Parallel

게임 소개

- 타이틀 : Parallel
- 개발인원 : 4명
- 장르 : 공포(방탈출게임)
- 개발기간 : 2021.03.14~2021.07.29
- 개발환경 : Unity,Visual Studio
- 게임스토리

흉가 체험 신입 유튜버인 주인공은 폐교에 대한 소문을 듣고 촬영을 위해 찾아갑니다. 폐교에 들어간 뒤 대문이 닫히며 잠기게 되지만 오히려 오기가 생겨 더욱 안으로 들어갑니다. 폐교 안을 돌아다니다가 귀신과 마주치게 되고,크게 놀라 도망치다가 유일하게 잠기지 않은 문이 있어 들어가니 화장실이었습니다.
주인공이 되어 귀신들을 피해 열쇠를 찾아 학교를 탈출하는 스토리입니다.

- 다운로드 링크

플레이스토어 : https://play.google.com/store/apps/details?id=com.BJSH.Parallel&hl=ko&gl=US
원스토어 : https://m.onestore.co.kr/mobilepoc/apps/appsDetail.omp?prodId=0000758221

Main Script

- Manager.cs : 게임의 전체적인 관리를 하는 스크립트
- MainBtnManager.cs : 게임의 모든 버튼을 관리하는 스크립트
- MonsterHelp.cs : 게임내 귀신을 마주칠때 귀신의 정보를 보여주는 스크립트
- mRotate.cs : 플레이어의 회전을 하는 스크립트
- mJoyStick.cs : 플레이어를 움직이게 해주는 조이스틱을구현한 스크립트
- Camera.cs : 게임내의 카메라의 기능을 하는 스크립트
- ScrollSnap.cs : 게임내 인벤토리를 관리하는 스크립트
- ScreenShot.cs : 게임에서 스크린샷을 구현해 이미지를 저장해주는 스크립트
- Battery.cs : 배터리를 관리하는 스크립트

MonsterHelp.cs

```
void Start()
{
    originPos = cam.transform.localPosition;
    if (!PlayerPrefs.HasKey("KrasueHelp"))
        PlayerPrefs.SetInt("KrasueHelp", 0);

    if (!PlayerPrefs.HasKey("WomanHelp"))
        PlayerPrefs.SetInt("WomanHelp", 1);

    if (!PlayerPrefs.HasKey("SpiderHelp"))
        PlayerPrefs.SetInt("SpiderHelp", 2);

    if (!PlayerPrefs.HasKey("MannequinHelp"))
        PlayerPrefs.SetInt("MannequinHelp", 3);

    if (!PlayerPrefs.HasKey("DemonicHelp"))
        PlayerPrefs.SetInt("DemonicHelp", 4);

}
```

```
public void ShowMonsterHelp(string s)
{
    switch (PlayerPrefs.GetInt(s))
    {
        case 0:
            krasueHelp.SetActive(true);
            QuitBtn.SetActive(true);
            TouchManager.Instance.joystick.isTouch = false;
            ManagerCS.MInstance.PlayingCheck = false;
            PlayerPrefs.SetInt("KrasueHelp", 5);
            break;
        case 1:
            womanHelp.SetActive(true);
            QuitBtn.SetActive(true);
            TouchManager.Instance.joystick.isTouch = false;
            ManagerCS.MInstance.PlayingCheck = false;
            PlayerPrefs.SetInt("WomanHelp", 6);
            break;
        case 2:
            spiderHelp.SetActive(true);
            QuitBtn.SetActive(true);
            TouchManager.Instance.joystick.isTouch = false;
            ManagerCS.MInstance.PlayingCheck = false;
            PlayerPrefs.SetInt("SpiderHelp", 7);
            break;
        case 3:
            mannequinHelp.SetActive(true);
            QuitBtn.SetActive(true);
            TouchManager.Instance.joystick.isTouch = false;
            ManagerCS.MInstance.PlayingCheck = false;
            PlayerPrefs.SetInt("MannequinHelp", 8);
            break;
        case 4:
            demonicHelp.SetActive(true);
            QuitBtn.SetActive(true);
            TouchManager.Instance.joystick.isTouch = false;
            ManagerCS.MInstance.PlayingCheck = false;
            PlayerPrefs.SetInt("DemonicHelp", 9);
            break;
    }
}
```

