유니티 프로젝트 결과보고서

소프트웨어융합전공 1715016 김주휘

1. 제목 및 개요 (최초 계획서 대비 변경 사항 명시)

1) 최초 계획서 요약

프로젝트명	당신이 잠든 사이에 (While you were sleeping)
주제 및 개요	
방 탈출 게임을 모티브로 한 1인칭 시점의 스릴러 게임. 플레이어는 정신병으로 인해 수면제를 복용하고 잠에 드는데, 잠결에 들리는 물소리로 인해서 잠에서 깨게 된다. 잠에서 깨어보니 집 안 곳곳에서 물소리가 나고 있고, 복용하던 수면제도 사라진 상태. 정신병으로 인한 어둠에 대한 두려움과 환각을 통해 보이는 적을 피해 집 안에서 더 이상 물소리가 나지 않게 하고 수면제를 복용한 뒤 다시 잠에 들도록 하자.	
주요 기능 (구현 방법은 '3. 주요 기능'에서 설명함)	
분류	기능
플레이어	방향키를 이용한 플레이어의 이동 및 점프
	1인칭 시점 구현
	아이템을 잡았을 시 손에 들고 있는 효과
게임 진행	인벤토리 구현 및 활용
	아이템, 수도꼭지 선택
아이템	손전등
	지도

2) 최초 계획서 대비 변경사항

(1) 추가한 기능

주요 기능 (구현 방법은 '3. 주요 기능'에서 설명함)	
분류	기능
플레이어	체력
게임 진행	게임 진행 상황 표시
	스토리 및 조작 방법 안내
	적 생성
아이템	손전등의 배터리
	수면제

(2) 수정한 기능

주요 기능 (구현 방법은 '3. 주요 기능'에서 설명함)		
분류	기능	구현 방법 수정
아이템	수도꼭지 잠그기	수도꼭지를 클릭하면 물이 잠기도록 변경

(3) 삭제한 기능

주요 기능 (구현 방법은 '3. 주요 기능'에서 설명함)	
분류	기능
아이템	렌치

2. 외부 에셋 목록 및 추가 구현 내용

1) 유니티 에셋

① Charlet style furniture (https://assetstore.unity.com/packages/3d/props/furniture/chalet-style-furniture-31966) (무료)
② Small Furniture Pack (https://assetstore.unity.com/packages/3d/props/furniture/small-pack-furniture-56628) (무료)
③ Kitchen Creation Kit (https://assetstore.unity.com/packages/3d/environments/kitchen-creation-kit-2854) (무료)
④ Toon Furniture (https://assetstore.unity.com/packages/3d/props/furniture/toon-furniture-88740) (무료)
가구들의 3D 모델의 Prefab을 담은 에셋. Prefab에 Collider와 RigidBody를 추가하여 게임 플레이에 가구를 이용할 수 있도록 함.

⑤ Yughues Free Wooden Floor Materials (https://assetstore.unity.com/packages/2d/textures-materials/wood/yughues-free-wooden-floor-materials-13213)
⑥ Hand Painted Modular Pack Basic Tavern (https://assetstore.unity.com/packages/3d/hand-painted-modular-pack-basic-tavern-87515)
바닥을 표현하는 material을 포함한 에셋. 에셋에 있는 material을 게임에서의 바닥인 Plain에 적용하여 바닥을 꾸밈.

2) 외부 3D 모델

① Blender Batteries (https://www.turbosquid.com/FullPreview/Index.cfm/ID/767400)
배터리 모양의 3D 모델. Prefab에 Collider와 RigidBody를 추가하여 게임 플레이에 모델을 이용할 수 있도록 함.

② Pills bottle (https://www.turbosquid.com/FullPreview/Index.cfm/ID/938419)
약을 담은 통 모양의 3D 모델. Prefab에 Collider와 RigidBody를 추가하여 게임 플레이에 모델을 이용할 수 있도록 함.

③ 3D model Super Mario: Boo (https://www.turbosquid.com/FullPreview/Index.cfm/ID/1316625)
유령 모양의 3D 모델. Prefab에 Collider와 RigidBody를 추가하여 해당 모델을 적으로 이용함.

④ Flashlight 3d model (https://free3d.com/3d-model/hand-lamp-64089.html)
손전등 모양의 3D 모델. Prefab에 Collider와 RigidBody를 추가하여 게임 플레이에 모델을 이용할 수 있도록 함.

3) 사운드

① 유니티 스탠다드 에셋 – 캐릭터 에셋 중 FootStep1 오디오소스
유니티의 스탠다드 에셋 중 캐릭터 에셋에 포함된 오디오소스 중 하나인 FootStep1을 활용하여 플레이어에게 발걸음 소리를 내도록 구현함.

② 물 떨어지는 소리
water drop-02.aif https://freesound.org/people/kijjaz/sounds/16747/
water-drop-02.wav https://freesound.org/people/junggle/sounds/274004/
water drop-01.aif https://freesound.org/people/kijjaz/sounds/16746/
물방울은 유니티 기본 모델인 Capsule을 이용하였는데, 캡슐의 y 좌표가 기준 이하로 내려가면 캡슐이 없어지면서 소리가 나도록 구현함. 3D 사운드 세팅을 이용하여 물방울과 멀리 있을 때는 소리가 들리지 않도록 함.

③ 심장 소리
Heart Beat https://freesound.org/people/thenudo/sounds/146765/
Heartbeat thumping.wav https://freesound.org/people/Doctor_Jekyll/sounds/254062/
체력이 30%이상 남았을 때와 30%미만으로 떨어졌을 때 다른 심장 박동 소리를 내도록 함

④ 수도꼭지 잠그는 소리
Rachet click https://freesound.org/people/KieranKeegan/sounds/418850/

수도꼭지를 잠그기 위해 수도꼭지를 클릭한 순간 수도꼭지 잠그는 소리가 나도록 함.

⑤ 적이 가까이 왔을 때 나는 소리

Come out and play with me song.mp3 https://freesound.org/people/AlucardsBride/sounds/179659/

적의 위치에 따라 소리가 크고 작게 들리도록 3D 사운드 세팅을 이용하여 적이 가까이 왔을 때 소리가 크게 들리고 멀리 있으면 들리지 않도록 함.

⑥ 아이템을 집었을 때 나는 소리

Pick up Item 2.wav https://freesound.org/people/SilverIllusionist/sounds/411178/

아이템을 집으려고 클릭한 순간 해당 오디오소스가 Play되도록 함.

3. 주요 기능

1) 방향키를 이용한 플레이어의 이동 및 점프 (캡쳐 생략)

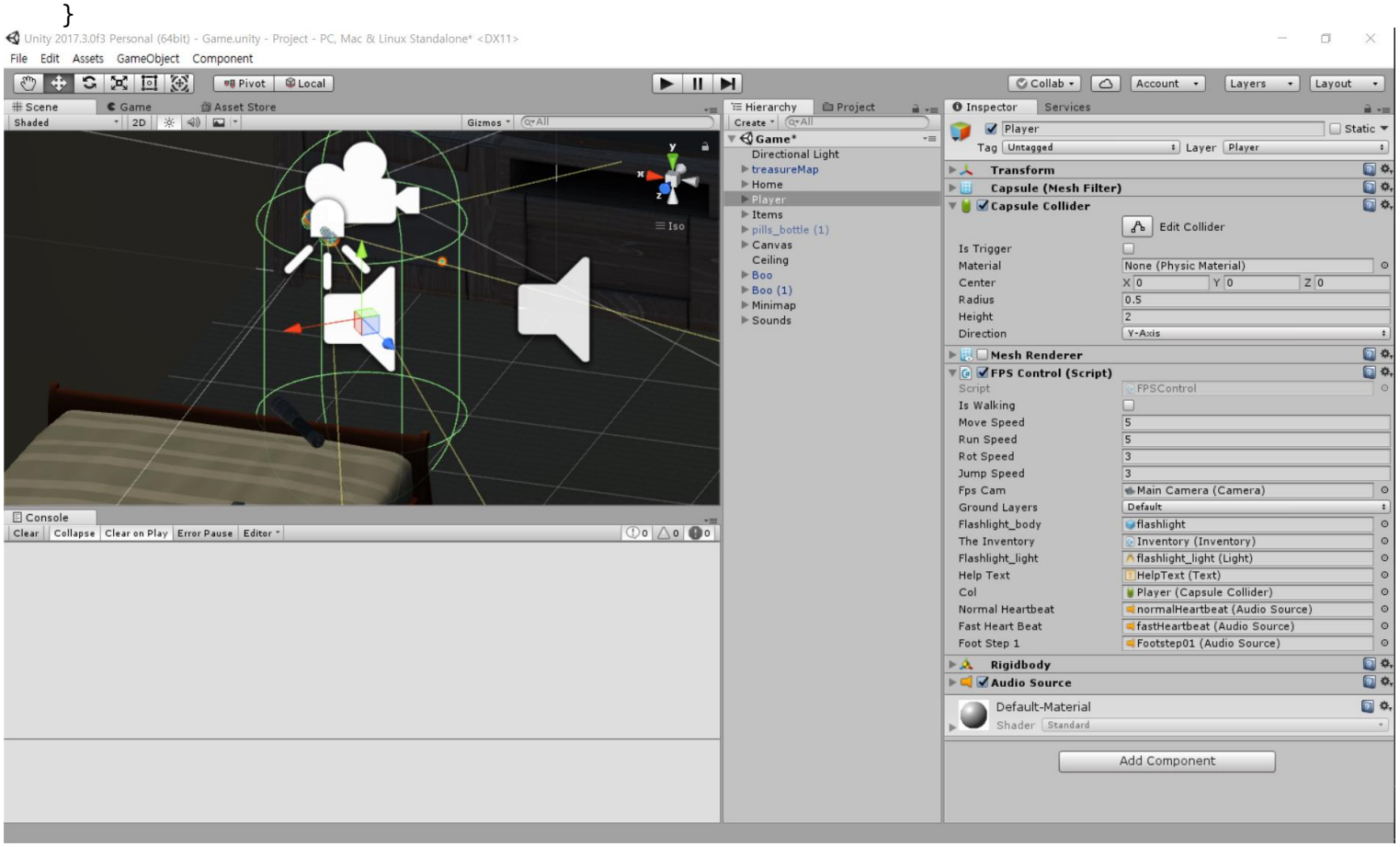
FPSControll.cs

```
    void MoveCtrl(){// 하드웨어에 상관 없이 일정한 속도로 이동할 수 있는 함수
        //W 키가 눌렸을 경우 앞으로 이동
         if(Input.GetKey(KeyCode.W)){
             this.transform.Translate(Vector3.forward*moveSpeed*Time.deltaTime);
            PlayWalkSound();
        }
        //S 키가 눌렸을 경우 뒤로 이동
        if(Input.GetKey(KeyCode.S)){
             this.transform.Translate(Vector3.back*moveSpeed*Time.deltaTime);
            PlayWalkSound();
        }
        //A 키가 눌렸을 경우 왼쪽으로 이동
        if(Input.GetKey(KeyCode.A)){
             this.transform.Translate(Vector3.left*moveSpeed*Time.deltaTime);
            PlayWalkSound();
        }
        //D 키가 눌렸을 경우 오른쪽으로 이동
        if(Input.GetKey(KeyCode.D)){
             this.transform.Translate(Vector3.right*moveSpeed*Time.deltaTime);
            PlayWalkSound();
        }
        //스페이스바를 눌렀고, 땅에 닿아 있을 경우 점프
        if(Input.GetKeyDown(KeyCode.Space)&&IsGrounded()){
             //ForceMode.Impulse 는 addForce 를 계속 하는게 아니라 한번만 하겠다는 것
            //플레이어에게 해당 스크립트 파일이 연결되어 있으므로 this 를 이용해 플레이어에게 힘을 가한다.
             this.GetComponent<Rigidbody>().AddForce(Vector3.up*jumpSpeed,ForceMode.Impulse);
         }
    }

    private bool IsGrounded(){
        //CheckCapsule 함수를 이용해 플레이어(캡슐)이 땅과 충돌해 있는지를 반환한다.
         return Physics.CheckCapsule(col.bounds.center,new
Vector3(col.bounds.center.x,col.bounds.min.y,col.bounds.center.z),col.radius*0.9f,groundLayers);
    }
```

