이름: 김민경

phone: 010-9922-6310

E-mail: 0098121@naver.com

**Escape From Kartell 개발 정리 문서**

(분량 상 생략한 부분은 회색 처리하였습니다.)

**1부. 시작 화면**

1장. 메인화면

1.1. 게임 시작/종료

2장. Tutorial

1.1. 배경 스토리 진행

**2부. 5F**

1장. 감옥 안

1.1. 수프 그릇을 클릭하여 비밀번호 확인

1.2. 비밀번호를 입력하여 탈출

2장. 감옥 밖

2.1. 경비병과 조우

2.2. 카드키 습득

2.3. 카드키 사용\_씬 체인지

**3부. 4F**

1장. 화장실

1.1. 양동이 습득

1.2. 양동이 사용

1.2. 파이프 열기\_스프라이트 체인지

1.4 뚜러뻥 습득

1.5. 뚜러뻥 사용

2장. 노란 다이얼 방

2.1. 큐브를 이용해 다이얼 해제\_씬 체인지

3장. 빨간 다이얼 방

3.1. 노란 방 다이얼 해제 전

3.2. 노란 방 다이얼 해제 후\_이전 상태 저장하기

3.3. 금고 열기

3.4. 스위치 켜기\_스프라이트 체인지

**4부. 3F\_Aisle**

1장. 3층 복도\_다급한 효과내기

1.1. 사이렌 사운드

1.2. 빨간 배경

1.3. 사운드 퀴즈\_씬 체인지

**5부. 3F**

1장. 3층 방 안

1.1. 목걸이 습득

1.2. 목걸이 사용(S.O.S 구조 신호 보내기)

1.3. 스포이드 습득

1.4. 스포이드 사용

1.5. 시험관\_스프라이트 체인지

1.6. 배터리 습득

1.7. 배터리 사용

**6부. 탈출**

1장. 탈출 스토리

1.1. 탈출 스토리 진행

1.2. 메인 화면으로 돌아가기

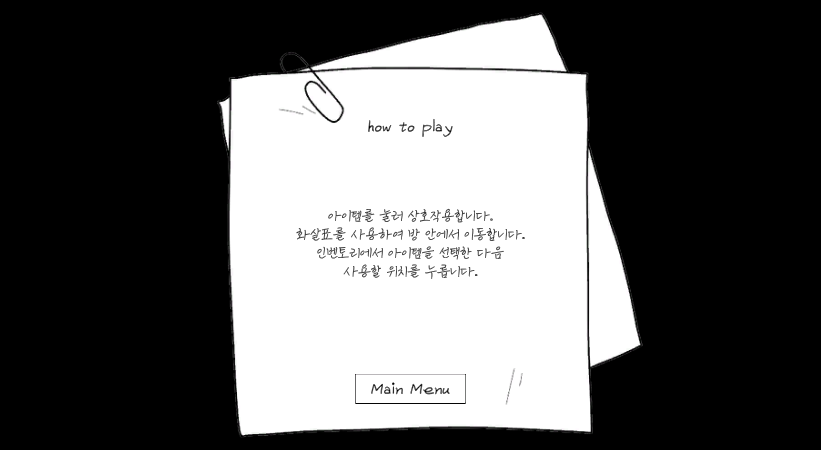
1. **시작화면**

이곳 1부 개요는 메인 화면 구성과 튜토리얼의 플로우를 설명한다.

* 1. **메인화면**
     1. **메인화면 구성**



|  |  |
| --- | --- |
| 코드 | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.SceneManagement;  public class SceneChange : MonoBehaviour  {  public void ChangeStory()  {  SceneManager.LoadScene("StoryScene");  } |
| 설명 | Play 버튼에는 StoryScene으로 넘어가는 ChangeStory함수가 걸려있다. |



