의 필요성

메타버스란, 앞으로의 상황이 사전에 프로그래밍한 다른 서비스와 다르게 이용자를 대표하는 아바타로 다른 이용자들과 소통하며 결정에 따라 달라지는 개방형 구조로, 현실에서 이뤄지던 활동이 가상의 환경에서 가능한 것뿐 아니라 현실에서는 불가능한 것까지 경험할 수 있는 서비스이다. 인간의 현실 공간을 중심으로 이뤄지던 활동이 가상의 환경에서 가능하게 되는 것에 주목하며, 새로운 또 하나의 가상공간을 형성하고 인간의 사유와 행동 방식을 변화할지 가능성에 주목하고 있다. 우리는 현실에서 주로 이루어지던 소통을 텍스트와 3D 화면을 통해 이루어지게 함으로써 온라인상의 친구는 물론 실제 친구까지 가상세계라는 공간에서 만날 수 있게 한다. 이러한 만남은 현실 공간에서만 할 수 있었던 일에서 벗어나 언제 어디서든 가능할 수 있게 만든다.

커스터마이징은 사용자 맞춤화 (user customization)와 개인화 (Personalization)로 구성된다. 사용자 맞춤화는 제품, 정보, 서비스 등을 제공하여, 개인의 행동 패턴이나 취향에 맞춰 사용자 스스로 환경을 설정 또는 변경할 수 있도록 하는 것이다. 개인화는 특정 사용자에 맞춰 밀접한 형태로 전용화하는 것을 의미한다. 선행연구에서 플레이어 스스로 캐릭터 커스터마이징을 통해 게임 캐릭터를 만든다는 것은 자기표현, 동일시, 자기개념으로 설명이라고 의미를 부여하고 있다.

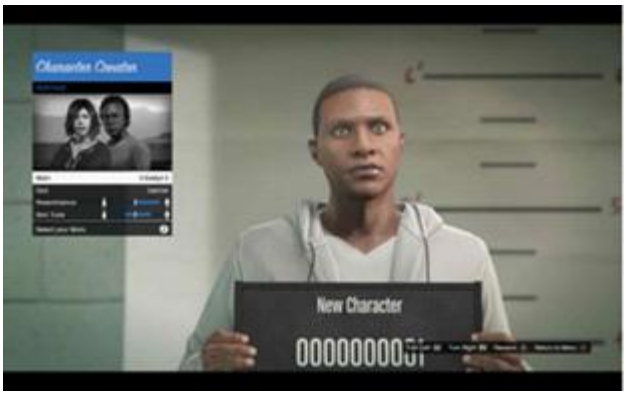


그림 1 GTA Customizing Scene

플레이어에 의해 생성된 캐릭터는 단순히 그래픽 데이터가 아니다. 플레이어 스스로 만든 게임 캐릭터는 가상세계 안에서 자신만의 정체성을 대변하는 아바타 (Avatar) 또는 플레이어 캐릭터이다. 스스로 만들고 꾸밈으로써 애착이 깊어진다. 또한, 게임 세계를 탐색하고 다른 플레이어와 동시적 커뮤니케이션을 수행하는 과정에서 마치 게임 세계에 있는 것 같다고 착각하게 만든다. 이러한 자신과 동일시하는 과정을 통해 플레이어는 더욱 몰입하게 된다.

이처럼 플레이어는 캐릭터 커스터마이징을 통해 자신을 표현하고자 하는 욕구가 강렬해진다. 또한, 플레이어의 의도대로 만들어진 캐릭터는 플레이어에게 동질감을 느끼게 한다. 이는 게임 수행 과정의 참여도를 증가시키는 중대한 기제로 작용한다. 플레이어의 성격을 반영할 여지가 큰 캐릭터 커스터마이징 시스템은 게임 분야에 없어서는 안 될 중요

한 시스템이며 앞으로의 게임 시장에서 더욱 중요한 역할을 하게 될 것이다.
위에서 보았던 내용과 같이 아바타는 자신의 자아 표출이며 더욱더 하고자 하는 것에 더욱 몰입하게 한다. 이런 점을 이용해 우리는 사용자가 메타버스라는 세계에 자신의 개성을 드러내고 몰입할 수 있도록 아바타를 커스터마이징 시킬 수 있고 더 나아가 자신만의 방을 꾸미기를 통해 나 자신을 온라인상에서 표현할 수 있게 된다.
나에 대한 어필이 중요해지는 시대에 맞추어 인터넷 공간에서 나를 표현하는 공간을 만들고자 한다.

1-2 선행연구 및 제품 관련 자료조사

1-2-1 로블록스(Roblox)

그림 2 로블록스(Roblox) 플레이 화면

2014년 설립된 로블록스는 이용자들이 레고처럼 생긴 아바타가 되어 가상세계에서 활동하는 게임이다. 다른 이용자와 함께 테마파크 건설 및 운영, 애완동물 입양, 스쿠버 다이빙, 슈퍼히어로 경험 등을 해볼 수 있다. 지난해 코로나19 사태로 등교를 못하게 된 미국 초등학생들이 다른 아이들과 소통할 수 있는 통로로 크게 인기를 얻었다. 로블록스는 미국에서 16세 미만 청소년의 55%가 가입했고, 하루 평균 접속자만 4,000만 명에 육박한다.
로블록스는 메타버스 산업의 대표주자로 꼽힌다. '가상(Meta)'과 '우주(Universe)'의 합성어인 메타버스는 온라인 속 3차원 입체 가상세계에서 아바타의 모습으로 구현된 개인들이 서로 소통하고, 돈을 벌고 소비하고, 놀이·업무를 하는 등 현실의 활동을 그대로 할 수 있는 플랫폼을 의미한다. 기존의 단순 가상현실(VR)보다 참여도가 높고 한 단계 진보한 개념이라는 평이다.
메타버스로 대표되는 리얼타임 콘텐츠 시장은 2019년 170억 달러(약 19조 원)에서 2022년에는 624억 달러(약 70조) 규모로 커질 것으로 전망된다. 로블록스 역시 지난해 매출이 전년보다 80% 넘게 증가하면서 1조 원을 돌파했고, 올해 매출은 2조2,000억 원이 넘을 것이라는 예상이 나온다.

1-2-2 제페토(ZEPETO)

그림 3 제페토(ZEPETO) 화면

제페토(ZEPETO)는 네이버 자회사인 네이버 Z가 운영하는 SNS 플랫폼이다. 증강현실을 기반으로 한 3D 아바타 앱이다. 제페토는 얼굴인식과 AR을 이용하여 아바타와 가상세계를 만드는데, 현재 전 세계가입자 수가 2억 명을 돌파하였다. 전체 이용자 중 80% 이상은 10대 청소년, 90% 이상이 외국인으로서, 다양한 연령층으로 확장을 꾀하고 있다. (이병권, 2021)

제페토는 3D 아바타를 기반으로 꿈꾸어 왔던 것들을 만들어 낼 수가 있는 가상세계 플랫폼이다. 상상하는 것들을 만들어내기 위하여 모바일 앱과 크리에이터, 빌더들의 플랫폼을 제공하고 있다. 독특한 3D 아이템과 아바타는 시각적으로 비즈니스 제품에 몰입할 수 있다. 제페토에서 세계를 창조하며 상상력에 생명을 불어넣을 수 있다. 자신이 살고 싶은 세계를 발견하거나 또 다른 자신을 만들 수 있다. 즉, 나만의 가상세계를 만드는 것이다. 제페토에서는 새로운 세계를 탐험하며 현실 세계에서는 경험할 수 없는 새로운 재미를 찾을 수 있고, 시간과 공간을 초월하는 가상세계에서 친구와 파트너 그리고 동료와의 새로운 관계를 구축할 수 있다. 또한, 제한 없이 세계에서 게임의 끝없는 다양성을 재생하며 다른 플레이어와 협력하거나 경쟁할 수 있다. 제페토 내에는 판매할 제품을 만들고 제품을 등록함으로써 수익을 창출할 수 있다. 제페토는 새로운 경험을 찾고 비용을 기꺼이 지출하는 고객과 소통하는데 알맞은 플랫폼이다. 이런 이유로 밀레니엄 세대와 Z 세대에 초점을 맞추었고, 인기를 얻을 수 있었다.

1-3 과제설계의 목표

메타버스에 대해 완벽하게 이해하고 플랫폼에 대한 이해도 또한, 높일 수 있도록 과제를 진행한다. 메타버스 플랫폼의 핵심 기능인 멀티플레이어 기능을 위해 사용하는 Photon에 대해 이해를 하고 코드를 작성하며 교수님과 멘토의 조언을 받고 코딩 실력을 키울 수 있도록 노력한다.

전주 한옥마을 경기전을 모델링함으로써 MAYA의 숙련도와 Substance Painter를 이용한 텍스쳐의 이해와 숙련도 또한 올리도록 한다. 숙련도뿐만 아니라 최소한의 폴리곤으로 만드는 로우폴리곤 오브젝트에 대한 이해도와 UV에 대한 이해도를 높이도록 한다.

팀 프로젝트를 진행하면서 실제 산업현장에서 부딪칠 수 있는 문제들을 학문 분야별로 습득한 전문 지식을 바탕으로 해결할 수 있는 능력을 기르도록 한다.

1-4 현실적 제한 요건

완벽한 시스템을 구축한 메타버스 플랫폼을 제작하기에는 3달이라는 시간과 4명밖에 없는 팀의 인원이 현실적으로 가장 큰 제한 요건이다.

메타버스 플랫폼은 일반 프로젝트보다 스케일이 큰 프로젝트이기 때문에 4명이서 3달 동안 열심히 작업을 진행하여도 일반 회사에서 제작된 메타버스 플랫폼보다는 퀄리티가 떨어질 것을 예상해야 하며 작업을 진행하면서 완벽하게 구현하지 못할 기능은 과감하게 빼는 결단력도 필요하다.

학교나 개인이 가지고 있는 컴퓨터로 작업을 진행하기에는 장비의 성능이 떨어진다는 제한 요건이 있다. 아무래도 학교에 있는 컴퓨터는 다수의 학생이 사용하다보니 성능이 떨어지고 처리 속도도 느려지게 된다. 노트북으로 3D 프로그램을 돌리는 것은 처리 속도가 많이 떨어진게 된다. 이런 성능이 좋지 못한 장비들로 제작을 진행하게 되면 프로젝트의 진행 속도가 눈에 띄게 느려지게 되고 3달이라는 시간 안에 해결하기에는 더욱 어려워진다.

1-5 작품의 특징 및 기대효과

젊은 세대에게 색다른 재미를 주는 포켓몬 빵 인기가 구매력을 갖춘 '어른이(어른+어린이)'들의 향수를 자극하고 있다가 또한, 온라인몰에서 포켓몬 빵 관련 굿즈 수요가 급증하고 있고 덩달아 복고풍 패션아이템까지 인기를 얻고 있다. 이런 포켓몬 빵의 재출시와 싸이월드 재오픈과 같은 사례를 보아 현재 뉴트로 감성이 주목받고 있다는 것을 알 수 있다.