2) 1인칭 시점 구현 (캡쳐 생략)

FPSControll.cs

```
    void RotCtrl(){
        //마우스의 이동을 받아옴
        //rotSpeed는 회전을 얼마나 빨리 할 지를 정하는 변수.
        float rotX = Input.GetAxis("Mouse Y")*rotSpeed;
        float rotY = Input.GetAxis("Mouse X")*rotSpeed;

        //스크립트가 연결된 object 안의 로컬 축 회전을 갱신
        //y축에 rotY값이 갱신되도록 표현. 실제로는 좌 우 회전 제어.
        this.transform.localRotation*=Quaternion.Euler(0,rotY,0);

        //연결된 카메라의 트랜스폼의 로컬 축 회전 갱신
        //x축을 기준으로 회전 -> 위아래로 회전. - 로 한 이유는 방향을 바꿔주기 위함
        //카메라가 transform.localRotation 기준 -0.6 에서 0.6 사이만을 볼 수 있도록 하기 위해(화면상 약 -90도 ~ +90도) if 문으로 제어
        if(fpsCam.transform.localRotation.x>-0.6&&fpsCam.transform.localRotation.x<0.6){
            fpsCam.transform.localRotation*=Quaternion.Euler(-rotX,0,0);
        }
        else if(fpsCam.transform.localRotation.x<-0.6&&-rotX>0){
            fpsCam.transform.localRotation*=Quaternion.Euler(-rotX,0,0);
        }
        else if(fpsCam.transform.localRotation.x>0.6&&-rotX<0){
            fpsCam.transform.localRotation*=Quaternion.Euler(-rotX,0,0);
        }
    }
```



3) 인벤토리 구현 및 활용, 아이템, 수도꼭지 선택

(1) 인벤토리 구현

인벤토리에 어떤 아이템을 넣어야 할지를 직관적으로 알기 위해 아이템 형식을 정의하여 만들었다. 아이템 형식에는 아이템의 이름, 아이템의 이미지, 아이템의 프리팹, 아이템의 타입이 있음. 아이템 타입은 enum으로 정의된 타입 중에서 선택해야한다.

Item.cs

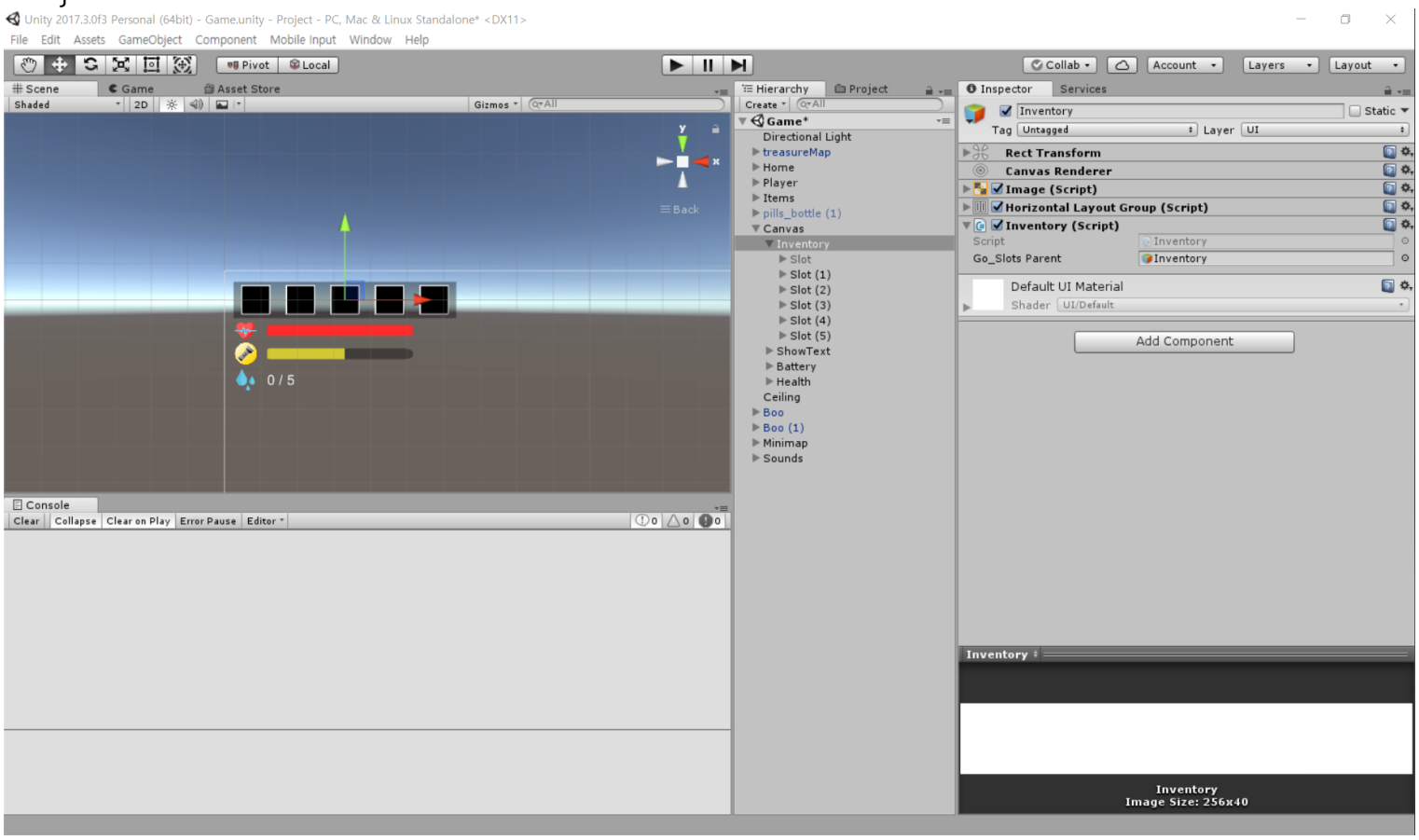
```
[CreateAssetMenu(fileName = "New Item", menuName = "New Item/item")]
public class Item : ScriptableObject
{

    public string itemName;//아이템 이름
    public Sprite itemImage; //아이템의 이미지
    //이미지는 canvas 에서만 띄울 수 있음. sprite 는 canvas 필요 없이 world 상에서 이미지를 띄울 수 있음
    public GameObject itemPrefab; //아이템의 프리팹
    public ItemType itemType;//아이템 타입
    public enum ItemType
    {
        Used,
        Bed,
        Battery,
        Map,
        Etc
    }
}
```

UI로 구현한 인벤토리를 제어할 수 있는 스크립트. go_SlotsParent에는 slot들을 모아둔 slot의 부모 오브젝트이다. slots는 배열이며, go_SlotParent 밑에 있는 slot을 slots배열에 넣었다.

Inventory.cs

```
//슬롯이 포함된 부모 오브젝트에 해당하는 변수
    public GameObject go_SlotsParent;
    //각각의 슬롯들
    private Slot[] slots;

    private int now = 0;

    public static string nowItemName;

    // Use this for initialization
    void Start()
    {
        //부모 오브젝트 밑에 포함된 슬롯들을 slots 배열에 넣음
        slots = go_SlotsParent.GetComponentsInChildren<Slot>();
    }

    void Update()
    {
        nowItemName = whatsNow();
    }
    //아이템 습득시
    public void AcquireItem(Item _item)
```