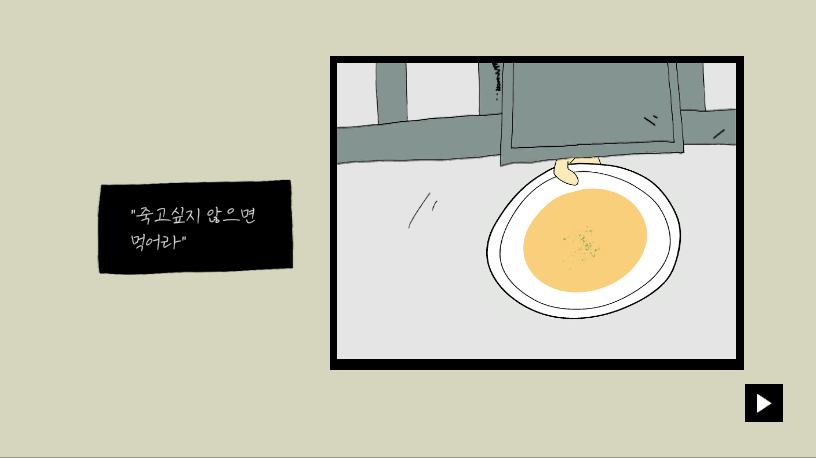
|  |  |
| --- | --- |
| 코드 | using System.Collections;  using System.Collections.Generic;  using UnityEngine;  public class Quit : MonoBehaviour  {  // Update is called once per frame  void Update()  {  }  public void GameQuit()  {  Application.Quit();  }  } |
| 설명 | Exit 버튼에는 게임을 종료하는 GameQuit함수가 걸려있다. |
|  | SceneChange 스프라이트를 추가하여 스크립트를 붙이고, 버튼에 스프라이트를 추가하여 내장 함수 사용. |

BGM은 MainCamera에 Audio Source로 추가하여, Play on Awake 설정으로 시작하면 재생되도록, Loop로 음악이 끝나면 다시 시작하도록 설정하였다.

* 1. **Tutorial**
     1. **배경 스토리 진행**



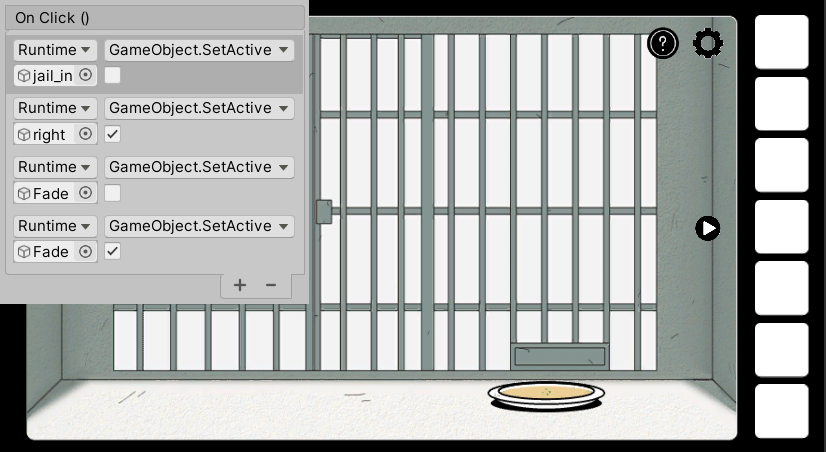




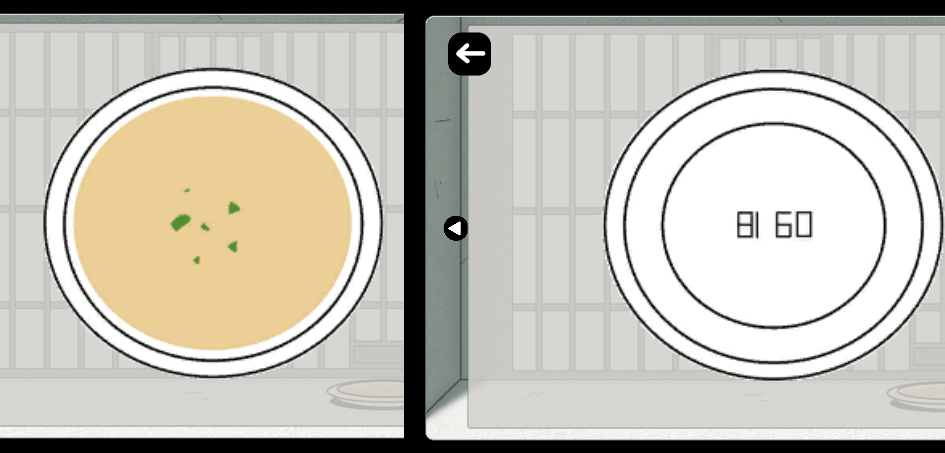
1. **5F**

2부 개요에서는 5층 플로우를 설명한다.

