Ghost Houes

작성자 : 김재완

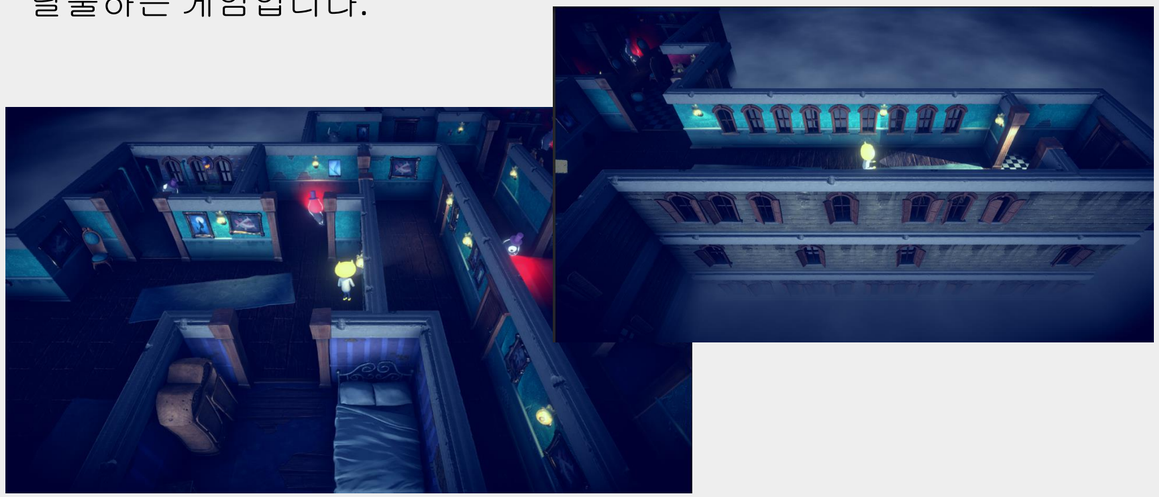
목차

1. 개요

항목	설명
제목	Ghost Houes(고스트 하우스)
장르	캐주얼 퍼즐 방탈출
디바이스 / 플랫폼	- PC
기획 의도	1. 유니티 튜토리얼을 따라하고 좀더 살을 붙여 좀더 재미있게 만들기 2. 튜토리얼 중 마음에 들지 않은 곳을 고쳐 좀더 좋게 말들기
특징	1. 기존 유령이 플레이어를 잡는 방법을 개선 2. 3가지의 퍼즐을 추가하여 좀 더 방탈출 같게 만들기
한 줄 소개	유령을 피해 아이템을 찾고 퍼즐을 풀어 유령의 집을 탈출하자

2. 기존 시스템

기존 게임은 단순히 유령을 피해 목적지 까지 도달하여
탈출하는 게임입니다.



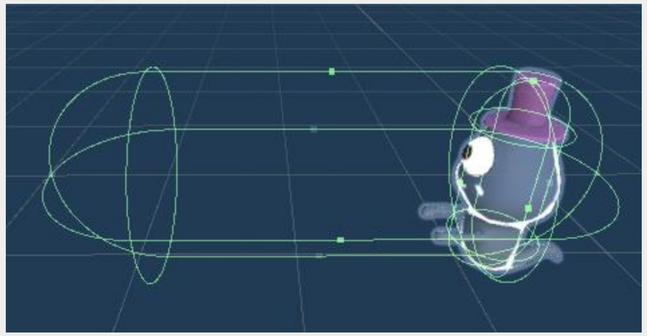
처음 이 게임의 목표는 단순 유니티에 적응하기
위한
프로젝트 따라하기 였지만

좀 더 나만의 시스템을 추가하여 재밌게 만들고
싶어 졌습니다.

우선 가장 큰 문제점이라고 생각했던 유령이
플레이어를 잡는 부분이였습니다.

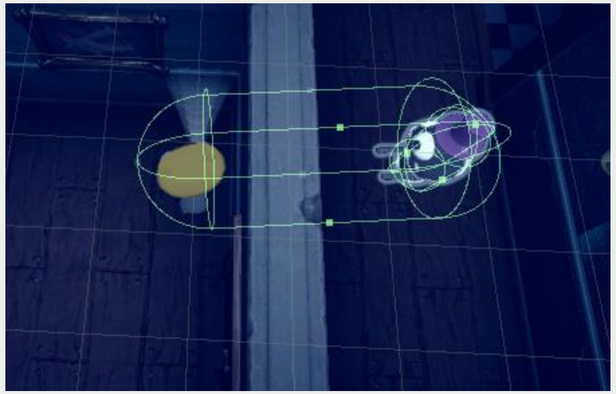
기존 튜토리얼에서는 플레이어를 찾을때
콜라이더를 이용하여 체크를 합니다.