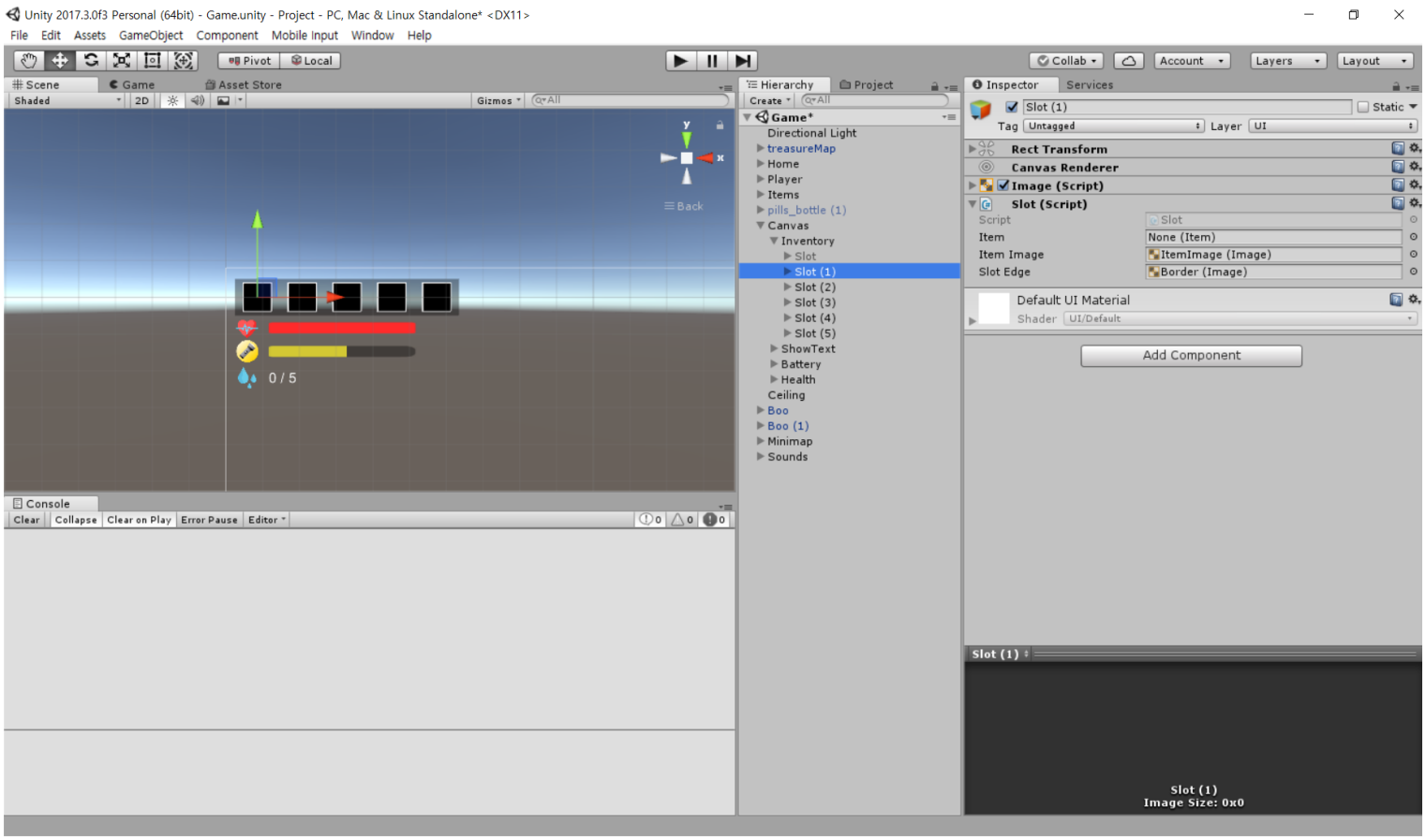
```
    {
        for (int i = 0; i < slots.Length; i++)
        {
            //전체 슬롯 중 비어 있는 슬롯을 찾음
            if (slots[i].item == null)
            {
                //아이템을 넣음
                slots[i].AddItem(_item);
                return;
            }
        }
    }
```



Slot.cs

각각의 slot 을 제어하는 스크립트. Inventory.cs 에서 아이템 습득 시 다음의 AddItem()을 이용하여 실질적으로 인벤토리에 아이템을 넣는 스크립트이다.

```
public Item item; //획득한 아이템
    public Image itemImage; //아이템의 이미지
//아이템 획득
    public void AddItem(Item _item)
    {
        item = _item;
        itemImage.sprite = item.itemImage;

        SetColor(1);
    }
```



(2) 활용

Inventory.cs 와 Item.cs 에서 정의된 내용들을 이용하여 ActionController 에서 아이템을 주웠을 때 인벤토리에 넣을 수 있도록 하였다. 또한 수도꼭지를 가리키고, 수도꼭지를 선택했을 경우 수도꼭지를 잠글 수 있도록 하였다.

ActionController.cs

```
private float range;//아이템 습득 가능한 범위
    private bool pickupActivated = false;//아이템 습득 가능할 시 true
    private RaycastHit hitinfo;//충돌체 정보 저장
//아이템 레이어에만 반응하도록 레이어마스크를 설정

    public LayerMask layerMask;

    //필요한 컴포넌트
    public Text actionText;//행동 텍스트
public Inventory theInventory;
public AudioSource getItem;
    public Camera minimapCamera;
private void CheckItem()
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);

        //Ray 에 어떤 collider 가 교차되면(어떤 오브젝트를 마우스가 가리키면) true 를 반환한다.
        if (Physics.Raycast(ray, out hitinfo, range, layerMask))
        {
            //교차된 물체의 tag 가 Item 이면
            if (hitinfo.transform.CompareTag("Item"))
            {
                ItemInfoAppear();
            }
            //교차된 물체의 tag 가 WaterTap 이면
```

```
            else if(hitinfo.transform.CompareTag("WaterTap"))
            {
                TapInfoAppear();
            }
            else
                InfoDisappear();
        }
        else
        {
            InfoDisappear();
        }
    }
    //아이템의 정보를 사용자에게 보여주는 함수
    private void ItemInfoAppear()
    {
        //습득 가능한 물건이다
        pickupActivated = true;
        //물건 정보를 출력하는 text 를 setActive(true)를 이용하여 표시함
        actionText.gameObject.SetActive(true);
        //text 의 text 를 다음과 같이 정의한다
        actionText.text = "[Item] "+hitinfo.transform.GetComponent<ItemPickUp>().item.itemName +
"\nClick Left Mouse Btn to pick up";
    }
//수도꼭지 정보를 보여주는 함수
    private void TapInfoAppear()
    {
        //수도꼭지를 잠글 수 있다는 표시
        turnoffActivated = true;
        //수도꼭지에 대한 정보를 출력하는 text 표시
        actionText.gameObject.SetActive(true);
        //text 의 내용을 다음과 같이 정의
        actionText.text = "Click Left Mouse Btn to turn off the water";
    }
    //아이템의 정보를 사용자에게 보여주지 않는 함수
    private void InfoDisappear()
    {
        //수도꼭지를 잠그는데 필요한 변수 false(수도꼭지 구현에서 필요)
        turnoffActivated = false;
        //습득 가능한 물건이 아님을 표시
        pickupActivated = false;
        //물건 정보를 출력하는 text 를 감춤
        actionText.gameObject.SetActive(false);
    }

    //아이템을 줍는 함수
    private void CanPickUp()
    {
        //주울 수 있는 물건이라면
        if (pickupActivated)
        {
            //마우스가 물건을 가리키고 있으면
            if (hitinfo.transform != null)
            {
                //마우스가 가리키는 물건의 아이템 타입이 Used 이면
                if(hitinfo.transform.GetComponent<ItemPickUp>().item.itemType==Item.ItemType.Used)
                {
                    //인벤토리에 가리키고 있는 아이템을 넣음(Inventory.cs 에 정의된 AcquireItem()이용)
```

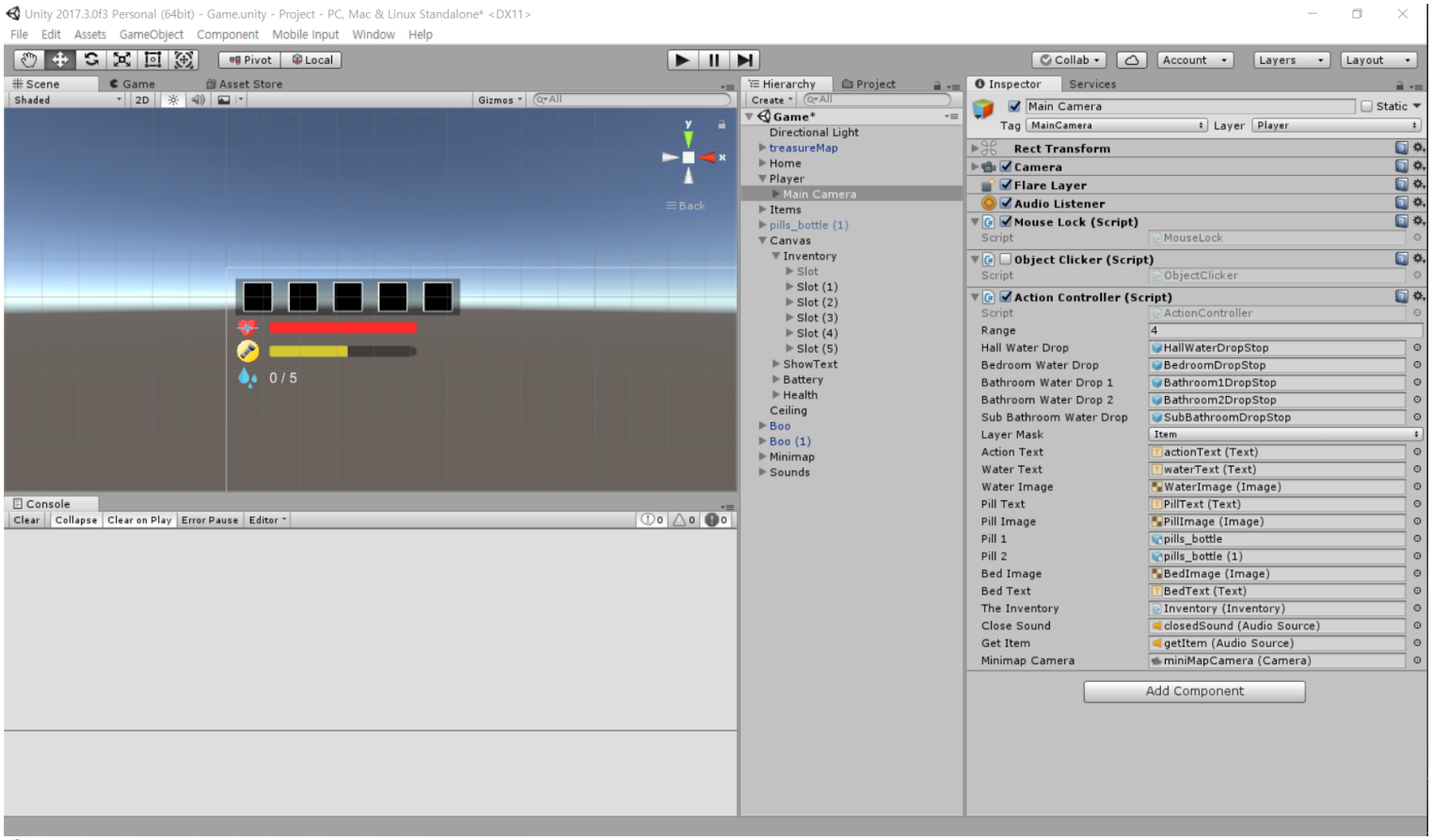
```
                    theInventory.AcquireItem(hitinfo.transform.GetComponent<ItemPickUp>().item);
                    //아이템을 주울 때 나는 소리 재생
                    PlayGetItemSound();
                    //scene 에서 마우스가 가리키고 있던 오브젝트 destroy
                    Destroy(hitinfo.transform.gameObject);
                    //아이템 정보를 보여주지 않도록 함
                    InfoDisappear();
                }
                //마우스가 가리키는 물건의 아이템 타입이 Battery 이면
                else if(hitinfo.transform.GetComponent<ItemPickUp>().item.itemType ==
Item.ItemType.Battery)
                {
                    //batteryBarController.cs 에 선언된 static 변수인 remainingBattery 의 양을 25 만큼
늘림
                    batteryBarController.remainingBattery += 25f;
                    //아이템을 주울 때 나는 소리 재생
                    PlayGetItemSound();
                    //배터리의 양이 100 을 넘지 않도록 함
                    if(batteryBarController.remainingBattery>100f)
                        batteryBarController.remainingBattery = 100f;
                    //scene 에서 마우스가 가리키고 있던 오브젝트 destroy
                    Destroy(hitinfo.transform.gameObject);
                    //아이템 정보를 보여주지 않도록 함
                    InfoDisappear();
                }
                //마우스가 가리키는 물건의 아이템 타입이 Map 이면
                else if(hitinfo.transform.GetComponent<ItemPickUp>().item.itemType ==
Item.ItemType.Map)
                {
                    //미니맵 카메라를 씬에서 보이도록 함
                    minimapCamera.gameObject.SetActive(true);
                    //아이템을 주울 때 나는 소리 재생
                    PlayGetItemSound();
                    //scene 에서 마우스가 가리키고 있던 오브젝트 destroy
                    Destroy(hitinfo.transform.gameObject);
                    //아이템 정보를 보여주지 않도록 함
                    InfoDisappear();
                }
            }
        }
    }
void PlayGetItemSound()
    {
        //해당 사운드 소스가 재생중이 아니라면 사운드를 재생함(겹쳐서 재생되는걸 방지하기 위함)
        if(!getItem.isPlaying)
            getItem.Play();
    }
//수돗물을 잠그는 함수
    private void TurnOff()
    {
        //수돗물을 잠글 수 있을 경우
        if(turnoffActivated)
        {
            //마우스가 물체를 가리키고 있으면
            if(hitinfo.transform!=null)
            {
                //마우스가 가리키는 물체의 이름에 따라 해당 수도꼭지를 잠금
```