MonsterHelp.cs

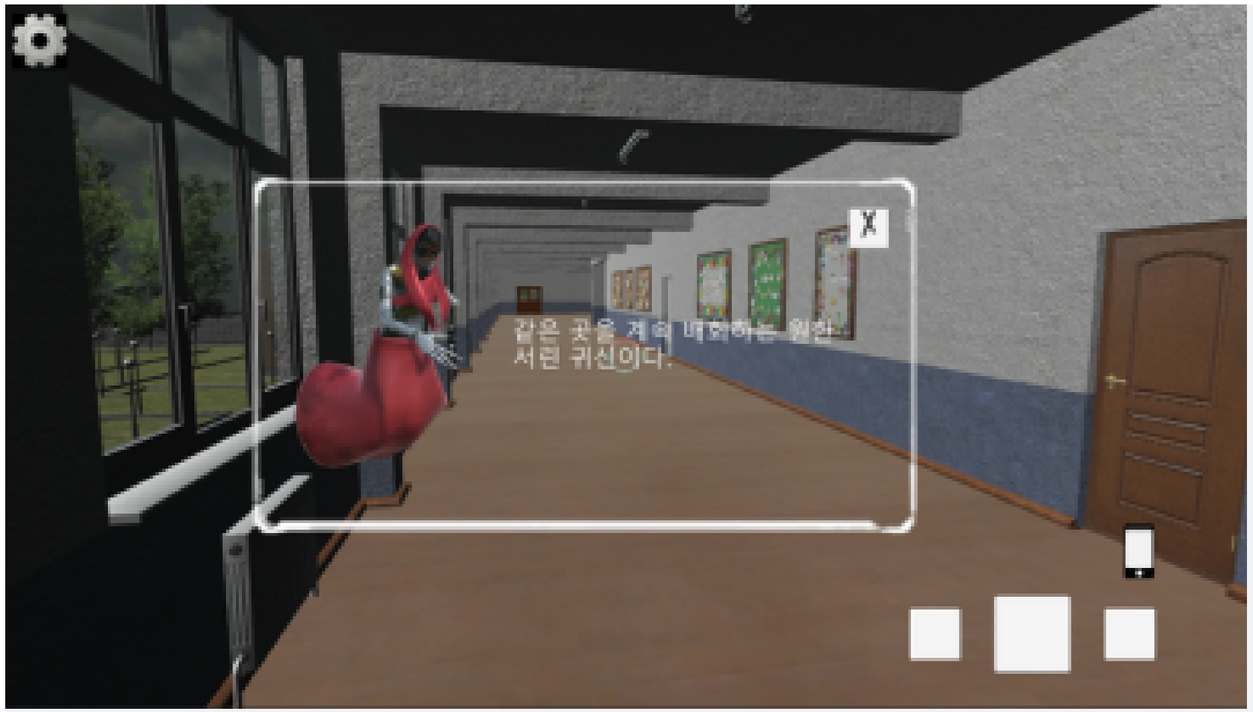
```
public void FindMonster()
{
    Collider[] targets = Physics.OverlapSphere(player.transform.position, viewDistance, targetMask);


    for (int i = 0; i < targets.Length; i++)
    {
        Transform ttarget = targets[i].transform;
        Vector3 dirToTarget = (ttarget.position - player.transform.position).normalized;
        if (Vector3.Dot(player.transform.forward, dirToTarget) > Mathf.Cos((viewAngle / 2) * Mathf.Deg2Rad))
        {
            float distToTarget = Vector3.Distance(player.transform.position, ttarget.position);

            if (!Physics.Raycast(player.transform.position, dirToTarget, distToTarget, elseMask))
            {
                Debug.DrawLine(player.transform.position, ttarget.position,Color.red);
                if (ManagerCS.MInstance.CheckObjectIsInCamera(krasue, cam) && EnemyDistance(player, krasue) < 7f && ManagerCS.MInstance.HorrorMap.activeSelf == true)
                {
                    ShowMonsterHelp("KrasueHelp");
                }

                if (ManagerCS.MInstance.CheckObjectIsInCamera(woman, cam) && EnemyDistance(player, woman) < 14f ||
                    (ManagerCS.MInstance.CheckObjectIsInCamera(woman2, cam) && EnemyDistance(player, woman2) < 14f))
                {
                    ShowMonsterHelp("WomanHelp");
                    StartCoroutine(Shake(0.005f, 1));
                }

                if (ManagerCS.MInstance.CheckObjectIsInCamera(spider, cam) && EnemyDistance(player, spider) < 5f && ManagerCS.MInstance.HorrorMap.activeSelf == true)
                {
                    ShowMonsterHelp("SpiderHelp");
                    StartCoroutine(Shake(0.005f, 1));
                }

                if (ManagerCS.MInstance.CheckObjectIsInCamera(mannequin, cam) && EnemyDistance(player, mannequin) < 7f)
                {
                    ShowMonsterHelp("MannequinHelp");
                    StartCoroutine(Shake(0.005f, 1));
                }

                if (ManagerCS.MInstance.CheckObjectIsInCamera(demonic, cam) && EnemyDistance(player, demonic) < 5f)
                {
                    ShowMonsterHelp("DemonicHelp");
                    StartCoroutine(Shake(0.005f, 1));
                }
            }
        }
    }
}
```

코드설명

일정거리 안에 들어온 귀신의 정보를 보여주는 코드입니다.
시작 시 PlayerPrefs를 통해 귀신들의 Key값을 지정해줍니다.
나중에 지정한 키 값을 비교해 한번 본 귀신들의 정보는 다시 뜨지 않게 설정해놨습니다.