그러다 플레이어가 걸리면 게임오버하는
방식입니다.



문제는 유령이 회전을 하거나 플레어어가 벽에
붙어 있다 벽넘어 걸리는 문제점이 있었습니다.

그리고 단순히 유령만 피하는
것이 아니라
좀더 다양한 이벤트를 주어
재미요소를 추가 하기로
하였습니다.



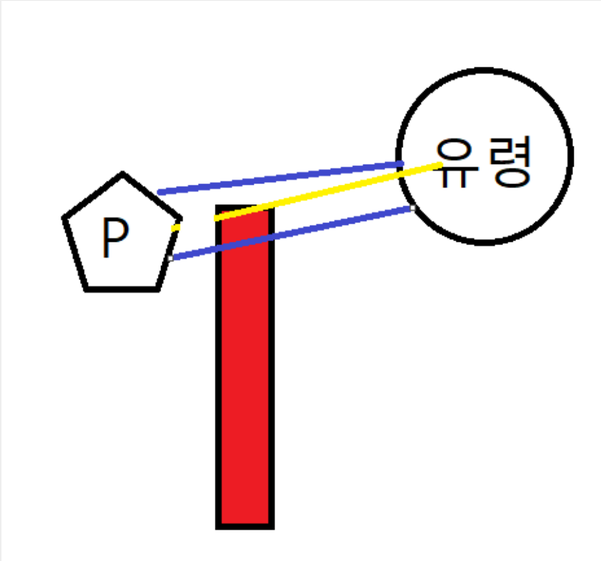
3. 추가 구현 설명 - 타겟시스템

기존 플레이어를 찾는 콜라이더를 지우고

코루틴과 레이케스터를 이용하여
좀더 사용하기 좋게 만들었습니다.

레이어를 사용하여 장애물과 플레이어를
구분하여 장애물 넘어 플레이어는 찾지
못하게 했습니다.

그리고 레이케스트를 3개를 쏘는 이유는
그림과 같이 잘못하면 보이는데 안보이게
처리가 되는 경우를 막기 위해서 입니다.



타겟을 찾는 코드의 일부

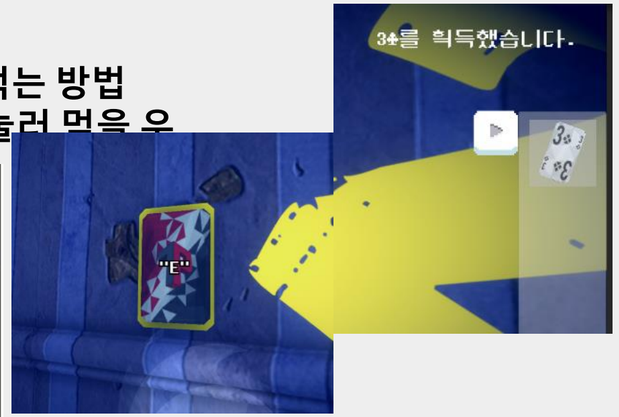
```
참조 1개
IEnumerator FindTargetsDelay(float delay)//일정 주기동안 타겟을 찾는
{
    WaitForSeconds delaySc = new WaitForSeconds(delay);

    while(true)
    {
        FindTargets();
        yield return delaySc;
    }
}

참조 1개
void FindTargets() //타겟을 찾는
{
    //사거리에 걸리는 모든 콜라이더
    Collider[] targets = Physics.OverlapSphere(transform.position, viewRadius, LayerMask_target);
    for (int i = 0; i < targets.Length; i++)
    {
        Vector3 targetpos = targets[i].bounds.center; //타겟의 콜라이더 센터 위치
        Vector3 dirToTarget = (targetpos - transform.position).normalized; //타겟으로의 방향
        if(Vector3.Angle(transform.forward,dirToTarget) < viewAngle/2)  //전방 벡터와 타겟방향벡터의 크기가 시야각 1/2이면 시야에 들어오는 상태
        {
            float radius = (targets[i] as CapsuleCollider).radius * 0.3f;
            Vector3 pos1 = targetpos + transform.right * radius;
            Vector3 pos2 = targetpos + (-transform.right) * radius;

            Vector3 pos1dir = (pos1 - transform.position).normalized;
            Vector3 pos2dir = (pos2 - transform.position).normalized; //타겟으로의 방향
            float disToTarget = Vector3.Distance(transform.position, targetpos);// 타겟까지의 거리 계산

            //타겟의 중앙, 타겟의 좌우방향으로 콜라이더를 한번더 확인하기
            if (!Physics.Raycast(transform.position, dirToTarget, disToTarget,LayerMask_obstacle) ||
                !Physics.Raycast(transform.position, pos1dir, disToTarget, LayerMask_obstacle) ||
                !Physics.Raycast(transform.position, pos2dir, disToTarget, LayerMask_obstacle)
                )  //타겟까지 또다른 레이저를 발사
            {//걸리면 장애물이 있다는 소리
                targets[i].GetComponent<Life>().TakeDamage(0.4f);
            }
        }
    }
}
```