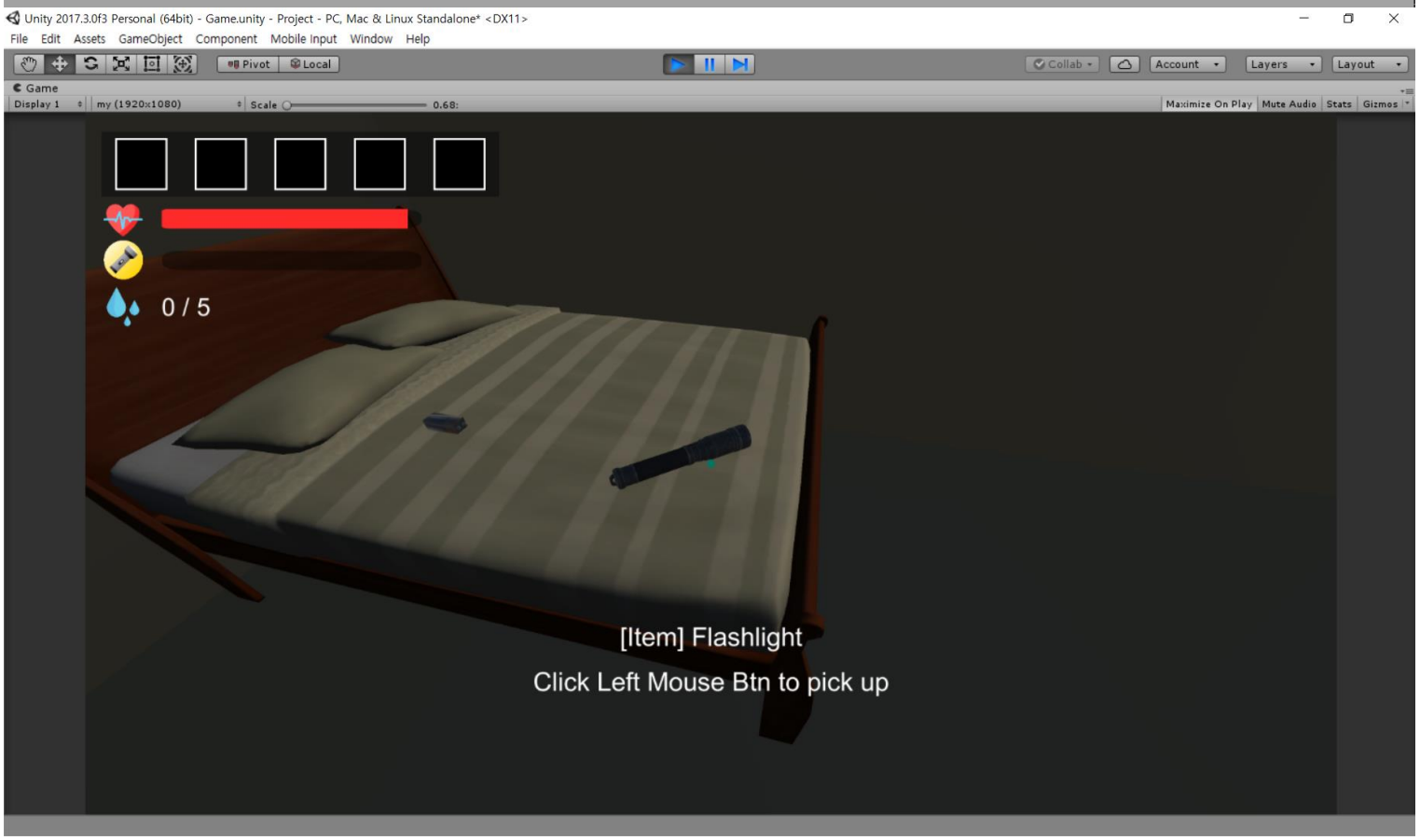
```
            if (hitinfo.transform.name == "HallWaterDropStop")
            {
                //물방울 Prefab 이 계속 생기는 것을 막음
                WaterDrop.drop = false;
                //수도꼭지 잠금 소리 재생
                PlayCloseSound();
                // 한번 잠근 수도꼭지는 다시 제어하지 못하도록 파괴함
                Destroy(hallWaterDrop);
                //수도꼭지 몇 개를 잠갔는지 개수를 표시
                done++;

            }
            else if (hitinfo.transform.name == "SubBathroomDropStop")
            {
                SubBathroomWaterDrop.drop = false;
                PlayCloseSound();
                Destroy(subBathroomWaterDrop);
                done++;
            }
            else if (hitinfo.transform.name == "Bathroom1DropStop")
            {
                BathroomWaterDrop1.drop = false;
                PlayCloseSound();
                Destroy(bathroomWaterDrop1);
                done++;
            }
            else if (hitinfo.transform.name == "Bathroom2DropStop")
            {
                BathroomWaterDrop2.drop = false;
                PlayCloseSound();
                Destroy(bathroomWaterDrop2);
                done++;
            }
            else if (hitinfo.transform.name == "BedroomDropStop")
            {
                BedroomWaterDrop.drop = false;
                PlayCloseSound();
                Destroy(bedroomWaterDrop);
                done++;
            }
            InfoDisappear();
        }
    }
}
```

batteryBarController.cs

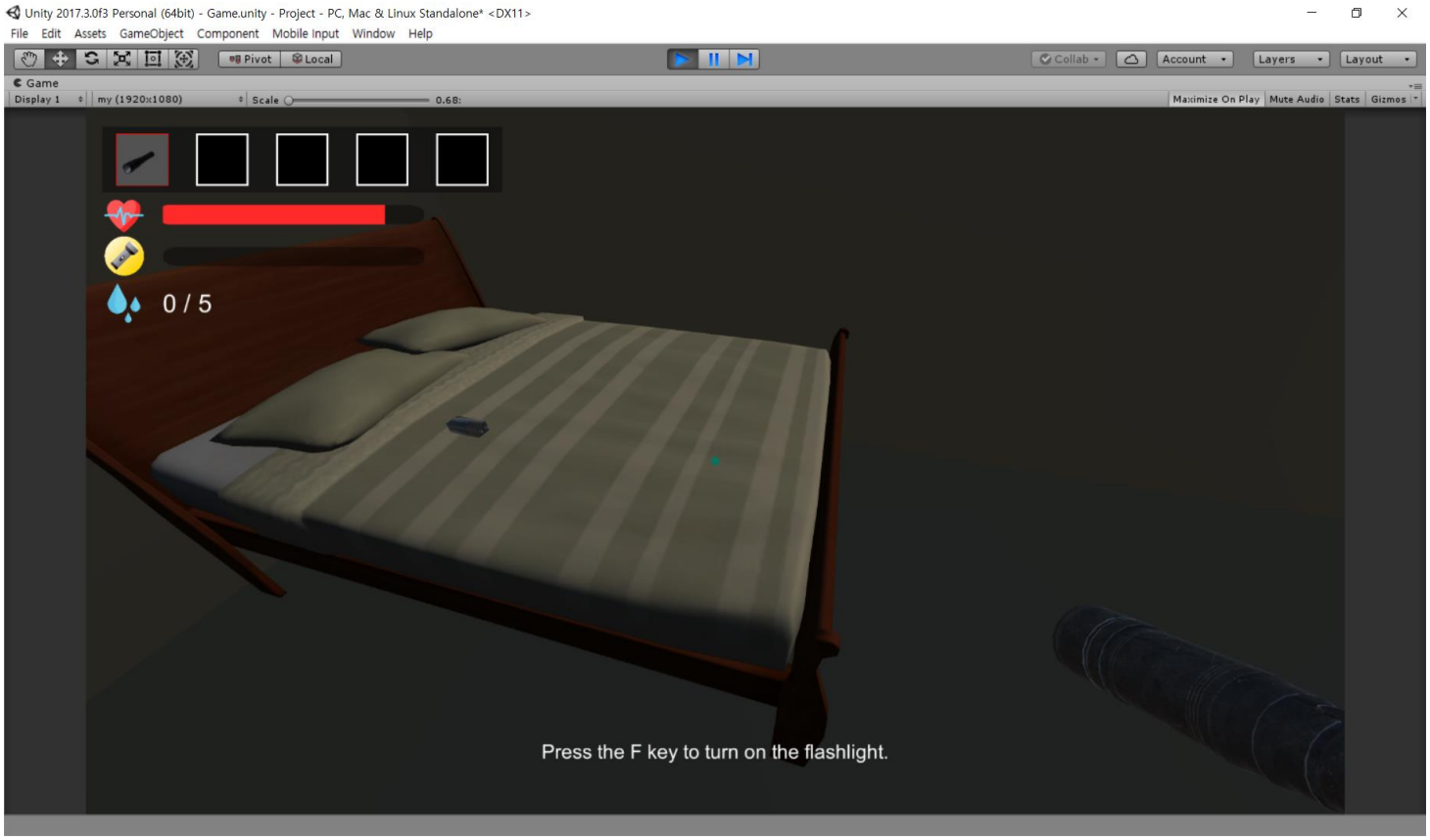
```
Image batteryBar;
float maxBattery = 100f;
public static float remainingBattery;

// Use this for initialization
void Start () {
    batteryBar = GetComponent<Image>();
    remainingBattery = 0f;
}
```

```
    // Update is called once per frame
    void Update () {
        batteryBar.fillAmount = remainingBattery / maxBattery;
    }
```