메타버스는 개인 블로그와 비슷한 의미로 2차원에서 3차원으로 넘어왔다는 것이 일반 블로그들과의 차이점이다.

메타버스의 장점과 개인 블로그, 미니홈피의 감성을 합쳐 뉴트로 스타일의 메타버스 플랫폼을 제작하고자 한다. 자신의 아바타를 커스터마이징하고 개성에 맞게 옷을 입힐 수 있게 만든다. 이런 아바타들은 자신만의 메타버스 공간인 '마이홈(가명)'이라는 공간에서 지내게 되는데 여기 또한, 사용자가 마음대로 기본 아이템들을 가지고 꾸미거나 기본으로 제공되는 맵들을 입혀 생활할 수 있다. 자신만의 감성의 BGM과 상태 메시지까지 띄울 수 있어 자신의 자아를 표출하는데 좋은 공간이 될 것이다. 이러한 '마이홈(가명)'은 친구들이 방문할 수도 있으며 모르는 사람들도 내 메타버스 공간에 방문할 수 있다. 방문객들은 메타버스 공간의 주인에게 방명록과 채팅과 같은 다양한 형태의 소통을 할 수 있다.

그림 4 전주 한옥마을 사진

그림 5 천안 독립기념관 사진

더 나아가 뉴트로 스타일에서만 그치는 것이 아닌 한국을 알릴 수 있는 관광지들(전주 한옥마을, 천안의 독립기념관…)을 맵 꾸미기에 기본 맵으로 주어 사용자가 기본 오브젝트를 가지고 자신의 공간을 꾸밀 수 있는 것뿐만 아니라 만들어진 테마를 이용해서 새로운 장소를 체험하고 경험할 수 있게 된다.

그림 6 다른 메타버스 플랫폼 한옥마을 맵

가상세계 속 현실을 구현하는 메타버스는 현실의 정치, 경제, 사회를 가상공간에 옮겨놓거나, 새롭게 생성하여 가상공간에서 현실과 같지만 다른 체험을 가능하게 한다. 메타버스 맵과 커스터마이징한 아바타를 통해서 자신의 자아를 표출하게 된다. 이런 자아 표출은 자신의 개성을 찾고 내가 어떤 것을 좋아하고 싫어하는지를 알 수 있게 해주는 지표가 될 것이다.

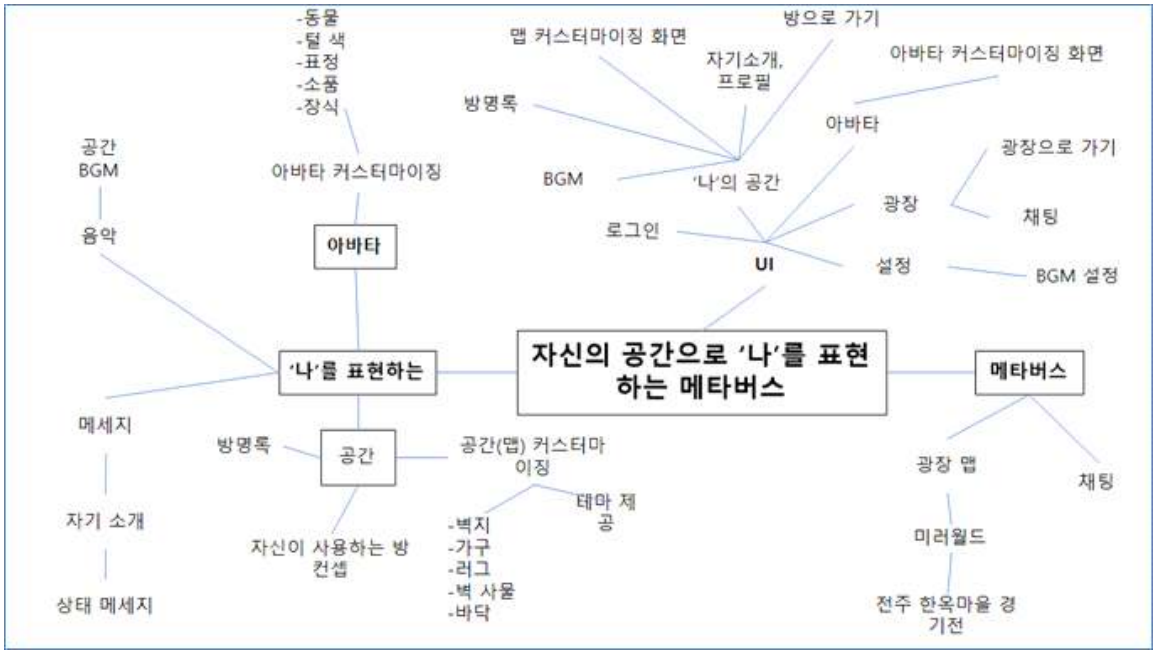
이런 메타버스 플랫폼은 현 사회의 새로운 형태의 소통의 공간이 되어질 것이다. 이러한 공간에서 이용자들은 물리적 접촉이 없는 환경일지라도 자신의 사회적, 공간적 존재감을 느낄 수 있는 실재감 (Presence)을 느끼고 개개인이 능동적으로 선택함으로써 보다 자유로운 형태의 소통이 가능할 것이다.

소통뿐만 아니라 현실세계에서 진행하던 사적 모임과 회의를 직접 만나지 않고 메타버스 안에서 해결할 수 있게 될 것이다.

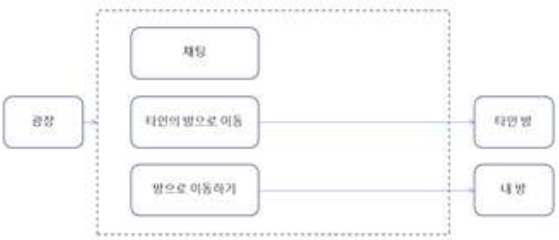
전주 한옥 마을 맵과 천안 독립기념관 맵을 가상으로 경험함으로써 직접 가보지 않고도 관광지를 즐길 수 있으며 실제로 체험해보고 싶은 사람들은 직접 관광지를 방문할 수 있다. 더 나아가 방문객이 늘어남으로써 지역 사회의 발전을 이룰 수 있을 것이다.

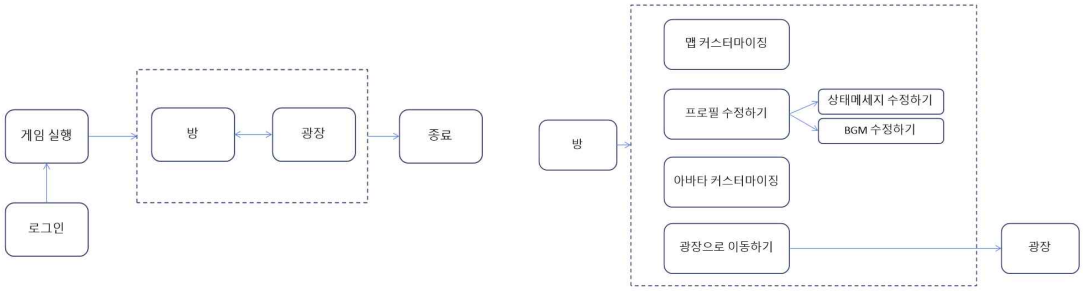
본론

2-1 문제 정의 및 아이디어 스케치



2-2 개념설계 등

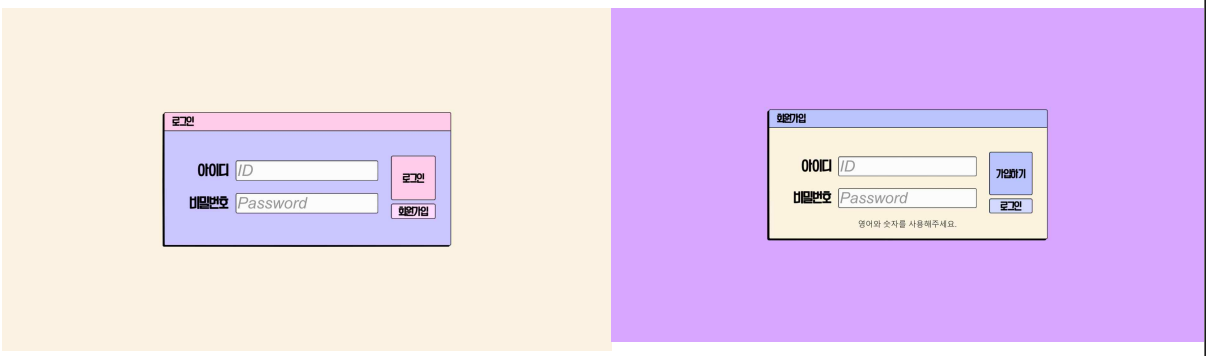




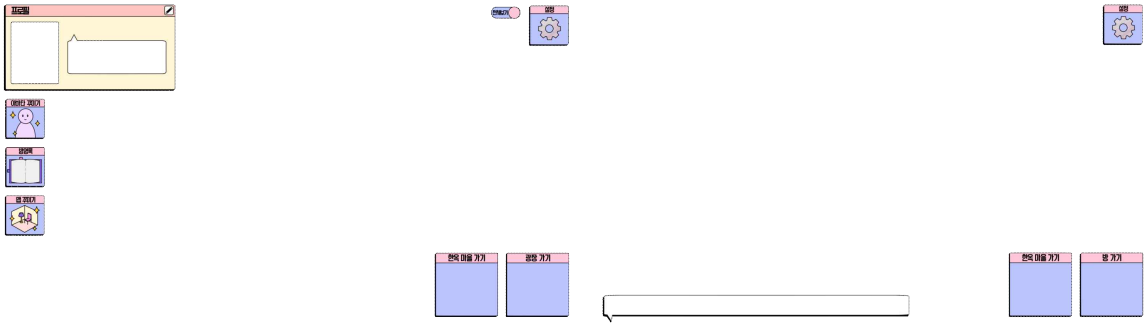
2-3 설계 제작 과정

UI UX

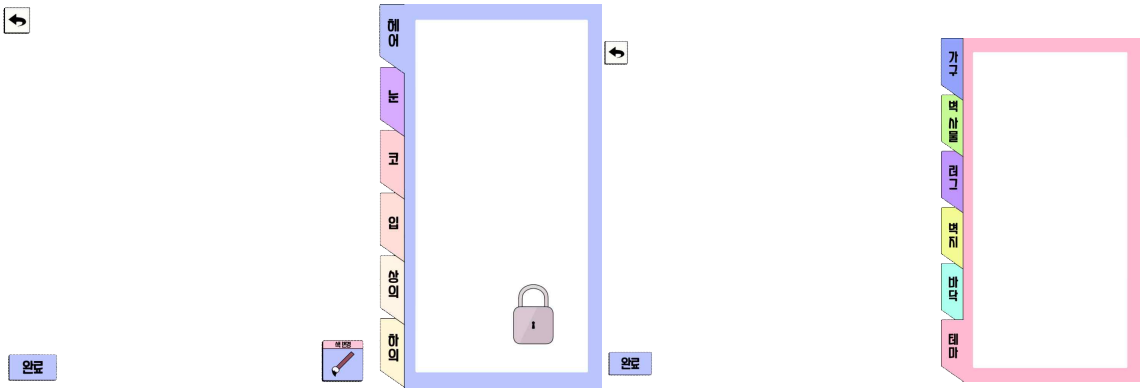
[로그인과 회원가입 UI]