3. 추가 구현 설명 - 아이템 찾기

처음으로 기존에 없던 시스템 으로
맵에 있는 아이템을 찾는 것입니다.

아이템은 문을 열때 사용하거나
퍼즐의 힌트로 사용하기 위한 아이템으로
만들었습니다.

각 아이템 마다 먹는 방법이 다르며
열쇠는 집적 가서 콜라이더 충돌로 먹는 방법
카드는 마우스를 이용하여 “E”키를 눌러 먹을 수



ItemData를 이용하여 필요한 정보를 만들어 놓고

```
[CreateAssetMenu(fileName = "ItemData", menuName = "ScriptableObjects/ItemData", order = 1)]
Unity 스크립트|참조 3개
public class ItemData : ScriptableObject
{
    public int item_Id;
    public Sprite item_Img;
    public string item_Name;
}
```

게임 메니저를 통해 필요한 아이템 정보를 가져와
사용할 수 있습니다.

```
Unity 스크립트(자산 참조 1개)|참조 11개
public class GameManager : MonoBehaviour
{
    public static GameManager Inst;

    public PlayerMovement player;
    WaypointPatrol[] waypointPatrols;
    참조 0개
    public Vector3 PlayerPos { get { return player.transform.position; } }

    [Header("ItemData")]
    [SerializeField] private ItemData[] itemDatas;
    public Dictionary<int, ItemData> ItemDates = new Dictionary<int, ItemData>();
```

아이템 관련 함수 일부 Inven.cs

```
참조 2개
public void AddItem(int item_Id)
{
    SFXManager.Inst.SoundOnShot(eSFX.GET);
    invenSlots.Add(InvenSlot.AddSlot(slotObj, invenSlotTr, item_Id));
    ShowTxtBox(GameManager.Inst.ItemDates[item_Id].item_Name + "를 획득했습니다.");
}

참조 1개
public bool FindItem(int item_Id)
{
    return invenSlots.Find(item => item.item_Id == item_Id) == null ? false : true;
}

참조 1개
public void UseItem(int item_Id)
{
    InvenSlot find = invenSlots.Find(item => item.item_Id == item_Id);
    if (find == null)
        return;

    invenSlots.Remove(find);
    find.UseItem();
}
```

3. 추가 구현 설명 - 문

열쇠 아이템을 사용하여 문을 여는 시스템

가지고 있는 아이템 중 맞는 열쇠가 있다면
문을 열수 있다.

열쇠의 유무의 따라 나오는 인터페이스가 다릅니다.