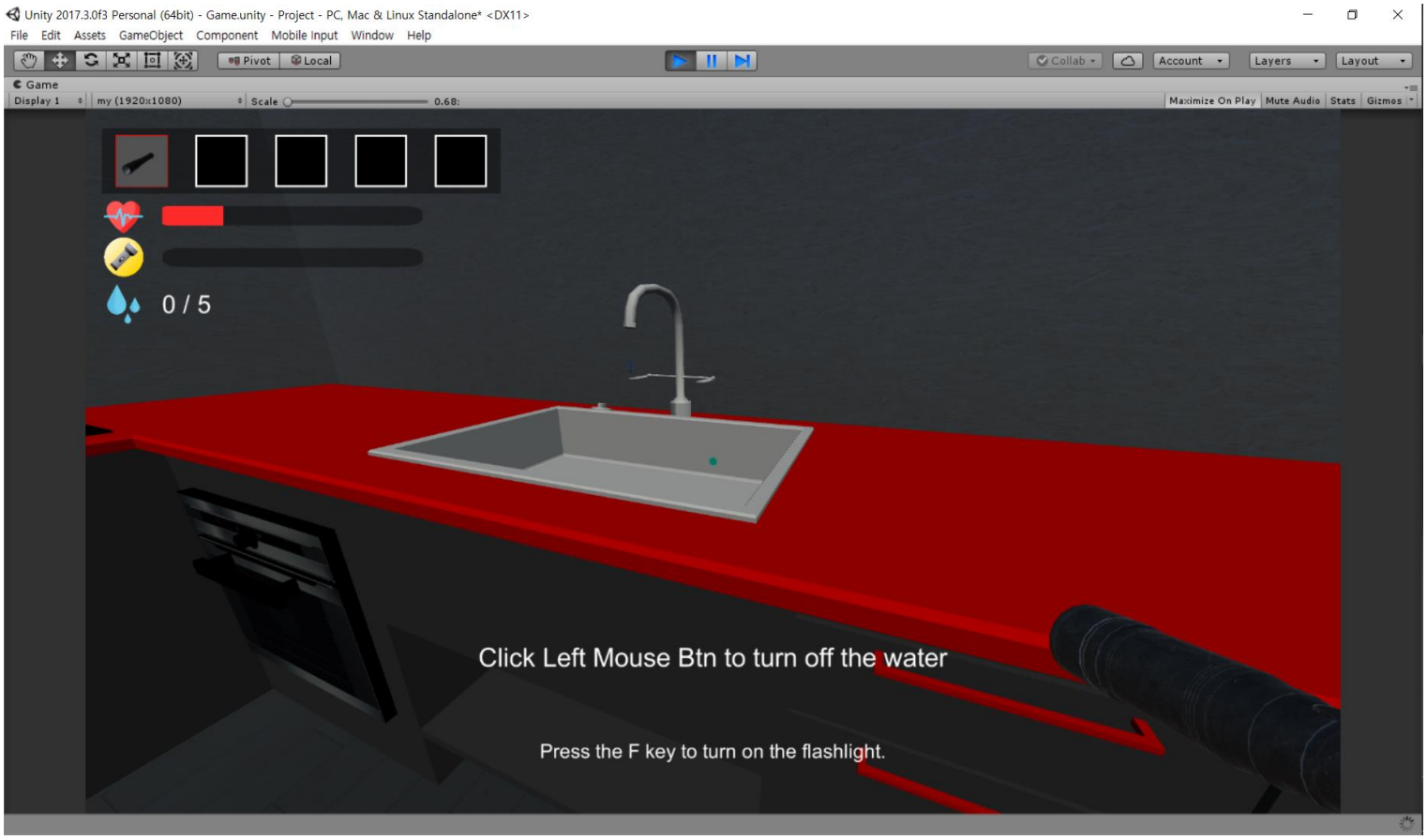




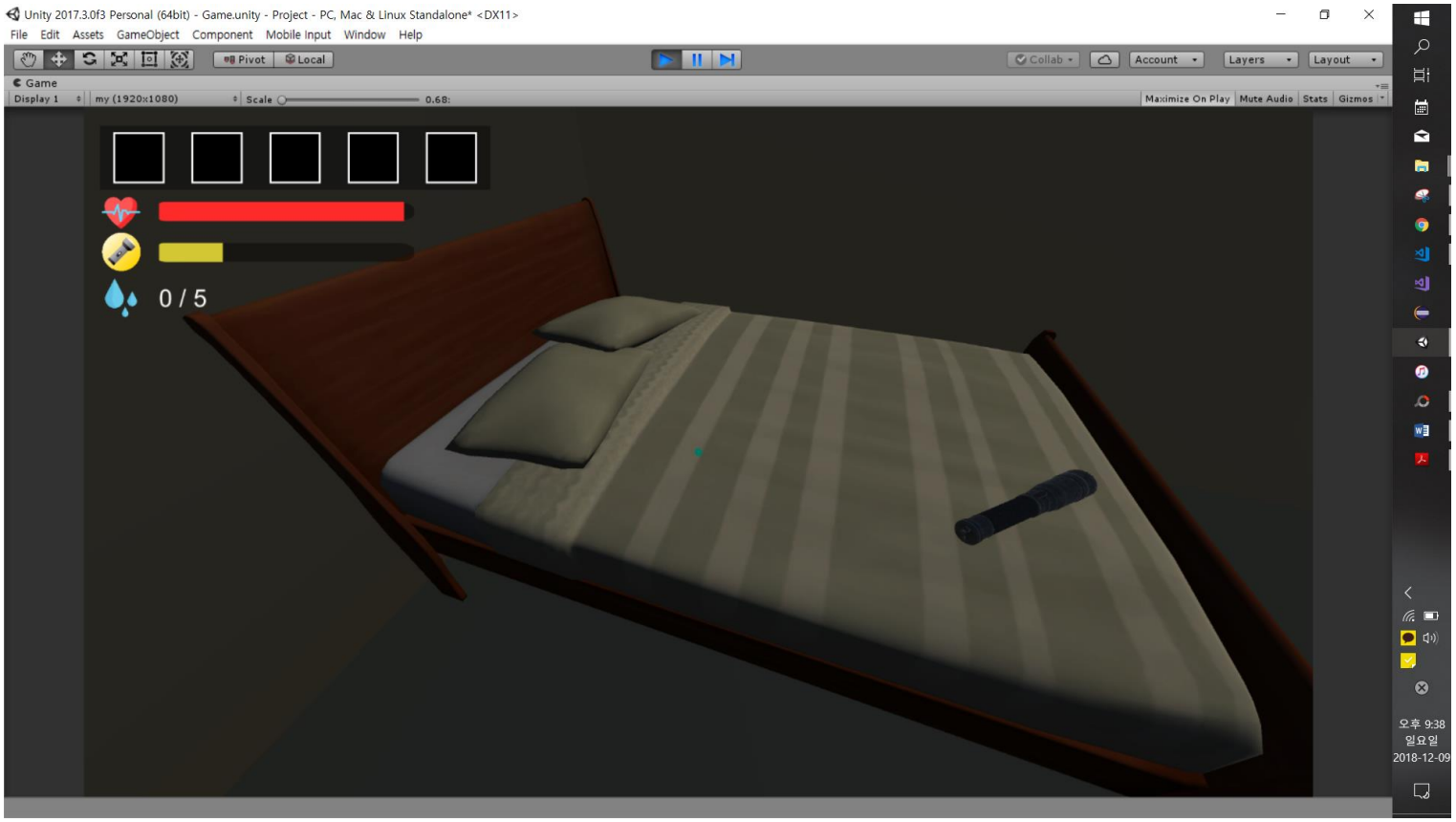
아이템을 획득하면 왼쪽 상단 인벤토리에 해당 아이콘이 들어간다.



수도꼭지를 선택하면 다음과 같이 적절한 안내문이 뜨도록 하였다.



배터리를 얻을 경우 배터리 바에 남은 배터리의 양이 표시된다.



4) 아이템을 잡았을 시 손에 들고 있는 효과

현재 인벤토리에서 어떤 아이템이 선택되어 있는지를 확인하고, 아이템의 이름을 Inventory.cs의 now 변수에 저장한다.