* 1. **감옥 안**
     1. **화면 구성**



* + 1. **수프 그릇을 클릭하여 비밀번호 확인**



* + 1. **비밀번호를 입력하여 탈출**

|  |  |
| --- | --- |
| 코드 | #Password.cs  using System.Collections;  using System.Collections.Generic;  using UnityEngine;  using UnityEngine.UI;  public class passwd : MonoBehaviour  {  public GameObject aisle\_5F;  public Text numbering;  public int num;  public string passward = "0918";  public bool IsOK = false;  public void Onclicked\_Num(int num)  {  if (IsOK == true)  {  numbering.text = "";  }  else  {  numbering.text = num.ToString();  }  if (numbering.text == "")  {  numbering.text = num.ToString();  }  else  {  numbering.text += num;  }  }  public void Onclicked\_0()  {  numbering.text += "0";  } |

|  |  |
| --- | --- |
| 코드 | .  .  public void Onclicked\_9()  {  numbering.text += "9";  }  public void Onclicked\_Cancel()  {  numbering.text = "";  }  public void Onclicked\_OK()  {  if (numbering.text == passward)  {  GameManager.Instance.JailSound();  this.gameObject.SetActive(false);  aisle\_5F.gameObject.SetActive(true);  }  else  {  GameManager.Instance.WrongSound();  numbering.text = "";  }  }  }  #GameManager.cs  public AudioClip wrongSound;  public AudioClip openJail;  public void JailSound()  {  audiosource.clip = openJail;  audiosource.Play();  }  public void WrongSound()  {  audiosource.clip = wrongSound;  audiosource.Play();  } |
| 설명 | 비밀번호를 입력하는 함수 0~9, 입력을 지우는 Cancel 함수, 비밀번호를 입력 후 확인하는 OK함수 구현. 비밀번호가 맞으면 GameManager에서 JailSound함수를 호출하고, 현재 패널을 Off, 5층 복도 패널을 On.  비밀번호가 틀렸을 경우, GameManager에서 WrongSound()함수 호출. JailSound함수는 감옥 문이 열리는 사운드, WrongSound함수는 기계음. |

* 1. **감옥 밖**
     1. **경비병과 조우**



* + 1. **카드키 습득**



|  |  |
| --- | --- |
| 코드 | #TouchArea.cs  public class TouchArea : MonoBehaviour  {  Image touchArea;  public int spaceNo;  public int itemNo;  public int interactNo;  public string dialog\_GetItem;  // Start is called before the first frame update  void Start()  {  touchArea = GetComponent<Image>();  // touchArea.enabled = false;  }  public void TouchedButton()  {  if (GameManager.Instance.isGameStart)  {  Debug.Log("이름 = " + gameObject.name);  GameManager.Instance.TouchCheck(spaceNo, itemNo, dialog\_GetItem);  this.gameObject.SetActive(false);  }  } |
| 설명 | 아이템을 획득하면 획득한 아이템과 글자가 표시되고 인벤토리에 들어간다. |
|  |  |

|  |  |
| --- | --- |
| 코드 | public class GameManager : MonoBehaviour  {  static GameManager instance = null;  public int currentItemNo;  public int currentSpaceNo;//현재 있는 공간  public int currentInteractNo;  public bool isGameStart;  public bool isItemSelected;  public Image getItemIamge;  public GameObject[] space; // 공간을 배열로 저장함  public Sprite[] itemSprite;// 아이템 이미지 모음  public Image[] before;  public int[] space1GetItem;// 얻어야할 아이템  public List<int> space1GotItemList;// 얻어낸 아이템 리스트  public static GameManager Instance//싱글톤  {  get  {  if (null == instance)  {  return null;  }  return instance;  }  }  public void TouchCheck(int spaceNo, int itemNo, string getItem)  {  getItemCount = 0;  if (isTexting) return;  if (itemNo == 0)  {  getItemIamge.gameObject.SetActive(false);  } |

|  |  |
| --- | --- |
| 코드 | else  {  if (spaceNo == currentSpaceNo)  {  switch (spaceNo)  {  case 1:  if (space1GotItemList.Count != 0)  {  for (int i = 0; i < space1GotItemList.Count; i++)  {  if (space1GotItemList[i] == itemNo)  {  getItemCount++;  }  }  if (getItemCount == 0)// 중복된 것이 없으니 등록  {  space1GotItemList.Add(itemNo);  getItemIamge.sprite = itemSprite[itemNo];  getItemIamge.GetComponent<Image>().SetNativeSize();  getItemIamge.gameObject.SetActive(true);  int crNo = space1GotItemList.Count - 1;  GameManager.Instance.itemButton[crNo].ItemButtonOn(space1GotItemList[crNo]);  TalkTexting(getItem);  }  else  {  getItemIamge.gameObject.SetActive(false);  }  break;  } |