MonsterHelp.cs

예시이미지



mjoyStick.cs,mRootate

```
public class mJoyStick : MonoBehaviour, IPointerDownHandler, IDragHandler, IPointerUpHandler
{
    public Vector3 JoyDirection
    {
        get
        {
            if (joyDirection == null)
            {
                Debug.LogError("joy Direction Value is missing!");
                joyDirection = Vector3.zero;
            }
            return (Vector3)joyDirection;
        }
    }
    private Vector3? joyDirection;
    [SerializeField] private RectTransform rect_BackGround;
    [SerializeField] private RectTransform rect_Joystick;

    public bool isTouch = false;
    private float stickRange;

    private void Start()
    {
        stickRange = rect_BackGround.rect.width * 0.3f;
        joyDirection = Vector3.zero;
    }
    public void OnPointerDown(PointerEventData eventData)
    {
        isTouch = true;
        rect_BackGround.position = eventData.position;
    }
```

```
public void OnDrag(PointerEventData eventData)
{
    if (ManagerCS.MInstance.PlayingCheck && isTouch == true)
    {
        Vector3 stickPosition = eventData.position - (Vector2)rect_BackGround.position;
        rect_BackGround.gameObject.SetActive(true);
        joyDirection = stickPosition.normalized;
        rect_Joystick.localPosition = Vector2.ClampMagnitude(stickPosition, stickRange);
    }
}

public void OnPointerUp(PointerEventData eventData)
{
    isTouch = false;
    rect_Joystick.localPosition = Vector2.zero;
    rect_BackGround.gameObject.SetActive(false);
    joyDirection = Vector3.zero;
}

public void playAnim()
{
    isTouch = false;
    joyDirection = Vector3.zero;
    rect_BackGround.gameObject.SetActive(false);
}
```

mjoyStick.cs,mRootate

```
public class mRotate : MonoBehaviour, IPointerDownHandler, IDragHandler
{
    Vector3 FirstPoint;
    Vector3 SecondPoint;
    public float xAngle = 0f;
    public float yAngle = 0f;
    public Vector3 anlge;
    float xAngleTemp;
    float yAngleTemp;

    public void OnPointerDown(PointerEventData eventData)
    {
        BeginDrag(eventData.position);
    }

    public void OnDrag(PointerEventData eventData)
    {
        OnDrag(eventData.position);
    }

    public void BeginDrag(Vector2 a_FirstPoint)
    {
        FirstPoint = a_FirstPoint;
        xAngleTemp = xAngle;
        yAngleTemp = yAngle;
    }

    public void OnDrag(Vector2 a_SecondPoint)
    {
        SecondPoint = a_SecondPoint;
        xAngle = xAngleTemp + (SecondPoint.x - FirstPoint.x) * 180 / Screen.width;
        yAngle = yAngleTemp - (SecondPoint.y - FirstPoint.y) * 90 * 3f / Screen.height;

        if (yAngle < -30f)
            yAngle = -30f;
        if (yAngle > 40f)
            yAngle = 40f;

        anlge = new Vector3(yAngle, xAngle);
    }
}
```

코드설명

Handler를 사용해 화면에 상호작용이 있을 시에 플레이어의 움직임과 회전을 관리하는 코드들입니다.