Inventory.cs

```
void Update()
    {
        nowItemName = whatsNow();
    }


public string whatsNow()
    {
        if(ItemCount()!=0)
        {
            return slots[now].item.itemName;
        }
        else return "";
    }
```

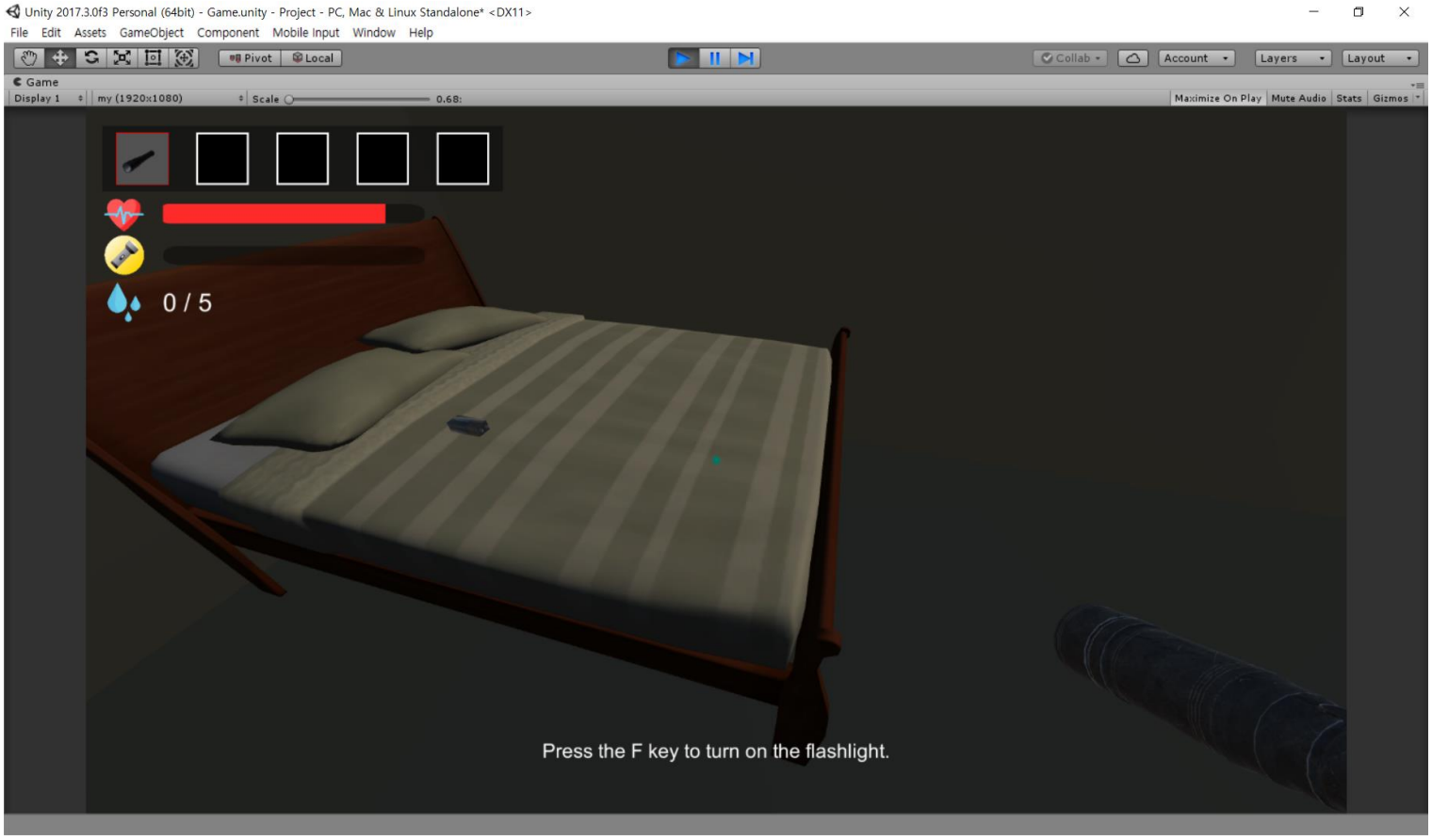
Inventory.cs의 now 변수에 저장된 이름을 통해서 어떤 아이템이 선택되었는지를 판단하고 아이템에 따라 다른 명령을 수행한다. 손전등을 들고 있을 경우 손전등을 들도록 한다. 손전등을 들고 있는 상태에서 손전등이 켜져 있을 경우 F 키를 누르면 손전등이 꺼지고, 손전등이 꺼져 있을 경우 F 키를 누르면 손전등이 켜진다.

FPSControll.cs

```
    void HoldItem()
    {
```

```
    //현재 아이템의 이름이 Flashlight 라면
     if(Inventory.nowItemName=="Flashlight")
     {
        //손에 들고 있는 flashlight 를 SetActive(true)한다.
        flashlight_body.SetActive(true);

        //Flashlight 사용을 돕는 문장 출력
        helpText.gameObject.SetActive(true);

        //손전등이 꺼져 있을 경우

        if (flashlight_light.enabled == false)
        {
            helpText.text = "Press the F key to turn on the flashlight.";
            //F 키를 누르면 손전등이 켜짐
            if (Input.GetKeyDown(KeyCode.F))
                flashlight_light.enabled = true;
        }
        //손전등이 켜져 있을 경우
        else
        {
            helpText.text = "Press the F key to turn off the flashlight.";
            //F 키를 누르면 손전등이 꺼짐
            if (Input.GetKeyDown(KeyCode.F))
                flashlight_light.enabled = false;
        }
    }
     else if(Inventory.nowItemName=="Pill")
     {
         if(ActionController.pilltake==false)
         {
             flashlight_body.SetActive(false);
            flashlight_light.enabled = false;
            helpText.gameObject.SetActive(true);
             if(ActionController.pilltake==false)
                helpText.text = "Press E to take sleeping pills";
             else
                helpText.gameObject.SetActive(false);
            if (Input.GetKeyDown(KeyCode.E))
            {
                ActionController.pilltake = true;
            }
         }
         else
            helpText.gameObject.SetActive(false);

    }
    else
    {
        //손에 아이템이 없다면 손전등과 도움말을 SetActive(false)함
        flashlight_body.SetActive(false);
        helpText.gameObject.SetActive(false);
    }
}
```

인벤토리에서 손전등이 선택되어 있으므로 손전등을 들고 있도록 한다.



손전등을 들고 있는 상태에서 손전등이 꺼져 있을 경우 F 키를 누르면 손전등이 켜진다.

5) 적 생성, 체력

LookAt함수와 transform.Translate함수를 이용하여 적과 플레이어의 거리가 일정 거리 이하이면 적이 플레이어를 따라오도록 구현하였다.

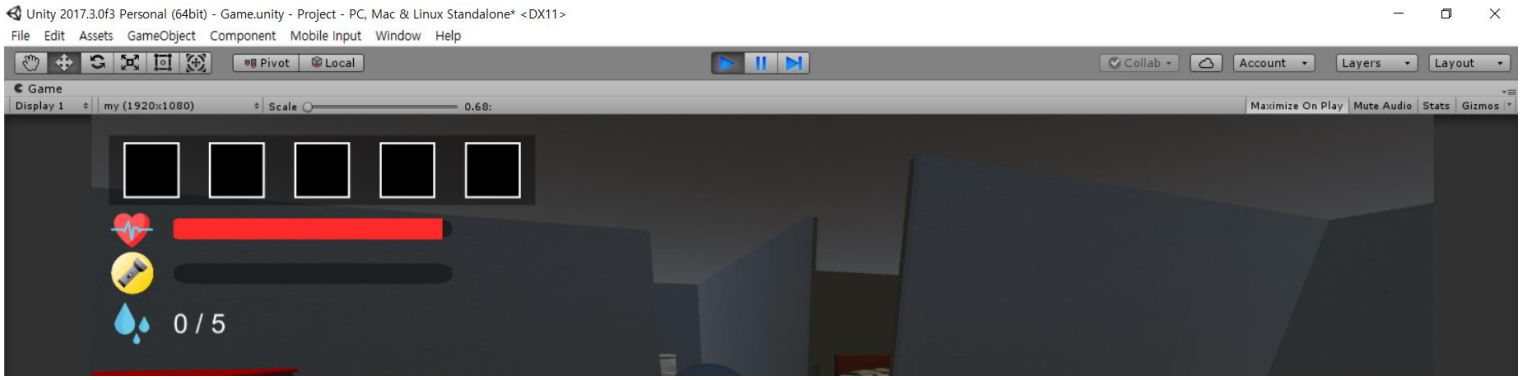
LookAtFollow.cs

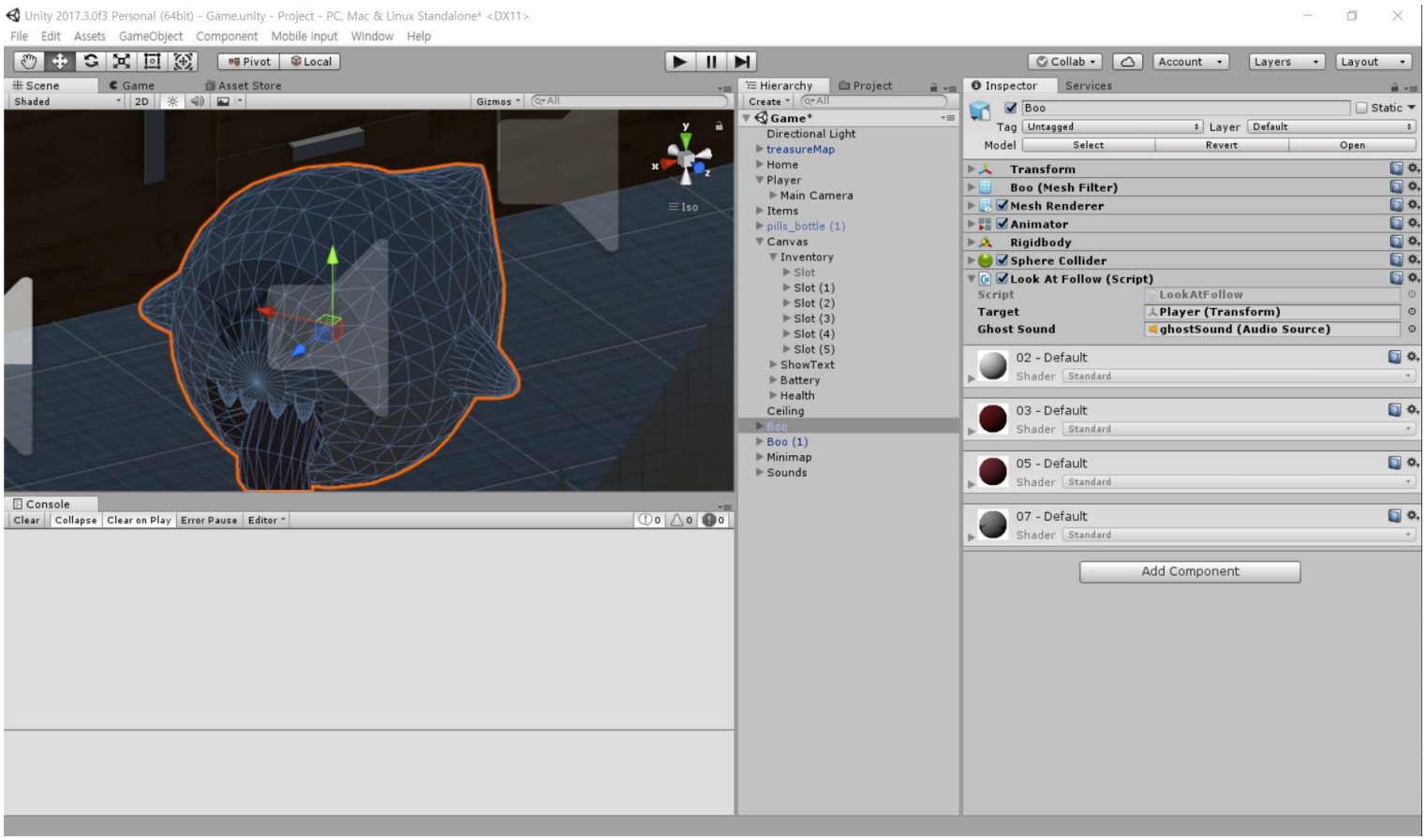
```
void Update()
    {
        //적이 target(플레이어)의 position 을 보도록 한다
        transform.LookAt(target.position);

        //플레이어와 gameobject 사이의 x-y 거리를 측정한다
        xdistance = (transform.position.x - target.position.x);
        zdistance = (transform.position.z - target.position.z);
        distance = Mathf.Sqrt(xdistance * xdistance + zdistance * zdistance);

        //거리가 일정 거리 안에 있을 경우 적이 플레이어를 따라가도록 한다.
        if ((distance > 2f) && (distance < 10f))
        {
            transform.Translate(0.0f, 0.0f, speed * Time.deltaTime);
        }
        attackPlayer();
        PlayGhostSound();
    }
    //적이 플레이어를 공격하는 함수
    void attackPlayer()
    {
        //적과 플레이어간의 거리가 2f 이하일 경우
        if (distance <= 2f)
        {
            //healthBarController 의 static 변수 remainingHealth 에 접근하여 플레이어의 체력을 깎음
            healthBarController.remainingHealth -= Time.deltaTime * 5;
        }
    }
    //적 소리 내기
    void PlayGhostSound()
    {
        //적과 플레이어간의 거리가 10f 이하일 경우
        if(distance<10f)
        {
            //사운드를 재생한다
            if(!ghostSound.isPlaying)
                ghostSound.Play();
        }
        else
            ghostSound.Stop();
    }
```