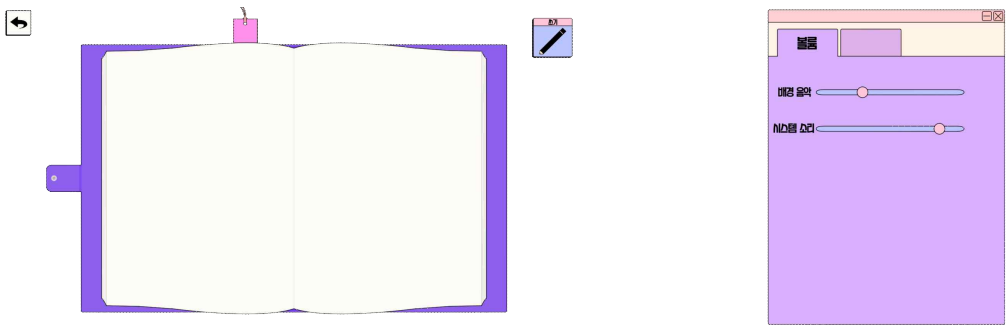
[메인 화면과 광장 UI]



[아바타 및 맵 커스터마이징 UI]



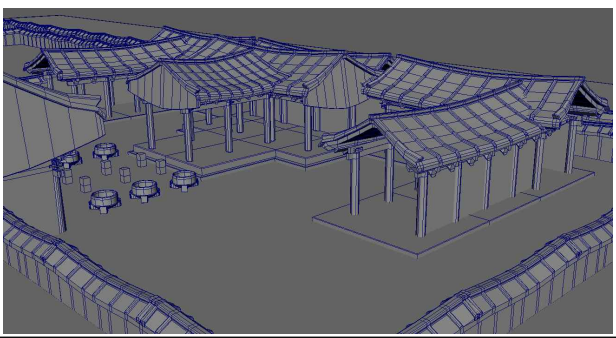
[방명록 및 설정창 UI]

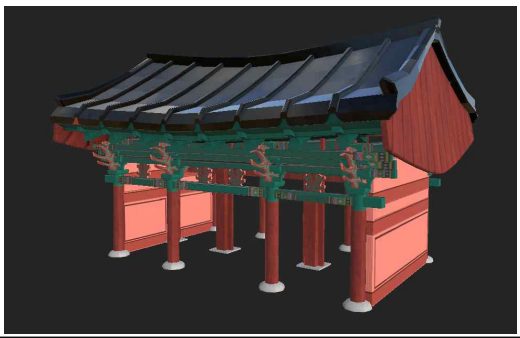


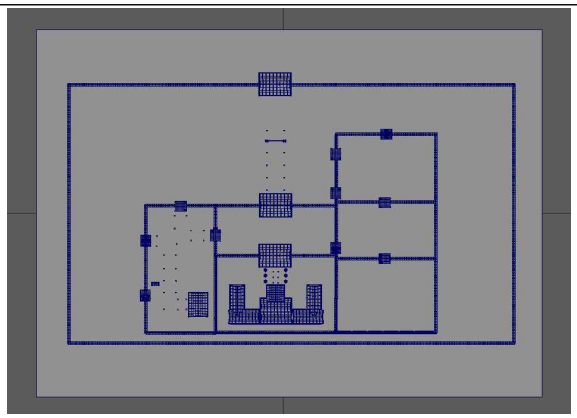
- 맵 구현

[광장의 배경이 될 한옥마을 경기전]

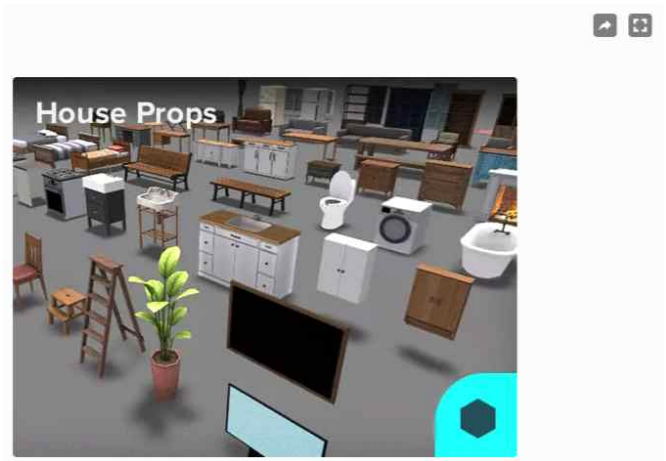
[경기전 모델링] - Maya와 Substance painter를 사용하여 구현

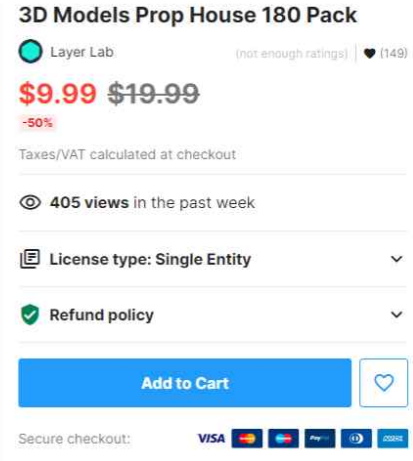






[메인 화면 및 플레이어 개인룸]





- 기능 구현

회원가입

입력값을 구글 스프레드시트에 저장

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using UnityEngine.SceneManagement;

public class GoogleSheetManager : MonoBehaviour
{
        const string URL = "구글 스프레드시트 주소 입력";
        public GoogleData GD;
        public InputField IDInput, PassInput, ValueInput;
        string id, pass;
        public Text RegistState;

        IEnumerable Start()
    {
                UnityWebRequest www = UnityWebRequest.Get(URL);
                yield return www.SendWebRequest();
                string data = www.downloadHandler.text;
                print(data);
    }

        bool SetIDPass()
        {
                id = IDInput.text.Trim();
                pass = PassInput.text.Trim();
                if (id == "" || pass == "") return false;
                else return true;
        }

        public void Register()
        {
                if (!SetIDPass())
                {
                        print("아이디 또는 비밀번호가 비어있습니다");
                        RegistStatement("아이디 또는 비밀번호가 비어있습니다");
                        return;
                }

                WWWForm form = new WWWForm();
                form.AddField("order", "register");
                form.AddField("id", id);
                form.AddField("pass", pass);
```

```
            StartCoroutine(Post(form));
        }

        public void RegistStatement(string regestState)
        {
            RegistState.text = regestState;
        }

        public void BackToLogin()
        {
            SceneManager.LoadScene("Login");
        }

        void OnApplicationQuit()
        {
            WWWForm form = new WWWForm();
            form.AddField("order", "logout");
            StartCoroutine(Post(form));
        }

        public void SetValue()
        {
            WWWForm form = new WWWForm();
            form.AddField("order", "setValue");
            form.AddField("value", ValueInput.text);

            StartCoroutine(Post(form));

        }

        public void GetValue()
        {
            WWWForm form = new WWWForm();
            form.AddField("order", "getValue");

            StartCoroutine(Post(form));
        }

        IEnumerator Post(WWWForm form)
        {
            using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
            // 반드시 using을 써야한다
            {
                yield return www.SendWebRequest();
```

```
                        if (www.isDone) Response(www.downloadHandler.text);
                        else print("웹의 응답이 없습니다.");
                        RegistStatement("웹의 응답이 없습니다.");
                }
        }

        void Response(string json)
        {
                if (string.IsNullOrEmpty(json)) return;

                GD = JsonUtility.FromJson<GoogleData>(json);

                if (GD.result == "ERROR")
                {
                        print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
                        return;
                }

                print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
                RegistStatement(GD.msg);
                if (GD.order == "getValue")
                {
                        ValueInput.text = GD.value;
                }
                if (GD.msg == "회원가입 완료")
                {
                        SceneManager.LoadScene("Login");

                }
        }

}
```

로그인

플레이어 아이디 값 저장 (Photon 방 이름으로 사용할 것)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Photon.Pun;
using Photon.Realtime;

namespace Com.MyCompany.MyGame
{
    /// <summary>
```

```
    /// Player name input field. Let the user input his name, will appear above the player in the game.
    /// </summary>
    [RequireComponent(typeof(InputField))]
    public class PlayerNameInputField : MonoBehaviour
    {
        #region Private Constants

        // Store the PlayerPref Key to avoid typos
        const string playerNamePrefKey = "PlayerName";

        #endregion


        #region MonoBehaviour CallBacks

        /// <summary>
        /// MonoBehaviour method called on GameObject by Unity during initialization phase.
        /// </summary>
        void Start()
        {
            string defaultName = string.Empty;
            InputField _inputField = this.GetComponent<InputField>();
            if (_inputField != null)
            {
                if (PlayerPrefs.HasKey(playerNamePrefKey))
                {
                    defaultName = PlayerPrefs.GetString(playerNamePrefKey);
                    _inputField.text = defaultName;
                }
            }
            PhotonNetwork.NickName = defaultName;
        }
        #endregion


        #region Public Methods

        /// <summary>
        /// Sets the name of the player, and save it in the PlayerPrefs for future sessions.
        /// </summary>
        /// <param name="value">The name of the Player</param>
        public void SetPlayerName(string value)
```

```
        {
            // #Important
            if (string.IsNullOrEmpty(value))
            {
                Debug.LogError("Player Name is null or empty");
                return;
            }
            PhotonNetwork.NickName = value;

            PlayerPrefs.SetString(playerNamePrefKey, value);
        }
        #endregion
    }
}
```

구글 스프레드시트에 저장된 값을 불러와 로그인

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;
using UnityEngine.SceneManagement;

[System.Serializable]
public class GoogleData
{
        public string order, result, msg, value;
}

public class GoogleLoginManager : MonoBehaviour
{
        const string URL = "구글 스프레드시트 주소 입력";
        public GoogleData GD;
        public InputField IDInput, PassInput, ValueInput;
        string id, pass;
        public static string NickName;
        public Text LoginState;
        public Network network;
        public void Awake()
        {
                Screen.SetResolution(3840, 2160, true);
                network.Connect();
                network.JoinLobby();
        }

        IEnumerable Start()
```

```
        {
            string a = "";
            LoginState.text = a.ToString();
            UnityWebRequest www = UnityWebRequest.Get(URL);
            yield return www.SendWebRequest();

            string data = www.downloadHandler.text;
            print(data);
        }

        bool SetIDPass()
        {
            id = IDInput.text.Trim();
            pass = PassInput.text.Trim();

            if (id == "" || pass == "") return false;
            else return true;
        }

        public void Register()
        {
            SceneManager.LoadScene("Regist");
        }

        public void Login()
        {
            if (!SetIDPass())
            {
                print("아이디 또는 비밀번호가 비어있습니다");
                LoginStatement("아이디 또는 비밀번호가 비어있습니다");
                return;
            }
            WWWForm form = new WWWForm();
            form.AddField("order", "login");
            form.AddField("id", id);
            form.AddField("pass", pass);