문 오브젝트에 붙어있는 스크립

```
Unity 스크립트(자산 참조 2개) | 참조 0개
public class Door : MonoBehaviour
{
    public int openNeedItem = 0; //문을 열기위한 아이템 코드
    [SerializeField] Color doorColor;
    [SerializeField] MeshRenderer doorMark;
    [SerializeField] GameObject doorTxtObj;
    [SerializeField] Collider doorCollider;
    Animation doorAnim;
    Text doorTxt;

    bool opened = false;  //문이 열려있는지
    bool isPlayer = false;  //플레이어가 가까이 있는지
    [SerializeField] bool isOpen = false;    //문을 열 수 있는지

    Unity 메시지 | 참조 0개
    private void Awake()
    {
        doorAnim = GetComponentInChildren<Animation>();

    }
    Unity 메시지 | 참조 0개
    private void Start()
    {
        doorTxt = doorTxtObj.GetComponent<Text>();
        if (doorMark)
            doorMark.material.color = doorColor;
    }

    Unity 메시지 | 참조 0개
    private void Update()
    {
        if (Input.GetKeyDown(KeyCode.E))
        {
            OpenDoor();
        }
    }

    참조 1개
    public void CheckDoor()
    {
        if(Inven.Inst.FindItem(openNeedItem))//아이템이 있는지 확인
        {
            isOpen = true;
            doorTxt.text = "E 키를 눌러 문을 여세요";
        }
        else
        {
            doorTxt.text = "문을 열 열쇠가 필요합니다.";
            isOpen = false;
        }
    }
```

```
    참조 1개
    void OpenDoor()
    {
        if (!opened && isOpen && isPlayer)
        {
            Inven.Inst.UseItem(openNeedItem);//아이템 사용
            SFXManager.Inst.SoundOnShot(eSFX.DOOR);
            opened = true;
            doorTxtObj.SetActive(false);
            doorAnim.Play("Door_Open");

            Invoke("OffCollider", 1.0f);
        }
        else if (!isOpen && isPlayer)
        {
            doorAnim.Play("Door_Jam");
        }
    }

    참조 0개
    void OffCollider()
    {
        doorCollider.enabled = false;
    }

    Unity 메시지 | 참조 0개
    private void OnTriggerEnter(Collider other)
    {
        if(other.CompareTag("Player") && !opened)
        {
            isPlayer = true;
            CheckDoor();
            doorTxtObj.SetActive(true);
        }
    }

    Unity 메시지 | 참조 0개
    private void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isPlayer = false;
            doorTxtObj.SetActive(false);
        }
    }
}
```

플레이어가 문의 콜라이더에 들어오면
열쇠가 있는 확인을 하고
맞는 열쇠가 있으면 문을 열어준다.

3. 추가 구현 설명 - 퍼즐

모든 퍼즐에 성공하면 길을 막고 있는 안개를 제거 합니다.

퍼즐 시스템을 구현할때 생각한것이
퍼즐을 실행시켜주는 스크립 과 퍼즐 스크립으로 나누어 작업을 하였습니다.

```
namespace MiniGameHelper
{
    public delegate void QuestEvent();

    참조 4개
    interface MiniGame
    {
        참조 3개
        void QuestSetFunc(QuestEvent ClearFunc , QuestEvent CloesFunc);
        참조 3개
        void MiniGameStart();
        참조 3개
        void MiniGameClear();

    }
}
```

미니게임이라는 인터페이스를 만들어
이 함수를 가지고 관리하도록하였습니다.

```
Unity 스크립트(자산 참조 2개)|참조 0개
public class MiniGameQuest : MonoBehaviour
{
    MiniGame miniGame;
    [SerializeField] Text infoTxt;
    [SerializeField] Fog fog;

    Outline meshOutline;

    bool inPlayer = false;
    bool questClear = false;

    Unity 메시지|참조 0개
    private void Awake()
    {
        meshOutline = GetComponentInChildren<Outline>();

    }
    Unity 메시지|참조 0개
    private void Start()
    {
        miniGame = GetComponent<MiniGame>();
        miniGame.QuestSetFunc(QuestClear, QuestCloes);
    }
```

퍼즐을 실행시켜주는 곳에 본 퍼즐의 스크립을
추가하여 클리어시 보상과 종료시 필요한 함수를 주어
연결 시켜 주었습니다.

```
참조 1개
void QuestClear()
{
    questClear = true;
    fog.OffFogs();
    meshOutline.SetColor(Color.clear);
    GameManager.Inst.EnemyMove(true);
    GameManager.Inst.PlayerMove(true);
}

참조 1개
void QuestCloes()
{
    GameManager.Inst.EnemyMove(true);
    GameManager.Inst.PlayerMove(true);
}
```

퍼즐 성공시 지정된 안개를 꺼주고
멈춰 있던 오브젝트를 움직여줍니다.

3. 추가 구현 설명 - 퍼즐(숫자 순서대로 누르기)

무작위로 설정된 넘패드를 순서대로 눌러 해결하는
퍼즐입니다.

넘패드의 숫자대로 순서대로 눌러 클리어 할수 있습니다.
실패시 무작위로 바뀌고 다시 처음부터 입력합니다.