체력 바를 UI 이미지를 이용하여 구현하였다.

healthBarContoller.cs

```
Image healthBar;
    float maxHealth = 100f;
    public static float remainingHealth;
    // Use this for initialization
    void Start () {
        healthBar = GetComponent<Image>();
        remainingHealth = 100f;
    }

    // Update is called once per frame
    void Update () {
        healthBar.fillAmount = remainingHealth / maxHealth;
    }
```



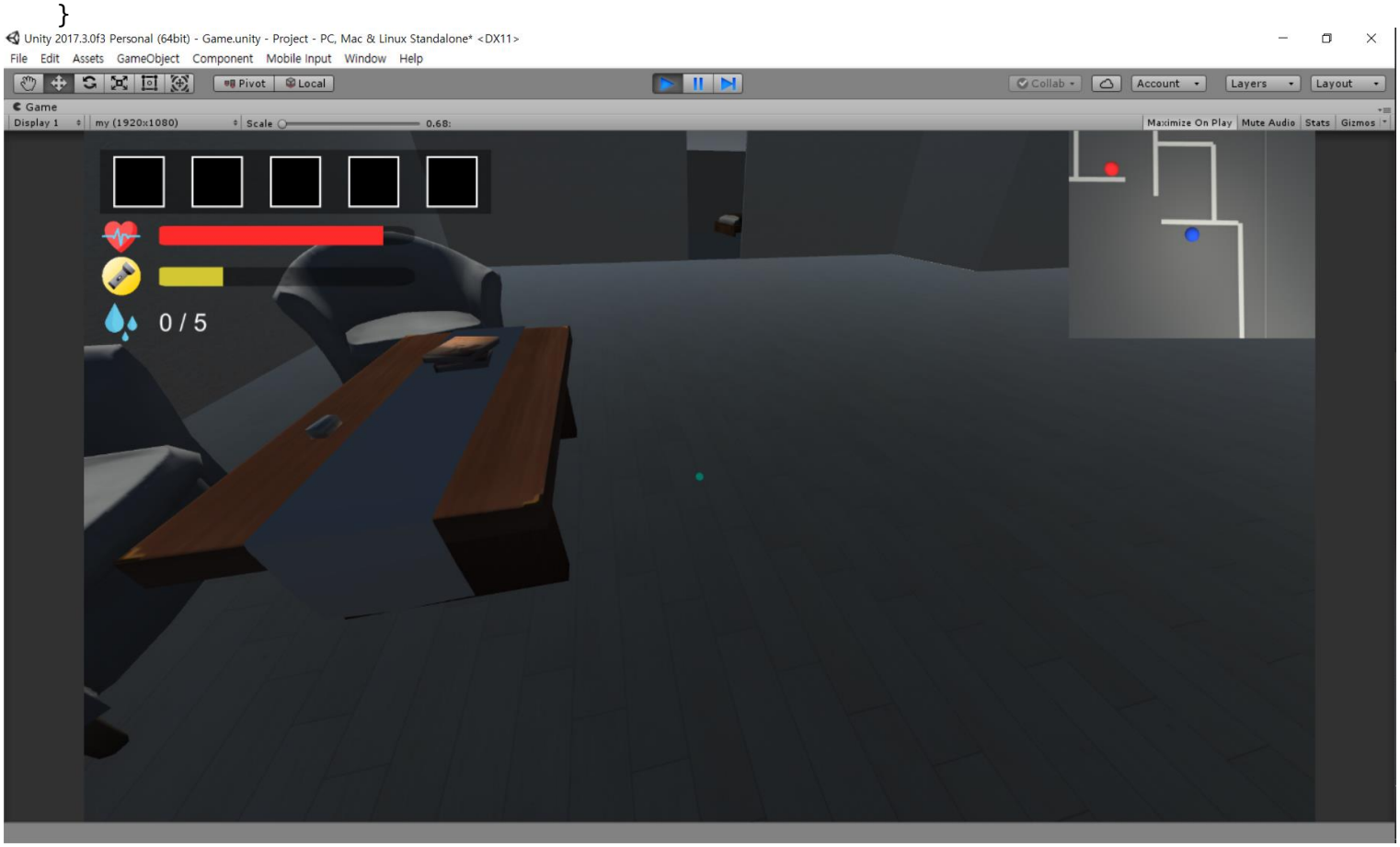


6) 지도(미니맵)

실제로 플레이하는 맵 위에 미니맵을 위치시킨 후, 미니맵에 위치한 오브젝트들이 실제 각각의 오브젝트의 위치를 따라오도록 하여 미니맵을 구현하였다.

miniMapController.cs

```
//실제 적들과 플레이어의 위치를 받아오기 위한 실제 물체들의 transform 변수
    public Transform player;
    public Transform enemy1;
    public Transform enemy2;

    //미니맵에 표시될 적들과 플레이어의 위치를 알려주기 위한 물체들의 transform 변수

    public Transform enemy1Point;
    public Transform enemy2Point;
    public Transform PlayerPoint;

    //실제 플레이어의 위치를 따라가기 위한 미니맵 카메라 변수
    public Transform minimapCamera;


    // Update is called once per frame
    void Update () {
        //실제 적들과 플레이어의 위치를 미니맵에 표시될 오브젝트들이 따라감
        minimapCamera.transform.position = new Vector3(player.transform.position.x,
minimapCamera.transform.position.y, player.transform.position.z);
        enemy1Point.transform.position = new Vector3(enemy1.transform.position.x,
enemy1Point.transform.position.y, enemy1.transform.position.z);
        enemy2Point.transform.position = new Vector3(enemy2.transform.position.x,
enemy2Point.transform.position.y, enemy2.transform.position.z);
        PlayerPoint.transform.position = new Vector3(player.transform.position.x,
PlayerPoint.transform.position.y, player.transform.position.z);
    }
```



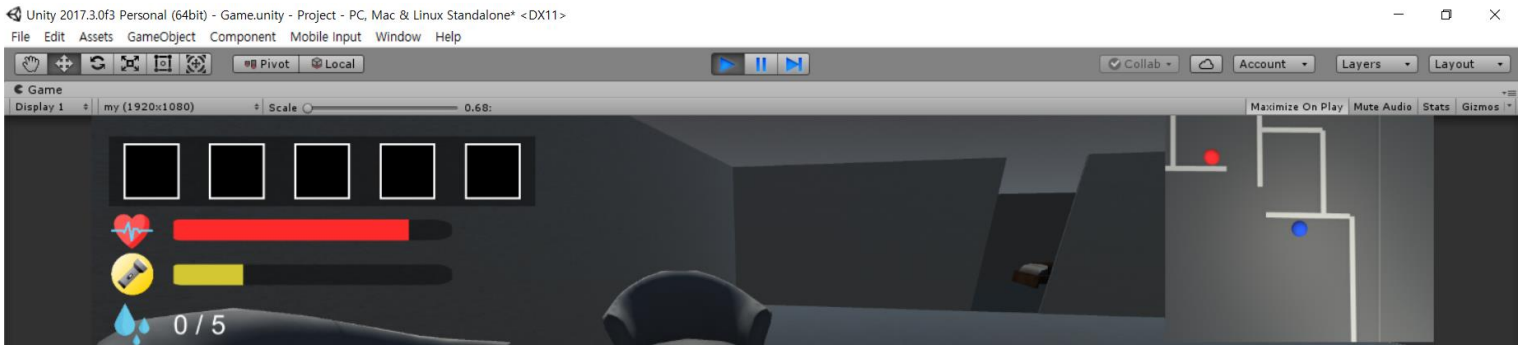
7) 게임 진행 상황 표시

게임 진행 상황을 왼쪽 상단에서 확인할 수 있도록 하여 현재 어떤 미션을 진행해야 하는지를 쉽게 알 수 있도록 함. SetActive()를 사용하여 구현하였다.