            StartCoroutine(Post(form));
            LoginStatement("로그인 중");
        }

        void OnApplicationQuit()
        {
            WWWForm form = new WWWForm();
            form.AddField("order", "logout");
```

```
        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone)
            {
                Response(www.downloadHandler.text);
            }
            else
            {
                print("웹의 응답이 없습니다.");
                LoginStatement("웹의 응답이 없습니다.");
            }
        }
    }

    public void LoginStatement(string loginstate)
    {
        LoginState.text = loginstate;
    }

    void logining()
    {
        NickName = id;
        network.JoinOrCreateRoom(NickName);

    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData>(json);
        LoginStatement("" + GD.msg);
        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
```

```
                }

                print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
                if (GD.msg == "로그인 완료")
                {
                        logining();
                        LoginStatement("로그인 완료");
                }
                else if (GD.msg == "로그인 실패")
                {
                        LoginStatement("로그인 실패");
                }
                if (GD.order == "getValue")
                {
                        ValueInput.text = GD.value;
                }
        }
}
```

Photon 서버 연결과 방 생성

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
using Photon.Realtime;
using UnityEngine.UI;

public class Network : MonoBehaviourPunCallbacks
{
    string NickName;
    int RoomPlace;

    private void Start()
    {
        RoomPlace = 0;
        PhotonNetwork.AutomaticallySyncScene = false;
        PhotonNetwork.ConnectUsingSettings();
        NickName = GoogleLoginManager.NickName;
    }

    public void Connect() => PhotonNetwork.ConnectUsingSettings();

    public override void OnConnectedToMaster()
    {
        print("서버접속완료");
```

```
        JoinLobby();
    }

    public void Disconnect() => PhotonNetwork.Disconnect();

    public override void OnDisconnected(DisconnectCause cause) => print("연결끊김");

    public void JoinLobby() => PhotonNetwork.JoinLobby();

    public override void OnJoinedLobby()
    {
        print("로비접속완료");
        if (RoomPlace == 2)
        {
            JoinPlaza();
        }
        else if (RoomPlace == 0)
        {
        }
        else if (RoomPlace == 1)
        {
            JoinOrCreateRoom(NickName);
        }
    }

    public void CreateRoom()
    {    // we check if we are connected or not, we join if we are , else we initiate the connection to the server.
        if (PhotonNetwork.IsConnectedAndReady)
        {
            // #Critical we need at this point to attempt joining a Random Room. If it fails, we'll get notified in OnJoinRandomFailed() and we'll create one.
            PhotonNetwork.CreateRoom(NickName,
                new RoomOptions { MaxPlayers = 10 });
        }
        else
        {
            // #Critical, we must first and foremost connect to Photon Online Server.
            PhotonNetwork.ConnectUsingSettings();
        }
    }

    public void JoinRoom(string ID)
    {    // we check if we are connected or not, we join if we are , else we initiate the connection to the server.
```

```
        if (PhotonNetwork.IsConnectedAndReady)
        {
            Debug.Log(PhotonNetwork.IsConnectedAndReady
                + " " + PhotonNetwork.IsConnected);
            // #Critical we need at this point to attempt joining a Random Room. If it fails, we'll get notified in OnJoinRandomFailed() and we'll create one.
            PhotonNetwork.JoinRoom(ID);
        }
        else
        {
            // #Critical, we must first and foremost connect to Photon Online Server.
            PhotonNetwork.ConnectUsingSettings();
        }
    }

    public void JoinOrCreateRoom(string ID)
    {
        Debug.Log(PhotonNetwork.IsConnectedAndReady
            + " " + PhotonNetwork.IsConnected);
        if (PhotonNetwork.IsConnectedAndReady)
        {
            // #Critical we need at this point to attempt joining a Random Room. If it fails, we'll get notified in OnJoinRandomFailed() and we'll create one.
            PhotonNetwork.JoinOrCreateRoom(ID,
                new RoomOptions { MaxPlayers = 6 }, null);
            PhotonNetwork.LoadLevel("Loading");
            print("방참가혹은만들기성공");
        }
        else
        {
            // #Critical, we must first and foremost connect to Photon Online Server.
            PhotonNetwork.ConnectUsingSettings();
            RoomPlace = 1;
        }
    }

    public void JoinPlaza()
    {
        if (PhotonNetwork.IsConnectedAndReady)
        {
            // #Critical we need at this point to attempt joining a Random Room. If it fails, we'll get notified in OnJoinRandomFailed() and we'll create one.
            PhotonNetwork.JoinOrCreateRoom("Plaza",
                new RoomOptions { MaxPlayers = 100 }, null);
            PhotonNetwork.LoadLevel("Plaza");
```

```
            print("방참가혹은만들기성공");
        }
        else
        {
            // #Critical, we must first and foremost connect to Photon Online Server.
            PhotonNetwork.ConnectUsingSettings();

            RoomPlace = 2;
        }
    }

    public void JoinRandomRoom() => PhotonNetwork.JoinRandomRoom();

    public void LeaveRoom()
    {
        PhotonNetwork.LeaveRoom();
        print("방 나가기 성공");
    }

    public override void OnCreatedRoom() => print("방만들기완료");

    public override void OnJoinedRoom()
    {
        print("방참가완료");
        PhotonNetwork.Instantiate("Player", Vector3.zero, Quaternion.identity);
    }

    public override void OnCreateRoomFailed(short returnCode, string message)
        => print("방만들기실패");

    public override void OnJoinRoomFailed(short returnCode, string message)
        => print("방참가실패");

    public override void OnJoinRandomFailed(short returnCode, string message)
        => print("방랜덤참가실패");

    public void CurrentRoom()
    {
        if (PhotonNetwork.InRoom)
        {
            print("" + PhotonNetwork.CurrentRoom.Name);
        }
    }

    void Instans()
```

```
    {
    }

    void OnPhotonInst(PhotonMessageInfo info)
    {
        info.Sender.TagObject = this.gameObject;
    }
}
```

메인 메뉴 - 프로필, 설정, 씬 이동, UI 숨기기

BGM 및 시스템 볼륨 조절

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class UISetting : MonoBehaviour
{
    [Header("설정창")]
    [SerializeField]
    GameObject Set_Cvs;

    [Header("배경음")]
    [SerializeField]
    Slider BGM_Slider;
    [SerializeField]
    AudioSource Great_Audio;
    [SerializeField]
    AudioSource Calm_Audio;
    [SerializeField]
    AudioSource Sorrow_Audio;

    [Header("시스템")]
    [SerializeField]
    Slider Sys_Slider;
    [SerializeField]
    AudioSource Sys_Audio;

    private float Vol = 1f;
    private float Vol_BGM;
    private float Vol_Sys;

    private void Start()
```

```
{
    Vol = PlayerPrefs.GetFloat("Vol");

    Vol_BGM = PlayerPrefs.GetFloat("BGM");
    Vol_Sys = PlayerPrefs.GetFloat("Sys");

    BGM_Slider.value = Vol_BGM;

    Great_Audio.volume = BGM_Slider.value;
    Calm_Audio.volume = BGM_Slider.value;
    Sorrow_Audio.volume = BGM_Slider.value;

    Sys_Slider.value = Vol_Sys;
    Sys_Audio.volume = Sys_Slider.value;
}

void Update()
{
    BGM_Bar();
    System_Bar();
}
// Start is called before the first frame update

public void Setting_Exit()
{
    if (Set_Cvs.activeSelf == true)
    {
        Set_Cvs.SetActive(false);
    }
    else
    {
        Set_Cvs.SetActive(true);
    }
}

public void BGM_Bar()
{
    Great_Audio.volume = BGM_Slider.value;
    Calm_Audio.volume = BGM_Slider.value;
    Sorrow_Audio.volume = BGM_Slider.value;

    Vol_BGM = BGM_Slider.value;
    PlayerPrefs.SetFloat("BGM", Vol_BGM);
}
```

```
    public void System_Bar()
    {
        Sys_Audio.volume = Sys_Slider.value;
        Vol_Sys = Sys_Slider.value;
        PlayerPrefs.SetFloat("Sys", Vol_Sys);
    }
}
```

메인메뉴 UI 설정 (프로필 창, BGM 설정, 씬 전환 버튼 설정, UI 숨기기)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.EventSystems;
using UnityEngine.SceneManagement;

public class UIMainMenu : MonoBehaviour
{
    public GameObject profile;
    public GameObject bgmEditor;
    public GameObject mainMenu;
    public InputField stateInput;
    public Text NickNameText;
    public Text NickNameMessage;
    public Text stateMessage;
    public Text BGMMessage;
    public Text BGMMainMenu;
    private bool UICanvas = true;
    public bool MainSetUI = true;
    public Button UICanvasBool;
    public Sprite Open;
    public Sprite Close;
    public GameObject Main;

    [Header("사운드")]
    public AudioSource Great;
    public AudioSource Calm;
    public AudioSource Sorrow;

    public Network network;

    void Awake()
    {
        Screen.SetResolution(3840, 2160, true);
        if (Loading.BGM == "Great")
```

```
        {
            Great.Play();
        }
        else if (Loading.BGM == "Calm")
        {
            Calm.Play();
        }
        else if (Loading.BGM == "Sorrow")
        {
            Sorrow.Play();
        }
        BGMMessage.text = Loading.BGM;
        BGMMainMenu.text = Loading.BGM;
    }

    void Start()
    {
        NickNameMessage.text = GoogleLoginManager.NickName;
        NickNameText.text = NickNameMessage.text;
    }

    public void OpenProfile()
    {
        MainSetUI = false;
        stateInput.text = stateMessage.text;

        profile.SetActive(true);

        bgmEditor.SetActive(false);
        Main.SetActive(false);
    }

    public void OpenMainMenu()
    {
        MainSetUI = true;
        Main.SetActive(true);
        profile.SetActive(false);

        bgmEditor.SetActive(false);
    }

    public void OpenAvatar()
    {
        SceneManager.LoadScene("Avatar");
    }
```

```
public void OpenGuestBook()
{
    SceneManager.LoadScene("Guest");
}

public void OpenMapAdditor()
{
    SceneManager.LoadScene("Map");
}

public void OpenPlaza()
{
    network.LeaveRoom();
    Load();
}

public void Load()
{
    network.JoinPlaza();
}

public void StateSave()
{
    stateMessage.text = stateInput.text;
}

public void BGMEditor()
{
    MainSetUI = false;
    MainSetUI = false;
    Main.SetActive(false);
    profile.SetActive(false);

    bgmEditor.SetActive(true);
}

public void BGMSetting1()
{
    MainSetUI = false;
    BGMMessage.text = EventSystem.current.currentSelectedGameObject.name;
    BGMMainMenu.text = BGMMessage.text;
    OpenProfile();
}
```

```
    public void UICanvasOpenAndDelete()
    {
        UICanvas = !UICanvas;
        if (UICanvas)
        {
            mainMenu.SetActive(true);
            UICanvasBool.image.sprite = Close;
        }
        else
        {
            mainMenu.SetActive(false);
            UICanvasBool.image.sprite = Open;
        }
    }

    public void Great_Sound()
    {
        Great.Play();
        Calm.Stop();
        Sorrow.Stop();
    }

    public void Calm_Sound()
    {
        Great.Stop();
        Calm.Play();
        Sorrow.Stop();
    }

    public void Sorrow_Sound()
    {
        Great.Stop();
        Calm.Stop();
        Sorrow.Play();
    }
}
```