camera.cs

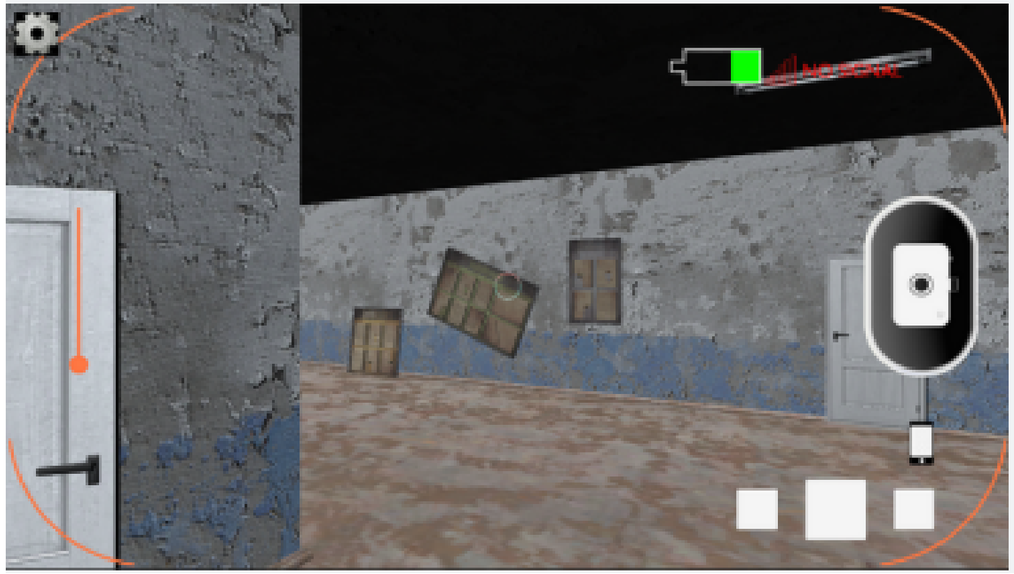
```
void Update()
{
    if (flashCtrl == true)
    {
        flashImage.color = Color.Lerp(flashImage.color, Color.clear, flashSpeed * Time.deltaTime);
    }
}
public void CameraFlash() // 스크린샷 기능 넣기
{
    flashImage.color = flashColor;
    flashCtrl = true;
    if (stunCount > 0)
    {
        stunCount--;
        ImageFill.instance.Fill(stunCount);
    }
}

public void BtnDown()
{
    cameraBtn.SetActive(true);
}
public void BtnUp()
{
    ManagerCS.MInstance.effectSource.clip = ManagerCS.MInstance.effecSound[3];
    ManagerCS.MInstance.effectSource.Play();
    cameraBtn.SetActive(false);
    StartCoroutine(CaptureIt());
    if (stunCount > 0)
    {
        if (ManagerCS.MInstance.CheckObjectIsInCamera(krasue, cam) ||
            ManagerCS.MInstance.CheckObjectIsInCamera(spider, cam) ||
            ManagerCS.MInstance.CheckObjectIsInCamera(monseterWoman, cam))
        {
            isStop = true;
        }
        else
        {
            isStop = false;
        }
    }
}
```

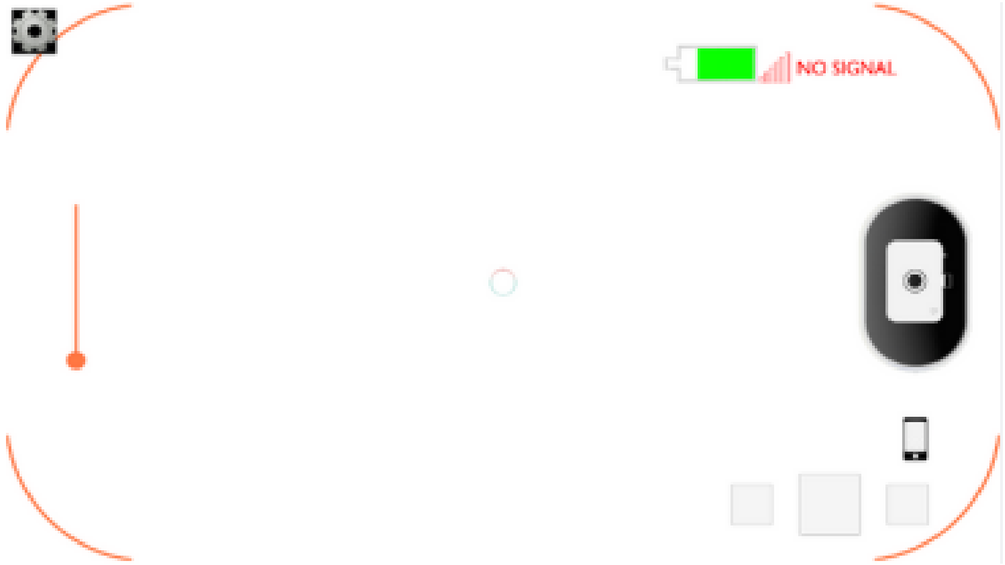
```
public bool CheckObjectIsInCamera(GameObject _target, Camera Cam)
{
    Transform enemy = _target.transform;

    Camera selectedCamera = Cam.GetComponent<Camera>();
    Vector3 left = selectedCamera.WorldToViewportPoint(enemy.position + enemy.right * TestFloat[0]
    Vector3 right = selectedCamera.WorldToViewportPoint(enemy.position + enemy.right * -TestFloat[
    Vector3 up = selectedCamera.WorldToViewportPoint(enemy.position + enemy.up * TestFloat[2]);
    Vector3 down = selectedCamera.WorldToViewportPoint(enemy.position + enemy.up * -TestFloat[3]);

    // not see
    if ((CanMove(left) && CanMove(right) && CanMove(up)) && CanMove(down))
    {
        return false;
    }
    else
        return true;
}
```

코드설명

카메라 버튼을 누를 경우 점멸 효과를 내는 코드입니다. 카메라 안에 귀신이 있을 경우 CheckObjectIsInCamera의 값이 true로 들어가 귀신이 멈추게 되는 기능입니다.