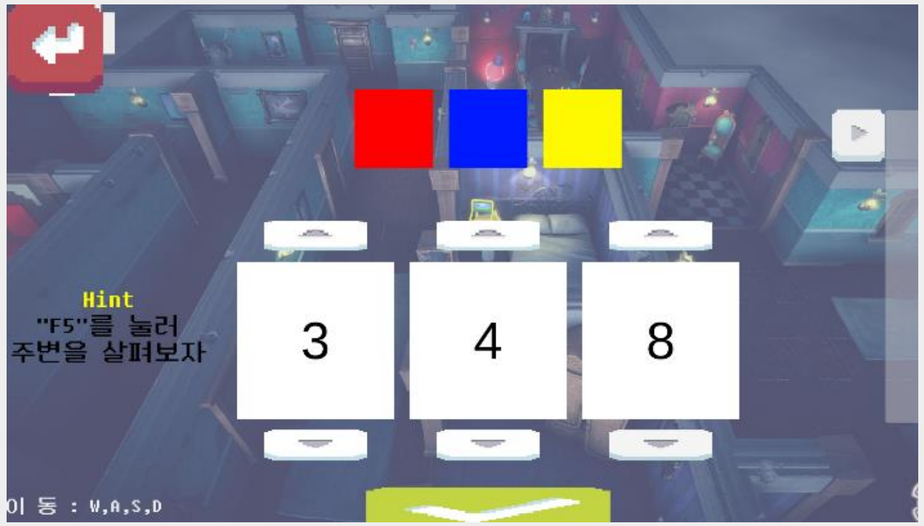
본 퍼즐 로직 코드 일부

```
참조 3개
public void MiniGameStart() //시작
{
    if (gameIng)
        return;

    gameIng = true;
    SettingNum();   //판 셋팅

    StartCoroutine(OpenUI()); //게임 UI Open
}

참조 1개
IEnumerator OpenUI() //오픈시 나타낼 효과
{
    gamePanel.SetActive(true);
    float timer = 0.0f;
    Vector2 originPos = numPadsRect.anchoredPosition;
    Vector2 endPos = Vector2.zero;
    Vector2 pos = Vector2.zero;

    while (timer < 1.0f)
    {
        timer += Time.deltaTime * 2.0f;
        yield return null;

        pos = Vector2.Lerp(originPos, endPos, timer);
        numPadsRect.anchoredPosition = pos;

    }

    infoBox[0].SetActive(true);
    infoBox[1].SetActive(true);
}
```

```
참조 1개
public bool PushNumPad(int num) //넘패드에서 호출될 함수
{
    audioSource.PlayOneShot(audioClips[0]);

    int pushNum = num - 1;
    if (curNum == pushNum)//성공
    {
        curNum++;

        if (curNum > 8)
            MiniGameClear();

        return true;
    }
    else //실패
    {
        NumPad.Push = false;
        StartCoroutine(GameFailed());
        return false;
    }

}

참조 3개
public void MiniGameClear() //성공시
{
    audioSource.PlayOneShot(audioClips[2]);
    ClearFunc?.Invoke();//외부 연결된 함수 실행
    gamePanel.SetActive(false); //UI off
}


참조 3개
public void SettingNum() //무작위 숫자 선정
{
    nums.Clear();
    for (int i = 1; i <= 9; i++)
        nums.Add(i);

    for (int i = 0; i < NumPads.Length; i++)
    {
        int rand = Random.Range(0, nums.Count);
        NumPads[i].SetNum(nums[rand]);
        nums.RemoveAt(rand);
    }

    curNum = 0;
    NumPad.Push = true;
}
```

3. 추가 구현 설명 - 퍼즐(숨은 숫자 찾기)

맵에 숨겨진 숫자를 찾아 입력하면되는 퍼즐입니다.