아바타 움직임 구현

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;
```

```
using Photon.Realtime;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class Moving : MonoBehaviour
{
    [SerializeField]
    public float Speed;          // 움직이는 스피드.
    public Animator Animator;
    public PhotonView PhotonView;

    private Transform Vec;       // 카메라 벡터.
    private Vector3 MovePos;     // 플레이어 움직임에 대한 변수.

    void Init()
    {
        MovePos = Vector3.zero;
    }

    void Start()
    {
        Vec = GameObject.Find("CameraVector").transform;
        Init();
    }

    void Update()
    {
        if (PhotonView.IsMine == false) { return; }

        var a = Vector3.forward * Speed * ((Input.GetAxis("Vertical")
            * Input.GetAxis("Vertical")) + (Input.GetAxis("Horizontal")
            * Input.GetAxis("Horizontal")));
        Animator.SetFloat("Speed", a.magnitude);
        Run();
    }

    // 플레이어 움직임.
    void Run()
    {
        int ButtonDown = 0;
        if (Input.GetKey("a")) ButtonDown = 1;
        if (Input.GetKey("d")) ButtonDown = 1;
        if (Input.GetKey("w")) ButtonDown = 1;
        if (Input.GetKey("s")) ButtonDown = 1;
```

```
        // 플레이어가 움직임. 버튼에서 손을 땠을 때 Horizontal, Vertical이 0으로 돌아감으로써
        // 플레이어의 회전상태가 다시 원상태로 돌아가지 않게 하기 위해서.
        if (ButtonDown != 0)
            Rotation();
        else
            return;
        transform.Translate(Vector3.forward * Time.deltaTime * Speed * ButtonDown);
    }

    // 플레이어 회전.
    void Rotation()
    {
        MovePos.Set(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));    // 벡터 셋팅.

        Quaternion q = Quaternion.LookRotation(Vec.TransformDirection(MovePos));
        // 회전
        Debug.Log(q);
        if (MovePos != Vector3.zero)
        { transform.rotation = q; }
    }
}
```

카메라 위치

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

public class CameraMan : MonoBehaviour        //카메라 따라오는거
{
    public float MoveSpeed;      // 플레이어를 따라오는 카메라 맨의 스피드.
    public Transform[] Target;

    // 비공개
    // 플레이어의 트랜스 폼.
    private Vector3 Pos;          // 자신의 위치.

    // 플레이어를 따라다님.
    void Update()
    {
        Pos = transform.position;
        transform.position += (Target[Loading.Animal].position - Pos) * MoveSpeed;
    }
}
```

카메라 시야 회전

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Photon.Pun;

public class Axis : MonoBehaviour    // 화면 회전
{
    public Quaternion TargetRotation;  // 최종적으로 축적된 Gap이 이 변수에 저장됨.
    public Transform CameraVector;

    public float RotationSpeed;      //회전 스피드
    public GameObject Camera;
    public Transform MainCamera; // 카메라 컴포넌트.
    public float ZoomSpeed;
    public float Distance;            // 카메라와의 거리.
    public PhotonView PhotonView;
    // 비공개
    private Vector3 AxisVec;          // 축의 벡터.
    private Vector3 Gap;

    void Start()
    {
        if (PhotonView.IsMine)
        { }
        else
        {
            Camera.SetActive(false);
        }
        MainCamera = Camera.transform;
    }

    void Update()
    {
        if (PhotonView.IsMine == false) return;

        Zoom();
        CameraRotation();
    }

    void Zoom()
    {
        Distance += Input.GetAxis("Mouse ScrollWheel") * ZoomSpeed * -1;
        Distance = Mathf.Clamp(Distance, 1f, 3f);

        AxisVec = transform.forward * -1;
```

```
        AxisVec *= Distance;
        MainCamera.position = transform.position + AxisVec;
    }

    void CameraRotation()
    {
        if (transform.rotation != TargetRotation)
            transform.rotation = Quaternion.Slerp(transform.rotation,
                TargetRotation, RotationSpeed * Time.deltaTime);

        if (Input.GetMouseButton(1))
        {
            // 값을 축적.
            Gap.x += Input.GetAxis("Mouse Y") * RotationSpeed * -1;
            Gap.y += Input.GetAxis("Mouse X") * RotationSpeed;

            // 카메라 회전범위 제한.
            Gap.x = Mathf.Clamp(Gap.x, -5f, 85f);
            // 회전 값을 변수에 저장.
            TargetRotation = Quaternion.Euler(Gap);

            // 카메라벡터 객체에 Axis객체의 x,z회전 값을 제외한 y값만을 넘긴다.
            Quaternion q = TargetRotation;
            q.x = q.z = 0;
            CameraVector.transform.rotation = q;
        }
    }
}
```

아바타, 맵 커스터마이징

아바타 커스터마이징 씬 전환

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class UIAvatar : MonoBehaviour
{
    public Text ID;

    // Start is called before the first frame update
    private void Awake()
    {
```

```
        Screen.SetResolution(3840, 2160, true);
    }

    void Start()
    {
        ID.text = GoogleLoginManager.NickName;
    }

    public void OpenMainMenu()
    {
        SceneManager.LoadScene("Loading");
    }
}
```

아바타 및 맵 커스터마이징 (털색, 표정)

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

public class Mat : MonoBehaviour
{
    [SerializeField]
    Material[] materials;
    [SerializeField]
    Renderer rend;

    // Start is called before the first frame update
    void Start()
    {
        rend = GetComponent<Renderer>();
        rend.enabled = true;
    }

    public void Mat_(int i)
    {
        Debug.Log(i);
        rend.material = materials[i];
    }
}
```

아바타 커스터마이징 (소품, 몸 장식, 머리 장식)

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

```
public class Acc_Hand : MonoBehaviour
{
    [SerializeField]
    GameObject acc;

    public void Acc()
    {
        if (this.gameObject.activeSelf == false)
        {
            for (int i = 0; i < acc.transform.childCount; i++)
            {
                acc.transform.GetChild(i).gameObject.SetActive(false);
            }
            gameObject.SetActive(true);
        }
    }
}
```

맵 커스터마이징 씬 전환

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class UIMap : MonoBehaviour
{
     public void OpenMainMenu()
    {
        SceneManager.LoadScene("Loading");
    }
}
```

- 플레이어의 정보 저장 및 불러오기

플레이어 상태메세지 값 저장 (구글 스프레드시트)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData2
{
    public string order, result, msg, value;
```

```
}

public class State : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData2 GD;
    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", ValueInput.text);

        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
```

```
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData2>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            ValueInput.text = GD.value;
        }
    }
}
```

플레이어의 볼륨 설정값 저장 (구글 스프레드시트)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData3
{
    public string order, result, msg, value;
}

public class GetStateValue : MonoBehaviour
```

```
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData3 GD;
    public InputField IDInput, PassInput;
    public Text ID, ValueInput, ValueBGMInput;
    string id, pass;

    void Awake()
    {
        GetValue();
    }

    bool SetIDPass()
    {
        id = ID.text.Trim();
        pass = PassInput.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", ValueInput.text);

        StartCoroutine(Post(form));
    }


    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
```

```
    }

    public void GetBGMValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getBGMValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();
            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData3>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            ValueInput.text = GD.value;
        }
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (동물)

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData6
{
    public string order, result, msg, value;
}

public class AvatarAnimalState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData6 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);

        StartCoroutine(Post(form));
    }

    public void GetValue()
```

```
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData6>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (털색)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData8
{
    public string order, result, msg, value;
```

```
}

public class AvatarClrState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData8 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);

        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
```

```
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData8>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (표정)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData10
{
    public string order, result, msg, value;
}

public class AvatarFaceState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData10 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;
```

```
bool SetIDPass()
{
    id = ID.text.Trim();

    if (id == "" || pass == "") return false;
    else return true;
}

void OnApplicationQuit()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "logout");

    StartCoroutine(Post(form));
}

public void SetValue(string Animal)
{
    WWWForm form = new WWWForm();
    form.AddField("order", "setValue");
    form.AddField("value", Animal);

    StartCoroutine(Post(form));
}

public void GetValue()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "getValue");

    StartCoroutine(Post(form));
}

IEnumerator Post(WWWForm form)
{
    using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
    // 반드시 using을 써야한다
    {
        yield return www.SendWebRequest();

        if (www.isDone) Response(www.downloadHandler.text);
        else print("웹의 응답이 없습니다.");
    }
}
```

```
    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData10>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (소품)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData12
{
    public string order, result, msg, value;
}

public class AvatarAccHandState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData12 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }
```

```
void OnApplicationQuit()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "logout");

    StartCoroutine(Post(form));
}

public void SetValue(string Animal)
{
    WWWForm form = new WWWForm();
    form.AddField("order", "setValue");
    form.AddField("value", Animal);

    StartCoroutine(Post(form));
}

public void GetValue()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "getValue");

    StartCoroutine(Post(form));
}

IEnumerator Post(WWWForm form)
{
    using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
    // 반드시 using을 써야한다
    {
        yield return www.SendWebRequest();

        if (www.isDone) Response(www.downloadHandler.text);
        else print("웹의 응답이 없습니다.");
    }
}

void Response(string json)
{
    if (string.IsNullOrEmpty(json)) return;

    GD = JsonUtility.FromJson<GoogleData12>(json);

    if (GD.result == "ERROR")
    {
```

```
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (몸 장식)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData14
{
    public string order, result, msg, value;
}

public class AvatarAccBodyState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData14 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
```

```
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);

        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData14>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타 값을 구글 스프레드시트에 저장 (머리 장식)

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData16
{
    public string order, result, msg, value;
}

public class AvatarAccHeadState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData16 GD;

    public Text ID;
    string id, pass;

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);

        StartCoroutine(Post(form));
    }

    public void GetValue()
```

```
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData16>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
    }
}
```

아바타에 저장된 값 불러오기

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

public class MainMetStart : MonoBehaviour
{
    int AvatarState;
    int AvatarAccBodyState;
    int AvatarAccHandState;
    int AccHeadState;
```

```
public GameObject[] Animals;
public GameObject[] BearBodys;
public GameObject[] RabbitBodys;
public GameObject[] CatBodys;

public GameObject[] BearHands;
public GameObject[] RabbitHands;
public GameObject[] CatHands;

public GameObject[] BearHeads;
public GameObject[] RabbitHeads;
public GameObject[] CatHeads;

void Start()
{
    AvatarState = Loading.Animal;
    AvatarAccBodyState = Loading.Body;
    AvatarAccHandState = Loading.Hand;
    AccHeadState = Loading.Head;
    Animal();
    Body();
    Hand();
    Head();
}

void Animal()
{
    for (int i = 0; i < Animals.Length; i++)
    {
        Animals[i].SetActive(false);
    }
    Animals[AvatarState].SetActive(true);
}
void Body()
{
    for (int i = 0; i < BearBodys.Length; i++)
    {
        BearBodys[i].SetActive(false);
    }
    for (int i = 0; i < RabbitBodys.Length; i++)
    {
        RabbitBodys[i].SetActive(false);
    }
    for (int i = 0; i < CatBodys.Length; i++)
```