camera.cs





Battery.cs

```
[Header("배터리 채워져 있는 이미지")]
public Sprite fullbty;
[Header("배터리 라인")]
public Sprite btyline;
[Header("배터리 채울 배경")]
public Sprite btybg;
[Header("무슨색으로할지")]
public Color color;
Imagemask imagemask;
[Header("스크린샷 몇번인지")]
public int gaze;
private void Awake()
{
    if (instance != null)
        Destroy(gameObject);
    instance = this;
    gaze = Stun.stunCount;
    GameObject s = new GameObject();
    s.AddComponent<Image>();
    s.transform.parent = transform;
    s.GetComponent<RectTransform>().anchoredPosition3D = Vector3.zero;
    s.GetComponent<RectTransform>().localScale = Vector3.one;
    imagemask = s.gameObject.AddComponent<Imagemask>();
}

public void Fill(int N, float f = 0.5f)
{
    imagemask.FillGaze(N, f);
}
public void FillPerfect(int n)
{
    imagemask.FillPerfect(n);
}
```

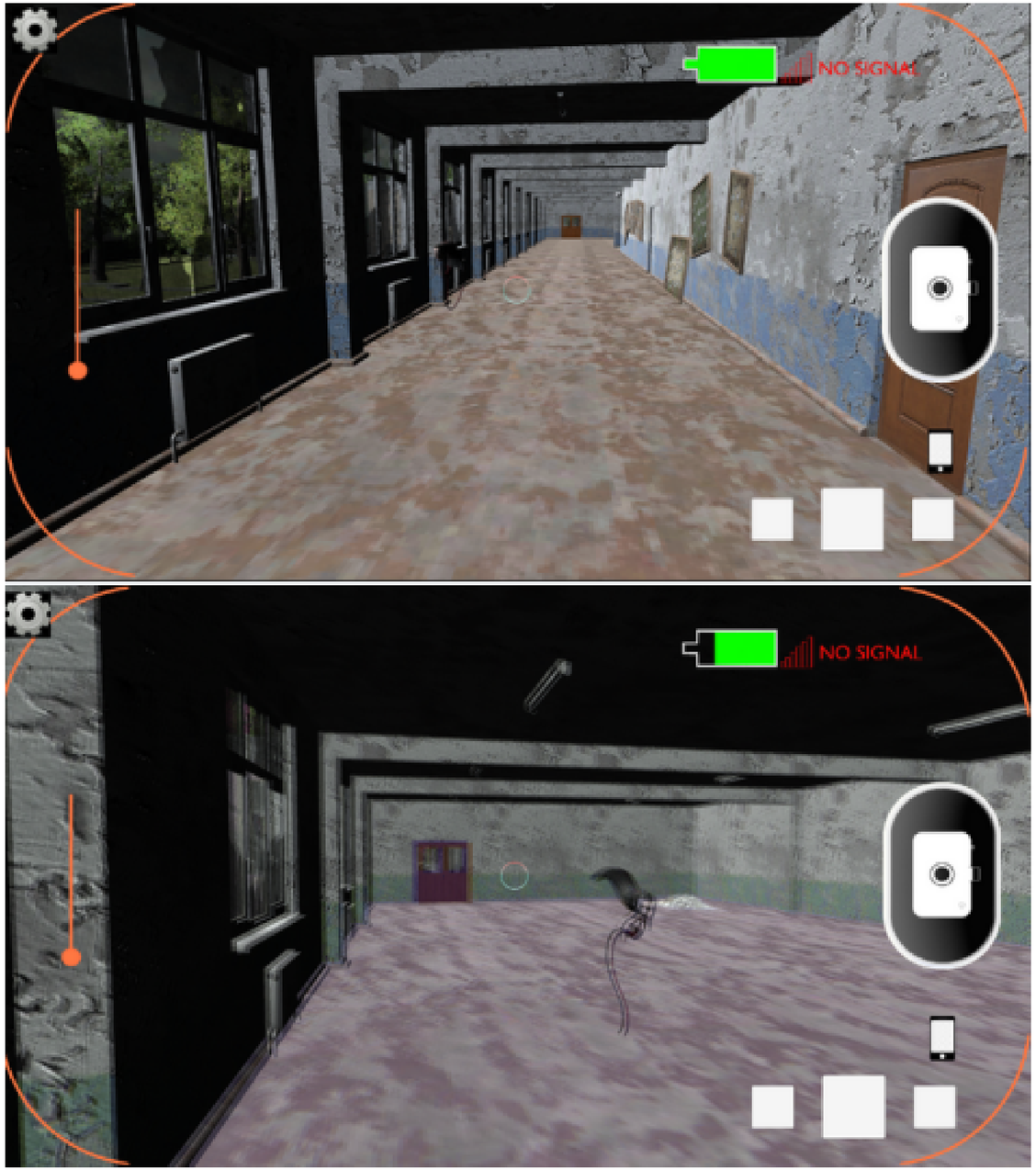
```
[RequireComponent(typeof(Mask))]

public class Imagemask : MonoBehaviour
{
    public Image img;
    public RectTransform imgrect;
    public RectTransform imgrect2;
    Image rectimg;
    Image rectimg2;
    RectTransform rect;
    float gaze;
    public float minus;
    public float plus;
    Mask m;
    private void Awake()
    {
        gaze = (float)ImageFill.instance.gaze;
        GameObject s = new GameObject();
        s.AddComponent<Image>();
        s.transform.parent = transform;
        s.GetComponent<RectTransform>().anchoredPosition3D = Vector3.zero;
        s.GetComponent<RectTransform>().localScale = Vector3.one;

        GameObject s1 = new GameObject();
        s1.AddComponent<Image>();
        s1.transform.parent = transform;
        s1.GetComponent<RectTransform>().anchoredPosition3D = Vector3.zero;
        s1.GetComponent<RectTransform>().localScale = Vector3.one;