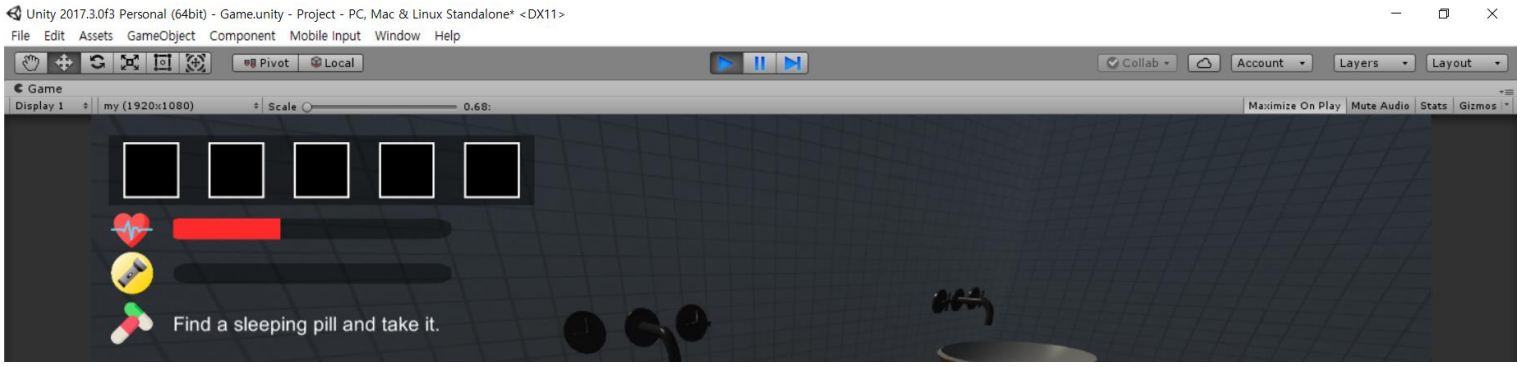
ActionController.cs

```
void setText()
    {
        //수도꼭지 다섯개를 다 잠갔고, 약은 아직 먹지 않은 상태일 경우
        if(done==5&&setPill==false)
        {
            waterImage.gameObject.SetActive(false);
            waterText.gameObject.SetActive(false);
            pillImage.gameObject.SetActive(true);
            pillText.gameObject.SetActive(true);
            pill1.gameObject.SetActive(true);
            pill2.gameObject.SetActive(true);
            setPill = true;
        }
        //수도꼭지 다섯개를 다 잠그지 않은 상태일 경우
        else
        {
            waterText.text = done.ToString() + " / 5";
        }
        //약을 먹은 상태일 경우
        if(pilltake==true)
        {
            pillImage.gameObject.SetActive(false);
            pillText.gameObject.SetActive(false);
            BedImage.gameObject.SetActive(true);
            BedText.gameObject.SetActive(true);
        }
    }
```

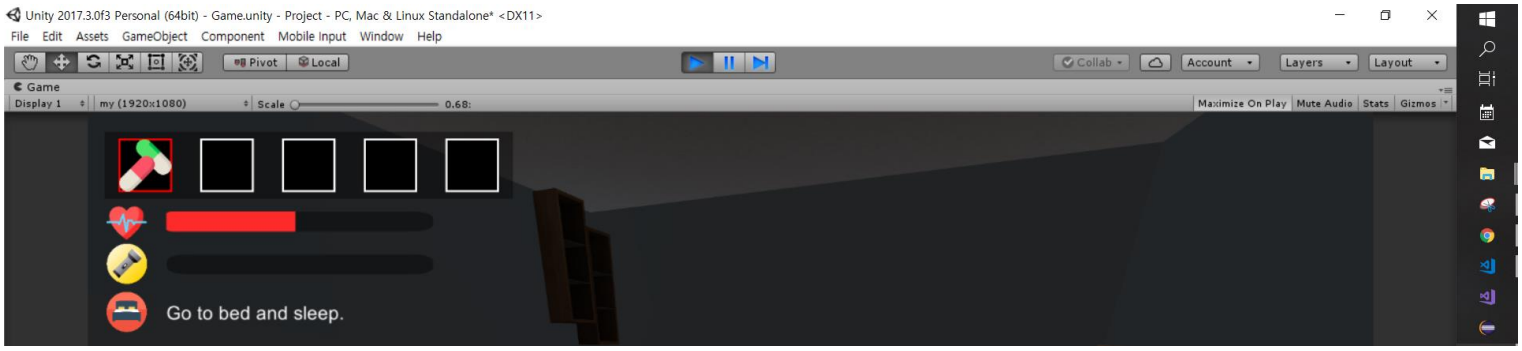
수도꼭지를 다 잠그지 않았을 경우 물방울 모양과 함께 남은 개수를 표시한다.



수도꼭지를 다 잠갔을 경우 약을 찾아서 먹으라는 문구를 표시한다.



약을 찾아서 먹었을 경우 침대로 가서 잠을 자라는 문구를 표시한다.



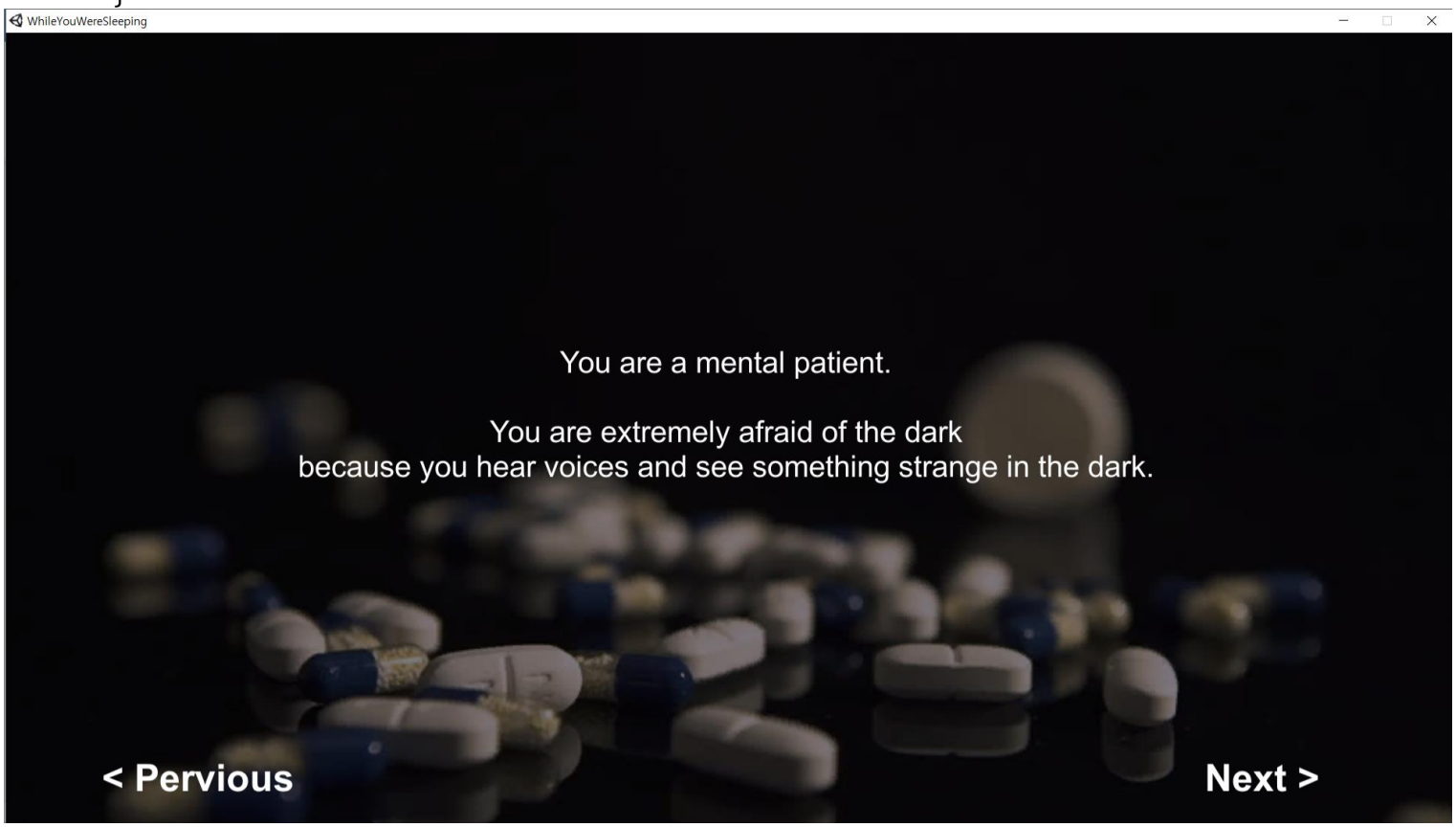
8) 스토리 및 조작 방법 안내

메뉴 화면에서 버튼을 통해 스토리와 조작 방법을 안내하고, 버튼을 통하여 각 씬으로 이동할 수 있도록 함.

Tutorial.cs

```
void ChangeContent()
    {
        //튜토리얼 첫 화면에서 previous를 눌렀을 경우 메인 화면으로 이동
         if(num == 0)
         {
            MainMenu.LoadMenu();
            num = 1;
        }
        //next를 눌렀을 경우 현재 컨텐츠는 숨기고 다음 컨텐츠를 보여줌
        else if (num == 1)
        {
            one.gameObject.SetActive(true);
            two.gameObject.SetActive(false);
        }
        //next를 눌렀을 경우 현재 컨텐츠는 숨기고 다음 컨텐츠를 보여줌
        else if(num==2)
         {
            one.gameObject.SetActive(false);
            two.gameObject.SetActive(true);
            three.gameObject.SetActive(false);
        }
        //next를 눌렀을 경우 현재 컨텐츠는 숨기고 다음 컨텐츠를 보여줌
        else if(num==3)
         {
            two.gameObject.SetActive(false);
            three.gameObject.SetActive(true);
            four.gameObject.SetActive(false);
```

```
    }
    //마지막 컨텐츠일 경우 next 버튼의 text 를 go to main 으로 바꿈
     else if(num==4)
     {
        three.gameObject.SetActive(true);
        four.gameObject.SetActive(true);
        nextText.text = "go to main";
    }
    //마지막 컨텐츠에서 go to main 버튼을 누르면 메인으로 돌아가게 함
     else if(num==5)
     {
        MainMenu.LoadMenu();
        num = 0;
    }
```



MainMenu.cs

```
public void LoadGame()
{
    //게임 화면을 불러옴
    SceneManager.LoadScene("Game");
}

public static void LoadMenu()
{
    //메뉴 화면을 불러옴
    SceneManager.LoadScene("MainMenu");
}
public static void LoadMenu2()
{
    SceneManager.LoadScene("MainMenu");
}

public static void LoadEnd()
{
```

```
    //게임오버 화면을 불러옴
    SceneManager.LoadScene("YouDied");
}

public void LoadTutorial()
{
    //튜토리얼 화면을 불러옴
    SceneManager.LoadScene("Tutorial");
}
```