```
        {
            CatBodys[i].SetActive(false);
        }
        if (AvatarState == 0)
        {
            BearBodys[AvatarAccBodyState].SetActive(true);
        }
        else if (AvatarState == 1)
        {
            RabbitBodys[AvatarAccBodyState].SetActive(true);
        }
        else if (AvatarState == 2)
        {
            CatBodys[AvatarAccBodyState].SetActive(true);
        }
    }

    void Hand()
    {
        for (int i = 0; i < BearHands.Length; i++)
        {
            BearHands[i].SetActive(false);
        }
        for (int i = 0; i < RabbitHands.Length; i++)
        {
            RabbitHands[i].SetActive(false);
        }
        for (int i = 0; i < CatHands.Length; i++)
        {
            CatHands[i].SetActive(false);
        }
        if (AvatarState == 0)
        {
            BearHands[AvatarAccHandState].SetActive(true);
        }
        else if (AvatarState == 1)
        {
            RabbitHands[AvatarAccHandState].SetActive(true);
        }
        else if (AvatarState == 2)
        {
            CatHands[AvatarAccHandState].SetActive(true);
        }
    }
```

```
    void Head()
    {
        for (int i = 0; i < BearHeads.Length; i++)
        {
            BearHeads[i].SetActive(false);
        }
        for (int i = 0; i < RabbitHeads.Length; i++)
        {
            RabbitHeads[i].SetActive(false);
        }
        for (int i = 0; i < CatHeads.Length; i++)
        {
            CatHeads[i].SetActive(false);
        }
        if (AvatarState == 0)
        {
            BearHeads[AccHeadState].SetActive(true);
        }
        else if (AvatarState == 1)
        {
            RabbitHeads[AccHeadState].SetActive(true);
        }
        else if (AvatarState == 2)
        {
            CatHeads[AccHeadState].SetActive(true);
        }
    }
}
```

구글 스프레드시트에 저장된 아바타 값 불러오기 (동물)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData7
{
    public string order, result, msg, value;
}

public class GetAvatarAnimalState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
```

```
public GoogleData7 GD;

public Text ID;
string id, pass;
public Loading Loading;
public string AvatarAnimalStateNum;

private void Start()
{
    GetValue();
}

bool SetIDPass()
{
    id = ID.text.Trim();

    if (id == "" || pass == "") return false;
    else return true;
}

void OnApplicationQuit()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "logout");

    StartCoroutine(Post(form));
}

public void SetValue(string Animal)
{
    WWWForm form = new WWWForm();
    form.AddField("order", "setValue");
    form.AddField("value", Animal);

    StartCoroutine(Post(form));
}

public void GetValue()
{
    WWWForm form = new WWWForm();
    form.AddField("order", "getValue");

    StartCoroutine(Post(form));
}
```

```
    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData7>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }
        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            AvatarAnimalStateNum = GD.value;
            Loading.Animal = int.Parse(AvatarAnimalStateNum.ToString());
            Loading.LoadingScore += 1;
            print("loading" + Loading.LoadingScore);
        }
    }
}
```

구글 스프레드시트에 저장된 아바타 값 불러오기 (털색)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData9
{
```

```
    public string order, result, msg, value;
}

public class GetAvatarClrState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData9 GD;
    //    public InputField IDInput, PassInput, ValueInput;
    public Text ID;
    string id, pass;
    public Loading Loading;
    public string AvatarClrStateNum;
    private void Start()
    {
        GetValue();
    }

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);

        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
```

```
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData9>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            AvatarClrStateNum = GD.value;
            Loading.Clr = int.Parse(AvatarClrStateNum.ToString());
            Loading.LoadingScore += 1;
            print("loading" + Loading.LoadingScore);
        }
    }
}
```

구글 스프레드시트에 저장된 아바타 값 불러오기 (표정)

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

```
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData11
{
    public string order, result, msg, value;
}

public class GetAvatarFaceState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData11 GD;

    public Text ID;
    string id, pass;
    public string AvatarFaceStateNum;
    public Loading Loading;

    private void Start()
    {
        GetValue();
    }
    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", Animal);
```

```
        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }


    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData11>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            AvatarFaceStateNum = GD.value;
            Loading.Face = int.Parse(AvatarFaceStateNum.ToString());
            Loading.LoadingScore += 1;
            print("loading" + Loading.LoadingScore);
        }
```

```
    }
}
```

구글 스프레드시트에 저장된 방값 불러오기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData21
{
    public string order, result, msg, value;
}

public class GetRoomState : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData21 GD;

    public Text ID;
    string id, pass;
    public string RoomNum;
    public Loading Loading;

    private void Start()
    {
        GetValue();
    }

    bool SetIDPass()
    {
        id = ID.text.Trim();

        if (id == "" || pass == "") return false;
        else return true;
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
```

```
    }

    public void SetValue(string Animal)
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "setValue");
        form.AddField("value", RoomNum);

        StartCoroutine(Post(form));
    }

    public void GetValue()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData21>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);
        if (GD.order == "getValue")
```

```
        {
            RoomNum = GD.value;
            Loading.Room = int.Parse(RoomNum);
            Debug.Log(Loading.Room);
            Loading.LoadingScore += 1;
            print("loading" + Loading.LoadingScore);
        }
    }
}
```

구글 스프레드시트에 저장된 BGM 값 불러오기

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

[System.Serializable]
public class GoogleData5
{
    public string order, result, msg, value;
}

public class GetBGMStateValue : MonoBehaviour
{
    const string URL = "구글 스프레드시트 주소 입력";
    public GoogleData5 GD;
    public string BGM;
    public Loading Loading;

    private void Start()
    {
        GetBGMValue();
    }

    void OnApplicationQuit()
    {
        WWWForm form = new WWWForm();
        form.AddField("order", "logout");

        StartCoroutine(Post(form));
    }

    public void GetBGMValue()
    {
```

```
        WWWForm form = new WWWForm();
        form.AddField("order", "getValue");

        StartCoroutine(Post(form));
    }

    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(URL, form))
        // 반드시 using을 써야한다
        {
            yield return www.SendWebRequest();

            if (www.isDone) Response(www.downloadHandler.text);
            else print("웹의 응답이 없습니다.");
        }
    }

    void Response(string json)
    {
        if (string.IsNullOrEmpty(json)) return;

        GD = JsonUtility.FromJson<GoogleData5>(json);

        if (GD.result == "ERROR")
        {
            print(GD.order + "을 실행할 수 없습니다. 에러 메시지 : " + GD.msg);
            return;
        }

        print(GD.order + "을 실행했습니다. 메시지 : " + GD.msg);

        if (GD.order == "getValue")
        {
            BGM = GD.value;
            Loading.BGM = BGM;
            Loading.LoadingScore += 1;
            print("loading" + Loading.LoadingScore);
        }
    }
}
```

Get(-)State에서 값 받아오기

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

```
using UnityEngine;
using UnityEngine.SceneManagement;

public class Loading : MonoBehaviour
{
    public int LoadingScore;
    public static int Animal;
    public static int Face;
    public static int Clr;
    public static int Head;
    public static int Hand;
    public static int Body;
    public static int Room;
    public static string BGM;

    void Start()
    {
        LoadingScore = 0;
        Animal = 0;
        Face = 0;
        Clr = 0;
        Head = 0;
        Hand = 0;
        Body = 0;
        Room = 0;
        BGM = "Great";
    }

    void Update()
    {
        if (LoadingScore == 8)
        {
            Debug.Log(Room);
            SceneManager.LoadScene("MainMenu");
        }
    }
}
```

- 멀티서버 (Photon)

- 광장 (한옥마을)

광장 씬 UI 설정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class UIPlaza : MonoBehaviour
{
    public GameObject MainCanvas;
    private bool UICanvas = true;
    public Button UICanvasBool;
    public Sprite Open;
    public Sprite Close;
    public Network network;
    string NickName;
    private void Awake()
    {

        Screen.SetResolution(3840, 2160, true);
    }

    void Start()
    {
        NickName = GoogleLoginManager.NickName;
    }

    public void OpenMainMenu()
    {
        network.LeaveRoom();
        Loading();
    }

    public void Loading()
    {
        network.JoinOrCreateRoom(NickName);
    }

    public void OpenSetting()
    {
        SceneManager.LoadScene("Setting");
    }

    public void UICanvasOpenAndDelete()
    {
        UICanvas = !UICanvas;
```

```
            if (UICanvas)
            {
                MainCanvas.SetActive(true);
                UICanvasBool.image.sprite = Close;
            }
            else
            {
                MainCanvas.SetActive(false);
                UICanvasBool.image.sprite = Open;
            }
        }
}
```

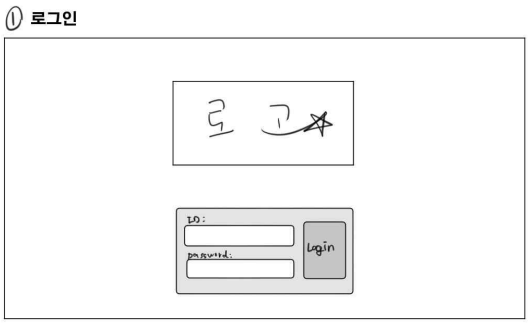
채팅 기능

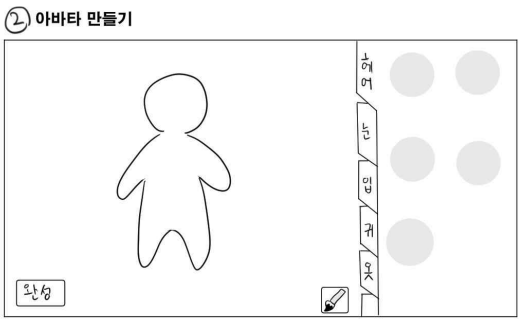
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

public class GoogleChatManager : MonoBehaviour
{
    const string URL = "채팅 기록 저장 주소“;
    const string WebURL = "구글 스프레드시트 주소 입력";

    public Text ChatText;
    public InputField NicknameInput, ChatInput;

    void Start()
    {
#if !UNITY_ANDROID
        Screen.SetResolution(960, 540, false);
#endif
        StartCoroutine(Get());
    }

    IEnumerator Get()
    {
        UnityWebRequest www = UnityWebRequest.Get(URL);
        yield return www.SendWebRequest();

        string data = www.downloadHandler.text;
        ChatText.text = data;

        StartCoroutine(Get());
```

```
    }

    public void ChatPost()
    {

        WWWForm form = new WWWForm();
        form.AddField("nickname", GoogleLoginManager.NickName);
        form.AddField("chat", ChatInput.text);