        m = GetComponent<Mask>();
        m.showMaskGraphic = false;
        rect = GetComponent<RectTransform>();
        img = GetComponent<Image>();
        img.sprite = ImageFill.instance.fullbty;
        imgrect = transform.GetChild(0).GetComponent<RectTransform>();
        rectimg = imgrect.GetComponent<Image>();
        rectimg.sprite = ImageFill.instance.btyline;
        imgrect2 = transform.GetChild(1).GetComponent<RectTransform>();
        rectimg2 = imgrect2.GetComponent<Image>();
        rectimg2.sprite = ImageFill.instance.btybg;
        SetSize();
        ManagerCS.MInstance.BtyFill = rectimg2;
    }
```

```
public void SetSize()
{
    img.SetNativeSize();
    imgrect.sizeDelta = rect.sizeDelta;
    imgrect2.sizeDelta = rect.sizeDelta;
    rectimg2.type = Image.Type.Filled;
    rectimg2.fillMethod = Image.FillMethod.Horizontal;
    rectimg2.fillOrigin = (int)Image.OriginHorizontal.Right;
    rectimg2.color = ImageFill.instance.color;
    rectimg2.fillAmount = 1;
}
public void FillGaze(int n,float f1=0.5f)
{
    StartCoroutine(Fill(n,f1));
}
public void FillPerfect(int n)
{
    float f = (1 / gaze) * n;
    rectimg2.fillAmount = f;
}
IEnumerator Fill(int n,float f1)
{
    float time = 0;
    float originf = rectimg2.fillAmount;
    float f = (1 / gaze)*n;
    minus = originf - f;
    plus = Mathf.Abs(originf - f);
    if (f<originf)
    {

        while (time < 1)
        {

            time += Time.deltaTime / f1;
            rectimg2.fillAmount = originf - (minus*time);
            yield return null;
        }
    }
    else
    {
        while (time < 1)
        {
            time += Time.deltaTime / f1;
            rectimg2.fillAmount = originf + (plus * time);
            yield return null;
        }
    }
}
```