|  |  |
| --- | --- |
| 코드 | else  {  getItemIamge.sprite = itemSprite[itemNo];  getItemIamge.gameObject.SetActive(true);  space1GotItemList.Add(itemNo);  int crNo = space1GotItemList.Count - 1;  GameManager.Instance.itemButton[crNo].ItemButtonOn(space1GotItemList[crNo]);  TalkTexting(getItem);  }  break;  .  .  (생략)  .  .  public void TalkTexting(string dial)  {  talkGUI.TalkText(dial);  talkGUI.gameObject.SetActive(true);  isTexting = true;  Invoke("TalkGUIOff", 1.5f);  } |
| 설명 | Touchedbutton 함수는 싱글톤으로 GameManager에 접근해 카드키에 설정되어 있는 spaceNo, itemNo, dialog\_GetItem 값을 TouchCheck 함수에 전달한다. TouchCheck 함수는 itemNo가 0번인지 확인해 0번이면 getItem을 끄고 있고, 0번이 아니면 spaceNo와 현재 spaceNo가 같은 지 확인하고 현재 spaceNo의 case문을 수행한다.  먼저 아이템이 인벤토리에 이미 있는지 체크를 하고 중복되면 아이템을 SetActive(false)로 둔다. 중복되지 않으면 얻은 아이템 리스트(space1GotItemList)에 추가한다.  그리고 ItemButton.cs의 ItemButtonOn함수에 인자를 전달하고, getItem을 켜서 이미지와 dialog를 화면에 띄운다. |

|  |  |
| --- | --- |
| 코드 | public class ItemButton : MonoBehaviour  {  // Start is called before the first frame update  public Button button;  public Image itemIcon;  public int itemNo;  public int count;  public bool isDoorOpen;  void Awake()  {  button = GetComponent<Button>();  itemIcon = transform.GetChild(0).GetComponent<Image>();  }  public void ItemButtonOn(int iconNo)  {  Debug.Log("ItemButtonON확인");  button.interactable = true;  itemNo = iconNo;  itemIcon.sprite = GameManager.Instance.itemSprite[iconNo];  } |
| 설명 | ItemButtonOn함수는 ItemIcon의 이미지를 itemSprite[iconNo]로 변경한다.ItemSprite 배열에는 TouchArea에서 설정한 아이템번호대로 이미지가 저장되어 있습니다.  ItemIcon은 ItemButton의 하위 오브젝트로 Awake의 transform.Getchild(0). GetCompoment<Image>;로 접근한다. |

* + 1. **카드키 사용\_씬 체인지**

|  |  |
| --- | --- |
| 코드 | #TouchArea.cs  public void InteractButton()  {  if (!GameManager.Instance.isItemSelected)  {  GameManager.Instance.InteractCheck(0);  }  else  {  GameManager.Instance.InteractCheck(interactNo);  }  }  #GameManager.cs  public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  case 1:  audiosource.clip = dooropen;  audiosource.Play();  Invoke("CardUse", 1f);  break;  } |
| 설명 | 카드키 클릭 시 IsItemSelected가 체크되고, 카드키에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. |

|  |  |
| --- | --- |
| 코드 | #GameManager.cs  public Sprite[] changeImage;  public void CardUse()  {  handle.SetActive(false);  before[0].sprite = GameManager.Instance.changeImage[0];  before[0].GetComponent<RectTransform>().sizeDelta = new Vector2(255, 275);  before[0].GetComponent<RectTransform>().anchoredPosition = new Vector3(51, -49, 0);  clear.SetActive(true); |
| 설명 | 카드키를 사용하면 문이 열린 이미지로 교체된다.  anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. |

1. **4F**

이곳 3부에서는 4층의 플로우를 설명한다.