        StartCoroutine(Post(form));
    }


    IEnumerator Post(WWWForm form)
    {
        using (UnityWebRequest www = UnityWebRequest.Post(WebURL, form))
        {  // 반드시 using을 써야한다
            yield return www.SendWebRequest();
        }
    }
}
```

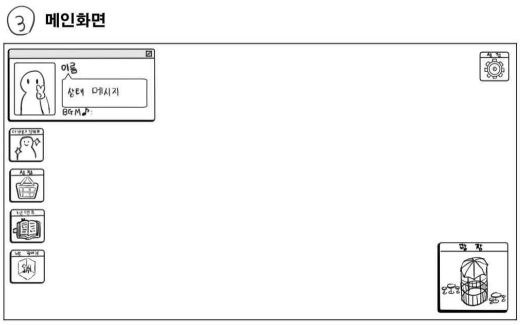
결론

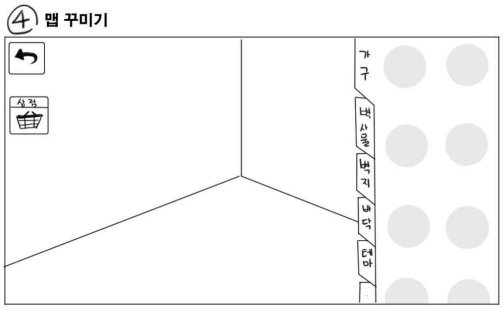
3-1 설계보완점 및 목표구현 정도





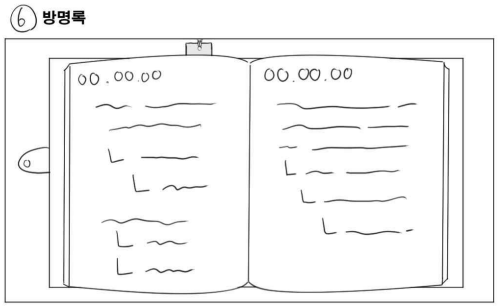
사용자로부터 아이디와 패스워드를 부여하여 개개인의 계정을 만든다. 이러한 개인 데이터들을 데이터베이스로 관리하게 된다. 각자 만들어진 계정에서 자신의 개성을 표출할 수 있는 아바타를 커스터마이징할 수 있다. 머리와 눈, 입, 귀 등의 파츠를 따로따로 제작해 여러 가지의 아바타 제작이 가능하다.

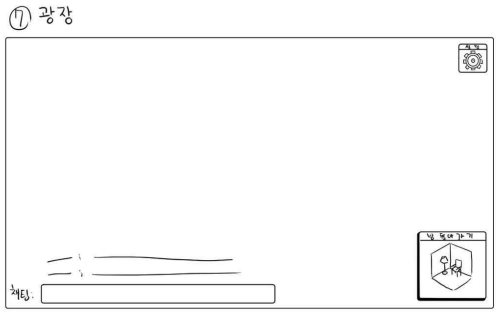




아바타를 커스터마이징한 후 자신의 메타버스 공간에 들어갈 수 있다. 이 공간 또한, 자신이 원하는대로 기본 오브젝트를 가지고 꾸밀 수 있다. 기본적으로 꾸며진 맵(한옥마을 등)을 제공하여 사용자의 편의성을 높인다. 개인적인 메타버스 공간에 친구를 초대하여 소통할 수 있고 모르는 사람이 자신의 메타버스를 구경할 수도 있다.

방문한 사람들은 방명록에 기록이 가능하다. 여기서 사람들은 서로의 안부를 묻거나 새로운 친구를 사귈 수 있게 된다.





광장 맵에 들어가게 되면 더 넓은 범위의 모르는 사람들과 만날 수 있다. 예쁜 광장 맵에서 사진을 찍고 채팅을 나누는 등의 소통의 공간이 될 수 있다.

3-2 완성작품 사진

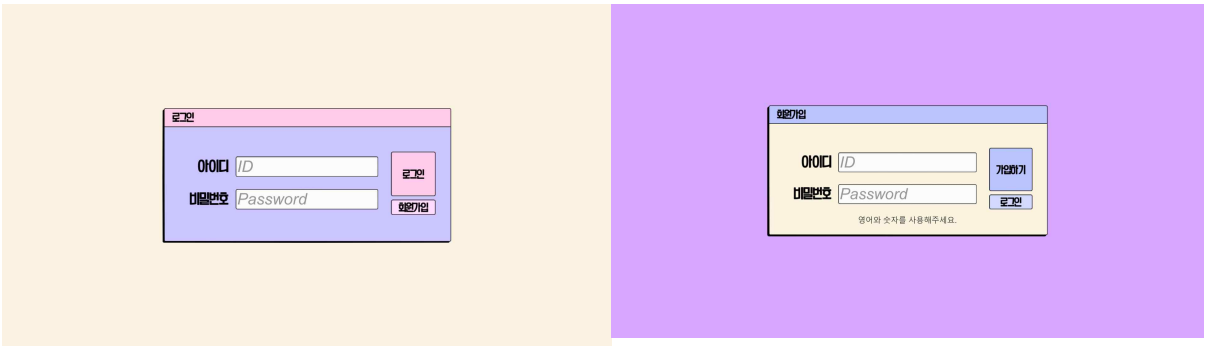


그림 29 로그인 화면과 회원가입 화면

아이디가 있다면 로그인을 진행하여 자신의 방으로 이동하고 아이디가 없다면 회원가입을 한 후 로그인을 진행해 자신의 방으로 이동할 수 있다.

그림 31 프로필 변경 화면

그림 32 자신의 방 화면

자신의 방에서는 상태메세지 및 배경 음악을 변경할 수 있디.
자신의 방과 아바타를 커스터마이징 할 수 있다.
현재 한옥 마을을 테마로 하고 있는 광장 맵으로도 이동하여 다른 유저들을 보고 소통할 수 있다.

그림 33 광장 맴 화면 (전주 한옥마을 경기전)

광장에선 자신의 방으로 돌아가기. 채팅을 이용하여 다른 사람들과 소통하기를 할 수 있으며, UI 끄고 키기, 볼륨 조절을 할 수 있다.

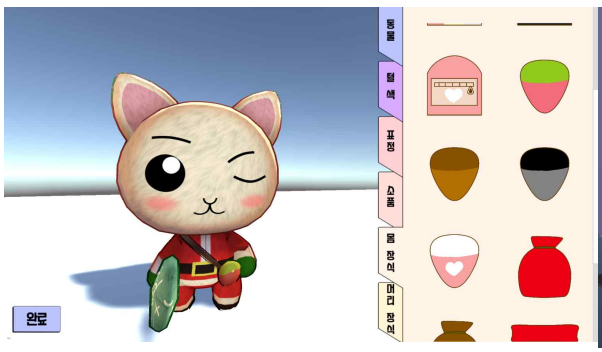


그림 34 아바타 커스터마이징 화면

그림 35 맵 커스터마이징 화면

자신의 아바타를 커스터마이징 하여 저장할 수 있다. 동물 캐릭터와 털 색, 표정과 소품 등을 골라 적용할 수 있다. 맵 커스터마이징에서는 벽지 색깔을 선택하여 저장하고 자신의 방에 적용할 수 있다.

3-3 향후 개선사항

그림 32 서로 다른 계정에서 동시에 찍은 사진

[그림 16]에서 보다시피 타 플레이어의 아바타도 자신의 아바타 값을 받아와서 서로 다른 계정이 같아보인다. 아바타 프리펩의 값을 아바타 주인 이름에 해당하는 값을 받아와 적용시켜 개선해야 한다. 이는 서버 접속 후 타인의 아바타를 볼 수 있을 뿐 아니라, 타인의 움직임 또한 정밀하게 볼 수 있게 한다.

4-1 출처

- John Smart, Jamais Cascio, Jerry Paffendorf, METAVERSE ROADMAP, 2007.06, https://www.w3.org/2008/WebVideo/Annotations/wiki/images/1/19/MetaverseRoadmapOverview.pdf, (접속일자 2021년 12월 7일.)
- 피에르 레비, 전재연 역,《디지털 시대의 가상현실》, 궁리, 2002.
- 주수완. "메타버스 플랫폼 사용자 경험 연구." 국내석사학위논문 한양대학교, 2022. 서울
- 메타버스 플랫폼 현황과 전망 한상열 | 소프트웨어정책연구소 선임연구원
- 손수정(Soo J Sohn). "메타버스(Metaverse) 플랫폼 기반 Co-creation 활성화를 위한 지식재산 이슈." STEPI Insight -.282 (2021): 1-46.
- Matthew Ball, The Metaverse: What It Is, Where to Find it, Who Will Build It, and Fortnite, 2020.1.13, https://www.matthewball.vc/all/themetaverse, (최종검색일 2021년 11월 22일.)
- 김상균,《메타버스 새로운 기회》, 베가북스, 2021.
- 류철균, 안진경,〈가상세계의 디지털 스토리텔링 연구-<세컨드라이프>와 MMORPG의 비교를 중심으 로〉,《게임산업저널》통권 16호, 한국콘텐츠진흥원, 2007.
- David Bell eds, Cyber culture, London : Routledge, 2004, p50
- 이재현,《인터넷과 온라인 게임》, 커뮤니케이션북스, 2003, pp215-216.
- 박찬익,〈게임 캐릭터의 조형적 특성에 따른 플레이어의 몰입도에 관한 연구〉,《디지털융복합연 구》제18권, 한국디지털정책학회, 2020.
- Smart, J., Cascio, J. & Paffendorf, J. (2007). Metaverse roadmap overview. Available: http://www.metaverseroadmap.org/ overview
- 대한민국 정책브리핑 가상세계와 현실 넘나들다...‘메타버스’ 열풍 최선영 (2021.08.04) https://www.korea.kr/news/policyNewsView.do?newsId=148891141
- Pwc 프라이스 워터 하우스 메타버스 전망

- 최은진, & 이영숙. (2021). 메타버스 플랫폼을 활용한 민화 미술관 기획 연구-제페토 사례를 중심으로. 한국게임학회 논문지, 21(6), 63-74.
- Lee Jihyun, Generation Z. "My avatar is Gucci". D2A, which has grown to a 50 trillion market, Hankyung.com, May 27, 2021
- 전자부품 전문 미디어 디일렉지난해 성장 주춤한 '3D 센싱 시장' 내후년부터 회복세이나리 기자 (2021.06.21 17:23) http://www.thelec.kr/news/articleView.html?idxno=12928
- MWC 2019 Preview 유지투자증권 사업분석 2019. 02.21
- 테크월드뉴스 생활TECH] 내 얼굴이 암호가 된다, '아이폰의 Face ID’ 김지윤 기자 (2019.04.08 09:09) http://www.epnc.co.kr/news/articleView.html?idxno=83111
- 게임 캐릭터 커스터마이징 디자인요소의 중요도 연구: 모바일 게임과 비디오 게임 중심으로 (논문 2020. 남기덕, 길태숙)
- 동작 인식 게임의 융합 발전 방향 이면재 백석대학교 정보통신학부
- Kiteok Nam, “A Study on the Personality Design of Game Characters using AHP”, Doctor’s Theses, Sangmyung University, pp.2, 2020.
- Suhyun Song, “An Analysis on the Relation between MMORPG [Aion] Player Types and Character Customizing”, Master’s Theses, Hongik University, pp.21-30, 2009.
- Cheolho Baek, “Researching the gender - recognition changes and gender differences in MMORPG avatar” Journal of Korea Digital Design Society, Vol. 11, No. 3, pp.355-363, 2011.
- Suhyun Song, “An Analysis on the Relation between MMORPG [Aion] Player Types and Character Customizing”, Master’s Theses, Hongik University, pp.21-30, 2009.
- O'Brien, L., & Murnane, J., “An investigation into how avatar appearance can affect interactions in a virtual world” International Journal of Social and Humanistic Computing, Vol. 1, No. 2, pp.192-202, 2009.
- Kihyun Yang, “A study on user flow factors and loyalty of MMORPG avatar customization”, Master’s Theses, Hongik University, pp.18-28, 2012.
- Kyunghee Kang, “User segmentation research for mobile phone user interface customization”, Master’s Theses, Ewha Womans University, pp.38-50, 2008.
- Oatley, K., “A taxonomy of the emotions of literary response and a theory of identification in fictional narrative” Poetics, Vol. 23, No. 1, pp.53-74, 1995.
- Hefner, D., Klimmt, C., & Vorderer, P., “Identification with the player character as determinant of video game enjoyment” In Entertainment Computing.ICEC 2007, Springer Berlin Heidelberg, pp.39-48, 2007.
- Miyoon Kim, “The study of people character design shown in domestic online game”, Master’s Theses, Yeungnam University, pp.19, 2006.
- Sukho Jung, “Character Customizing System Based on Physiognomy”, Master’s Theses, Kongju National University, pp.5-9, 2011.

- Danlu Zhou, “A Study of the Relevance of Gender Identity and Character Customizing among Female Players in Massive Multiplay Online Role-Playing Games: Focused on the Game”, Master’s Theses, Konkuk University, pp.12-17, 2018.