Battery.cs

코드설명

사진을 찍을 때 줄어드는 배터리 코드입니다. 배터리 기능을 나눈 하위 클래스를 만들어 전체 게이지 횟수 만큼 계산 후 나온 값 만큼 줄어들거나 늘어나게 구현해놨습니다.



ScreenShot.cs

```
IEnumerator CaptureIt() // 파일 저장명
{
    string timeStamp = System.DateTime.Now.ToString("yyyy_MM_dd_HH_mm_ss");
    string fileName = "ScreenShot_" + timeStamp + ".png";
    string pathToSave = fileName;
    ScreenCapture.CaptureScreenshot(Application.persistentDataPath + "/" + pathToSave);
    yield return new WaitForSeconds(0.1f);
    CameraFlash();
    yield return new WaitForEndOfFrame();
}
```

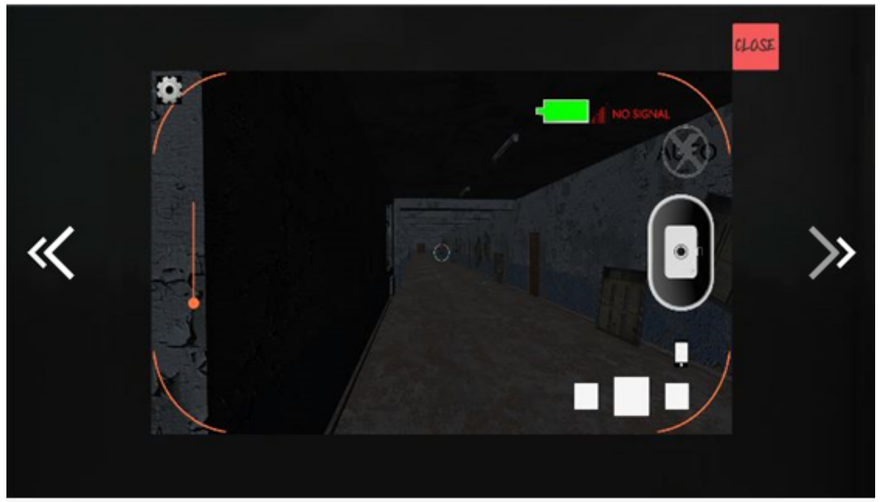
```
void Start()
{
    files = Directory.GetFiles(Application.persistentDataPath +"/", "*.png");
    if(files.Length>0)
    {
        GetPictureAnsShowIt();
    }
}

void GetPictureAnsShowIt()
{
    string pathToFile = files[whichScreenShotIsShown];
    Texture2D texture = GetScreenshotImage(pathToFile);
    Sprite sp = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height), new Vector2(0.5f, 0.5f));
    canvas.GetComponent<Image>().sprite = sp;
    picCountText.text = (whichScreenShotIsShown+1) + "/" + files.Length.ToString();
}
Texture2D GetScreenshotImage(string filePath)
{
    Texture2D texture = null;
    byte[] fileBytes;
    if(File.Exists(filePath))
    {
        fileBytes = File.ReadAllBytes(filePath);
        texture = new Texture2D(2, 2, TextureFormat.RGB24, false);
        texture.LoadImage(fileBytes);
    }
    return texture;
```

ScreenShot.cs

```
public void NextPicture() // 다음 캡쳐장면으로
{
    if(files.Length>0)
    {
        whichScreenShotIsShown += 1;
        picCountText.text = whichScreenShotIsShown.ToString() + "/" + files.Length.ToString();
        if (whichScreenShotIsShown > files.Length - 1)
            whichScreenShotIsShown = 0;
        GetPictureAnsShowIt();
    }
}
public void PreviousPictrue() // 이전 캡쳐장면으로
{
    if(files.Length>0)
    {
        whichScreenShotIsShown -= 1;
        picCountText.text = whichScreenShotIsShown.ToString() +"/"+ files.Length.ToString();
        if (whichScreenShotIsShown < 0)
            whichScreenShotIsShown = files.Length - 1;
        GetPictureAnsShowIt();
    }
}

public void ScreenShotReset()
{
    if (System.IO.Directory.Exists(Application.persistentDataPath + "/"))
        {
            string[] files = System.IO.Directory.GetFiles(Application.persistentDataPath +"/", "*.png");
            foreach (string s in files)
            {
                string fileName = System.IO.Path.GetFileName(s);
                string deletefile = Application.persistentDataPath +"/" + fileName;
                System.IO.File.Delete(deletefile);
            }
        }
}
```

코드설명

Camera코드에서 화면을 찍을 때
작동되는 코드입니다.
현재 카메라가 비추는 이미지를
현재 날짜의 파일로 만든 후 Sprite로
형변환 시켜 저장합니다.
게임 엔딩 때 내가 찍은 화면들을
보여주는 기능입니다.



ScrollSnap.cs

```
void Start()
{

    int btnLength = btn.Length;
    distance = new float[btnLength];
    distReposition = new float[btnLength];

    btndistance = (int)Mathf.Abs(btn[1].GetComponent<RectTransform>().anchoredPosition.x
        - btn[0].GetComponent<RectTransform>().anchoredPosition.x);
    //btndistance = 두번째버튼의 x좌표 - 첫번째버튼의 x좌표 의 절댓값


}
```

```
void LerpToBtn(float position)
{
    float newX = Mathf.Lerp(panel.anchoredPosition.x, position, Time.deltaTime * 10f);
    Vector2 newPosition = new Vector2(newX, panel.anchoredPosition.y);

    panel.anchoredPosition = newPosition;

}
```