본 퍼즐 로직 코드 일부

```
Unity 스크립트 | 참조 0개
public class FindNum : MonoBehaviour, MiniGame
{
    public GameObject gamePanel;
    [Header("ColorNum")]
    [SerializeField] Text[] answerNumText;
    [SerializeField] Button[] numUpDownBtn;
    [SerializeField] Image[] numImage;
    [SerializeField] Button OkBtn;
    [SerializeField] Button closeBtn;
    int[] answerNums = new int[3] { 0, 0, 0 };

    QuestEvent ClearFunc;
    QuestEvent CloesFunc;

    [SerializeField] bool isPush = false;


    [SerializeField] private AudioSource audioSource;
    [SerializeField] private AudioClip[] audioClips;

    참조 3개
    public void QuestSetFunc(QuestEvent ClearFunc, QuestEvent CloesFunc)
    {
        this.ClearFunc = ClearFunc;
        this.CloesFunc = CloesFunc;
    }
    참조 3개
    public void MiniGameClear()
    {
        ClearFunc?.Invoke();
    }

    참조 3개
    public void MiniGameStart()
    {
        isPush = true;
        gamePanel.SetActive(true);
    }
```

```
참조 1개
void OkBtnFunc()
{
    if(answerNums[0] == 7 && answerNums[1] == 5 && answerNums[2] == 2)//성공
    {
        ClearQuest();
    }
    else //실패
    {
        StartCoroutine(FailedQuest());
    }
}

참조 1개
void ClearQuest()
{
    audioSource.PlayOneShot(audioClips[1]);
    ClearFunc?.Invoke();//외부 연결된 함수 실행
    gamePanel.SetActive(false);

}
```

```
참조 6개
void UpDownBtn(int idx, bool Up)
{
    if (!isPush)
        return;

    if (Up)
    {
        answerNums[idx]++;
        if (answerNums[idx] > 9)
            answerNums[idx] = 0;
    }
    else
    {
        answerNums[idx]--;
        if (answerNums[idx] < 0)
            answerNums[idx] = 9;
    }

    answerNumText[idx].text= answerNums[idx].ToString();
}
```

3. 추가 구현 설명 - 퍼즐(그림 맞추기)

맵에 숨겨진 카드(힌트)를 찾고
그림을 알맞게 회전 시켜 풀어내는 퍼즐입니다.

본 퍼즐 로직 코드 일부

```
Unity 스크립트 | 참조 2개
public class PicturePuzzle : MonoBehaviour
{
    [SerializeField]Fog fog;

    //0 6Spades , 1 6Heart , 2 9Diamond , 3 3Club
    public Picture[] pictures;

    참조 1개
    public void CheckPuzzle()
    {
        if (pictures[0].num == 6 && pictures[1].num == 6 && pictures[2].num == 9 && pictures[3].num == 3)
        {
            fog.OffFogs();
            gameObject.SetActive(false);
        }
    }
}
```

각 그림오브젝트에 들어가는 코드 일부

```
참조 1개
void StateUpdate() //상태에 따른 업데이트
{
    switch(state)
    {
        case State.Idle:
            {
                if (isPlayerSee)
                    if (Vector3.Distance(playerTr.position, transform.position) < Distance)
                        ChangeState(State.OnPlayer);
            } break;
        case State.OnPlayer:
            {
                E_txt.transform.position = Camera.main.WorldToScreenPoint(transform.position);
                if (Input.GetKeyDown(KeyCode.E))
                    ChangeState(State.Rot);

                if (Vector3.Distance(playerTr.position, transform.position) >= Distance)
                    ChangeState(State.Idle);

            } break;
        case State.Rot:
            {
                rotTime += Time.deltaTime;
                transform.rotation = Quaternion.Slerp(origin, next, rotTime);
                if (rotTime >= 1.0f)
                {
                    transform.rotation = next;
                    ChangeState(State.Idle);
                }

            } break;
    }
}
```

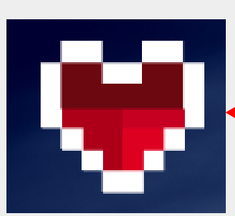
```
참조 6개
void ChangeState(State newState) //상태가 변화 했을때
{
    if (state == newState)
        return;

    state = newState;
    switch (state)
    {
        case State.Idle:
            {
                outline.SetColor("_OutlineColor", Color.clear);
                picturePuzzle.CheckPuzzle();
                E_txt.SetActive(false);
            } break;
        case State.OnPlayer:
            {
                outline.SetColor("_OutlineColor", Color.yellow);
                E_txt.SetActive(true);

            } break;
        case State.Rot:
            {
                audioSource.Play();
                E_txt.SetActive(false);
                origin = transform.rotation;
                next = origin * Quaternion.Euler(0, 0, 90.0f);
                rotTime = 0.0f;
            } break;
    }

}

참조 2개
public void OnPointerEnter(PointerEventData eventData)
{
    isPlayerSee = true;
}

참조 2개
public void OnPointerExit(PointerEventData eventData)
{
    isPlayerSee = false;

    if (state != State.Rot)
        ChangeState(State.Idle);
}
```

UI/UX

4. 인게임 화면

화면 구성

남은 체력을 나타냄



가지도 있는 아이템 정보를 알려줌 안보이게도 할 수 있다.

설정 버튼을 눌러 소리 설정 과 종료를 할 수 있다.