- Kleinsmith, A., & Gillies, M., “Customizing by doing for responsive video game characters” Human-Computer Studies, pp.77, 775-784, 2013.
- 게임에서 캐릭터의 융복합 커스터마이징 사례 분석 (2016. 하예진, 오승환)
- 윤인섭,마인크래프트 활용한 ‘인천크래프트 1945’ 탄생... 8·15 광복절에 무료 공개, 한국뉴스, 2021.08.12
- 내 손안에 서울, 가상세계 메타버스에 개장한 ‘서울어린이대공원’ 놀러 오세요!. 서울정보소통광장, 2021.08.20
- 이면재. "동작 인식 게임의 융합 발전 방향." 한국융합학회논문지 5.4 (2014): 1-7.
- Fox, J., Ahn, S. J., Janssen, J. H., Yeykelis, L., Segovia, K. Y. & Bailenson, J. N. (2015). "Avatars versusagents: a meta-analysis quantifying the effect of agency on social influence." Human–Computer Interaction, 30(5), 401-432.
- Blascovich, J. (2002). "Social influence within immersive virtual environments." In R. Schroeder (Ed.), The social life of avatars: Presence and interaction in shared virtual environments (pp. 127-145). London, UK: Springer-Verlag.
- Ahn, S. (2011). "Mirror worlds creation technology." Robot and Human, 8(4), 17-25
- Gelernter, D. (1992). Mirror worlds. NY: Oxford University Press.
- Castelfranchi, C., Piunti, M., Ricci, A. & Tummolini, L. (2012). AmI systems as agent-based mirror worlds: bridging humans and agents through stigmergy. In Agents and ambient intelligence (pp. 17-31). IOS Press.
- IBM. (2019). Digital twin: Bridging the physical-digital divide - watson IoT blog. IBM Business Operations Blog. February 27. https://www.ibm.com/blogs/internet-of-things/iot-digital-twinenablers/ (Retrieved on July 22, 2021)
- Negri, E. (2017). "A review of the roles of digital twin in CPS-based production systems." Procedia Manufacturing, 11, 939-948.
- Eshkenazi, A. (2019). "Real benefits from digital twins." ASCM. September 21. https://www.ascm.org/ ascm-insights/scm-now-impact/real-benefitsfrom-digital-twins/ (Retrieved on July 15, 2021).
- Deloitte. (2017). "Meet manufacturing's digital twin." WSJ. August 9. https://deloitte.wsj.com/articles/meet-manufacturings-digitaltwin-1502251346 (Retrieved on June 1, 2021).
- Nomoko. (2020). "The era of digital twins and the mirror world." Medium. May 5. https://nomoko.medium.com/the-era-of-digital-twins-and-the-mirror-world82b33e3e3d46 (Retrieved on July 23, 2021).
- Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. & MacIntyre, B. (2001). "Recent advances in augmented reality." IEEE computer graphics and applications, 21(6), 34-47.

- Rouillard, J (2008). Contextual QR codes. In 2008 The Third International Multi-Conference on Computing in the Global Information Technology (iccgi 2008) (pp. 50-55). IEEE.
- Chung, D. (2017). "User-based theories and practices on virtual reality." Informatization Policy, 24(1), 3-29
- Chung (2021). Digital transformation in content business. Seoul: Nexus.
- Gurrin, C., Smeaton, A. F. & Doherty, A. R. (2014). "Lifelogging: Personal big data." Foundations and trends in information retrieval, 8(1), 1-125.
- Bae, Y. (2012). "A study on the Diffusion of Life Log and the Right to be Forgotten." Internet and Information Security, 3(4), 86-99
- Silver, N. (2012). The signal and the noise: why so many predictions fail-but some don't. NY: Penguin.
- Cho, W. (2020). "Files that are not used by 52% of the total. Data cleanup is competitive." Maeil Economics, July 9. https://www.mk.co.kr/news/business/view/2020/07/702088 (Retrieved on July 15, 2021).
- 송원철, 정동훈. 메타버스 해석과 합리적 개념화. 2021; 28(3), 3-22. Available from: doi:https://doi.org/10.22693/NIAIP.2021.28.3.003
- 이연빈. "VR에서 디지털휴먼과 사회적 상호작용이 사회적 실재감에 미치는 영향." 국내석사학위논문 홍익대학교 영상·커뮤니케이션대학원, 2020. 서울
- https://biz.chosun.com/site/data/html_dir/2021/03/15/2021031501872.html
- 메타버스 시대의 제페토성공사례에 관한 연구- 9가지 빌딩블록 비즈니스 모델을 중심으로- 호서대학원 AI콘텐츠융합학과 AI융합마케팅전공 김 다 빈
- http://www.riss.or.kr/search/detail/DetailView.do?p_mat_type=be54d9b8bc7cdb09&control_no=012e7a9616b20056ffe0bdc3ef48d419&keyword=unity
- 메타버스 기반의 가상 현실 공간 상담 서비스 플랫폼 연구 및 개발 - 김제현·오석희 (가천대학교 컴퓨터공학과)

※ 예산 집행현황(기술이전형 과제수행의 경우 기업연계 재료비 사용 내역도 추가하여 작성)

구분	일자	사용 내역	금액
합계			

※ 최종 결과보고서에는 반드시 개발 작품의 사진이 포함되어야 함

캡스톤디자인 산학연계 교육 실적보고서

캡스톤디자인 교과목명 (교과목코드)	캡스톤디자인2(기업연계프로젝트) 374120			
캡스톤디자인 과제명	메타버스 플랫폼 제작			
교육기간	2022년 03월 02일 ~ 2022년 06월 12일			
교육개요	본 교육은 실제 산업현장에서 부딪칠 수 있는 문제들을 학문 분야별로 습득한 전문 지식을 바탕으로 해결할 수 있는 능력을 기르도록 교육을 실시함.			
기업체전문가	소속		성명	
교육내용	메타버스에 대한 정의와 미래, 핵심 기능에 대한 지식을 공유해주고 어떤 점을 중점으로 진행하여야 하는지에 대해 멘토링을 하였다. 팀 프로젝트를 진행하는 데 있어서 어떤 흐름으로 해야 효율적이고 결과의 퀄리티를 높일 수 있는지와 역할을 어떤 식으로 분담해야 하는지에 대해 지도를 해주었다. 진행하고 있는 프로젝트를 보고 피드백을 진행하였다. 멀티플레이어 개발을 지원해주는 Photon을 메타버스에 붙이는 과정에서 겪었던 어려운 점들에 대해 질문을 하고 조언하였다. C#을 이용해 아바타 커스터마이징 코딩 작업을 하는 과정에서 오류가 났던 부분에 대해 피드백을 해주고 어떤 식으로 하면 현재보다 체계적으로 코딩할 수 있는지에 대해 멘토링을 진행하였다.			
교육운영결과	개인이 가지고 있었던 메타버스 지식보다 수준 높은 지식을 얻게 되었다. 메타버스의 구체적인 정의와 앞으로 메타버스가 나아가야 할 방향과 흐름을 알게 되었다. 팀 프로젝트를 진행하는 데 있어서 막막한 것이 많았는데 흐름을 제시해주시고 여러 가지 조언을 해주셔서 어떤 식으로 팀을 이끌어나가고 프로젝트를 진행해야 할지 감을 잡게 되었다. Photon에 대한 이해와 응용력을 높일 수 있게 되었다. 아바타 커스터마이징 기능에 대한 직접 조언을 받고 도움을 받아 문제를 오랫동안 잡고 있지 않아서 프로젝트를 진행하는데 차질이 생기지 않았다.			

위와 같이 캡스톤디자인(과제명) 산학연계 교육 실적보고서를 제출합니다.

2022 년 06월 16일

기업체전문가 : (인)

원광대학교 LINC 3.0 사업단장 귀하

캡스톤디자인 지도 실적 보고서(지도교수용)

캡스톤디자인 교과목명 (교과목코드)	캡스톤디자인2(기업연계프로젝트) 374120			
캡스톤디자인 과제명	메타버스 플랫폼 제작			
지도학생				
지도개요	본 교육은 학문 분야별로 습득할 수 있는 전문 지식을 어디서 배울 수 있고 실제 산업 현장에서 마주하게 되는 팀별 문제 해결할 수 있는 능력을 기르도록 교육을 실시함.			
지도교수	소속	디지털콘테츠공학과	성명	
세부 지도내용	큰 주제의 방향을 제시해주고 이를 토대로 학생들이 자립적으로 구체적인 목표를 잡을 수 있도록 지도함. 메타버스라는 아이디어를 던져주고 세부적인 기능과 구현 방식들은 학생들끼리 구체적으로 계획을 짤 수 있도록 함. 학생들이 프로젝트를 진행함에 있어서 정해진 기간 안에 순조롭게 작업을 진행할 수 있도록 지도함. 어려워하는 부분의 해결 방향을 제시해주고 조언을 해주어 작업 진행에 차질이 없도록 지도함. 부족한 부분에 대해 피드백을 진행하고 색다른 방향성을 제시하여 학생들의 과제 결과의 퀄리티를 높이고자 함.			
수행기간	2022년 03월 02일 ~ 2022년 06월 12일			

위와 같이 캡스톤디자인(과제명)의 실적 보고서를 제출합니다.

2022년 06월 16일

지도교수 : (인)

원광대학교 LINC 3.0 사업단장 귀하