* 1. **화장실**
     1. **양동이 습득**

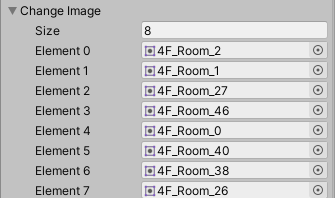
|  |  |
| --- | --- |
| 코드 |  |
| 설명 | 카드키와 구현 방식은 동일하다. |
|  |  |

* + 1. **양동이 사용**

|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  .  .  .  case 2:  audiosource.clip = usewater;  audiosource.Play();  Invoke("WaterBasketUse", 1f);  break; |

|  |  |
| --- | --- |
| 코드 | public GameObject waterbasket;  public void WaterBasketUse()  {  before[1].sprite = changeImage[0];  before[1].GetComponent<RectTransform>().anchoredPosition = new Vector3(147, 103, 0);  before[1].GetComponent<RectTransform>().sizeDelta = new Vector2(800, 800);  waterbasket.SetActive(true);  } |
| 설명 | 양동이 클릭 시 IsItemSelected가 체크되고, 양동이에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. case 2번 WaterBasketUse 함수는 anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. |

* + 1. **파이프 열기\_스프라이트 체인지**



|  |  |
| --- | --- |
| 코드 | #SpriteChange.cs  public class SpriteChange : MonoBehaviour  {  public Image[] before;  public void Pipe()  {  before[0].sprite = GameManager.Instance.changeImage[3];  before[0].GetComponent<RectTransform>().sizeDelta = new Vector2(150, 195);  before[0].GetComponent<RectTransform>().anchoredPosition = new Vector3(-12, 77, 0);  before[1].sprite = GameManager.Instance.changeImage[3];  before[1].GetComponent<RectTransform>().sizeDelta = new Vector2(155, 197);  before[1].GetComponent<RectTransform>().anchoredPosition = new Vector3(-13, 78, 0);  }  #GameManager.cs  public Sprite[] changeImage; |
| 설명 | anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. |

* + 1. **물이 든 양동이 사용**

|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  .  .  .  case 4:  audiosource.clip = poorwater;  audiosource.Play();  Invoke("FillSink", 1f);  break;  } |
| 설명 | 물이 든 양동이 클릭 시 IsItemSelected가 체크되고, 양동이에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. |
|  | 물을 붓는 사운드 재생. |

* + 1. **뚜러뻥 습득**

|  |  |
| --- | --- |
| 코드 |  |
| 설명 | 카드키와 구현 방식은 동일하다. |
|  |  |

* + 1. **뚜러뻥 사용**

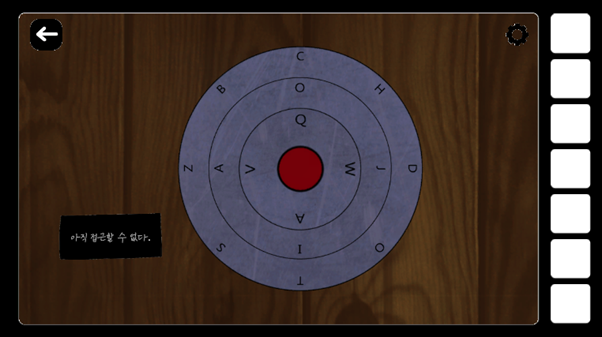
|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  .  .  .  case 3:  audiosource.clip = drainwater;  audiosource.Play();  Invoke("DduruUse", 1f);  break;  } |
| 설명 | 양동이 클릭 시 IsItemSelected가 체크되고, 양동이에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. |
|  | 물이 내려가는 사운드 재생. |

* 1. **노란 다이얼 방**
     1. **큐브를 이용해 다이얼 해제\_씬 체인지**

|  |  |
| --- | --- |
| 코드 | public class dial : MonoBehaviour  {  //public GameObject button1;  public GameObject button2;  public GameObject button3;  public GameObject button4;  public GameObject button5;  public GameObject button6;  public GameObject button7;  (OnClickNo1은 생략)  public void OnClickNo2()  {  GameManager.Instance.RotateDialSound();  button2.transform.Rotate(0.0f, 0.0f, 30.0f);  GameManager.Instance.buttoncnt2++;  }  public void OnClickNo3()  {  GameManager.Instance.RotateDialSound();  button3.transform.Rotate(0.0f, 0.0f, -45.0f);  GameManager.Instance.buttoncnt3++;  }  public void OnClickNo4()  {  button4.transform.Rotate(0.0f, 0.0f, 90.0f);  GameManager.Instance.buttoncnt4++;  }  public void OnClickNo5()  {  button5.transform.Rotate(0.0f, 0.0f, 45.0f);  GameManager.Instance.buttoncnt5++;  } |