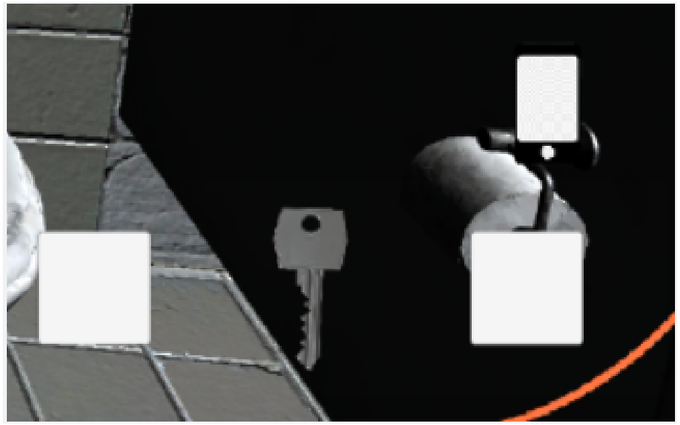
```
for (int i=0; i<btn.Length; i++)
{
    distReposition[i] = center.GetComponent<RectTransform>().position.x
        - btn[i].GetComponent<RectTransform>().position.x;
    //center :960 변동 없음
    //distReposition[i] = 센터의 x좌표 - 버튼[i]의 x좌표
    distance[i] = Math.Abs(distReposition[i]);

   if(distReposition[i]>=-50 && distReposition[i]<=50)
    {
        btn[i].GetComponent<RectTransform>().localScale =
           Vector2.Lerp(btn[i].GetComponent<RectTransform>().localScale,
           new Vector2(1.2f, 1.2f), 10f*Time.deltaTime);


    }
    else
    {
        btn[i].GetComponent<RectTransform>().localScale =
           Vector2.Lerp(btn[i].GetComponent<RectTransform>().localScale,
           new Vector2(0.8f, 0.8f), 10f * Time.deltaTime);


    }
    if (distReposition[i] > 450)
    {

        float curX = btn[i].GetComponent<RectTransform>().anchoredPosition.x;
        float curY = btn[i].GetComponent<RectTransform>().anchoredPosition.y;
        Vector2 newAnchoredPos = new Vector2(curX + (btn.Length * btndistance), curY);

        btn[i].GetComponent<RectTransform>().anchoredPosition = newAnchoredPos;

    }

    if (distReposition[i] < -450)
    {
        float curX = btn[i].GetComponent<RectTransform>().anchoredPosition.x;
        float curY = btn[i].GetComponent<RectTransform>().anchoredPosition.y;

        Vector2 newAnchoredPos = new Vector2(curX - (btn.Length * btndistance), curY);
        btn[i].GetComponent<RectTransform>().anchoredPosition = newAnchoredPos;

        btn[i].GetComponent<RectTransform>().localScale =
            Vector2.Lerp(btn[i].GetComponent<RectTransform>().localScale,
            new Vector2(0.8f, 0.8f), 1f);


    }
}
```

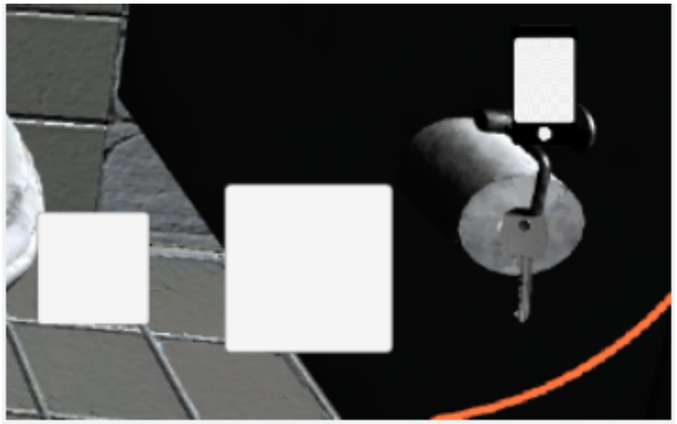
ScrollSnap.cs

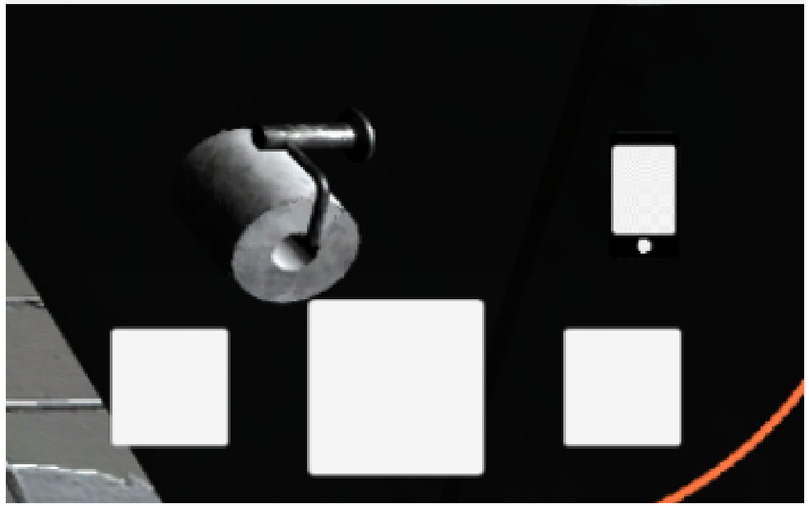
코드설명

Mask를 사용해 5개의 아이템칸 3칸만 보여지게 되며, 그 중 가운데 위치한 아이템만 사용할 수 있게 설정해 놓았습니다. 사용할 수 있는 아이템은 크기가 커지게 됩니다.

드래그 한 방향으로 움직여 일정거리를 벗어났을 때 반대 방향으로 이동시키는 코드입니다.







긴 글 봐주셔서 감사합니다.