|  |  |
| --- | --- |
| 코드 | public void OnClickNo6()  {  button6.transform.Rotate(0.0f, 0.0f, 90.0f);  GameManager.Instance.buttoncnt6++;  }  public void OnClickNo7()  {  button7.transform.Rotate(0.0f, 0.0f, 90.0f);  GameManager.Instance.buttoncnt7++;  }  }  //public int buttoncnt1;  public int buttoncnt2;  public int buttoncnt3;  public int buttoncnt4;  public int buttoncnt5;  public int buttoncnt6;  public int buttoncnt7;  public GameObject dialActivate;  public GameObject dial2Activate;  public int clearNum=0;  void Dialcheck()  {  if (buttoncnt2 % 12 == 11)  {  if (buttoncnt3 % 8 == 4)  {  if (buttoncnt4 % 4 == 3)  {  dial.SetActive(false);  dialActivate.SetActive(false);  isDialClear = true;  PlayerPrefs.SetInt("clearNum", 2);  }  }  }  } |
| 설명 | Rotate로 설정된 값대로 버튼 이미지가 회전하며, 한 번 누를 때마다 buttoncnt값이 1씩 증가한다. 다이얼을 올바르게 설정하면 isDialClear 부울함수가 true가 되고, PlayerPrefs로 clearNum=2로 초기화한다. |
|  |  |

* 1. **빨간 다이얼 방**
     1. **노란 방 다이얼 해제 전**

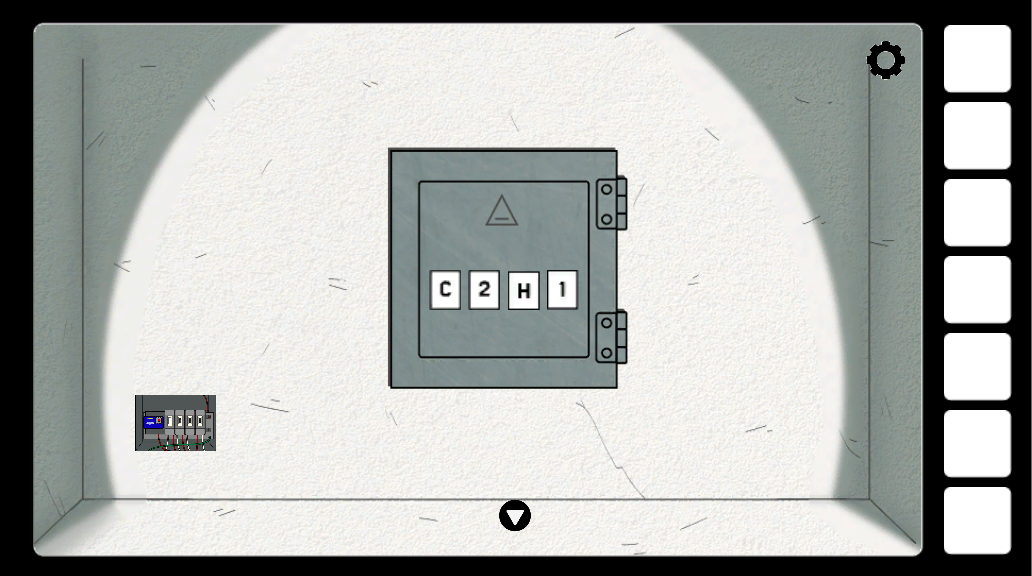


* + 1. **노란 방 다이얼 해제 후\_이전 상태 저장하기**

|  |  |
| --- | --- |
| 코드 | void Start()  {  if (null == instance)  {  instance = this;  }  Debug.Log(clearNum);  if(PlayerPrefs.GetInt("clearNum") == 2)  {  isDialClear = true;  } |

|  |  |
| --- | --- |
| 코드 | if (SceneManager.GetActiveScene().name == "4FScene" && isDialClear == true)// 씬번호와 클리어가 참이면 해당 데이터로 실행  {  Debug.Log(isDialClear + "2");  isDialClear = false;  clearCheck.SetActive(false);  }  else  {  Debug.Log(isDialClear + "3");  isDialClear = false;  }  void Dial2Check()  {  if (buttoncnt5 % 8 == 2)  {  if (buttoncnt6 % 4 == 2)  {  if (buttoncnt7 % 4 == 2)  {  dial2.SetActive(false);  dial2Activate.SetActive(false);  }  }  }  } |
| 설명 | 노란 방의 다이얼이 해제되고 스토리씬으로 씬 체인지가 일어나면 4층 플레이 정보는 초기화되나, clearNum를 PlayerPrefs로 초기화한다면 씬 체인지가 일어나도 정보를 저장할 수 있다. 따라서 clearNum가 2일 경우, isDialClear를 true로 둬서, 이후 if문으로 빨간 방의 다이얼을 사용할 수 있도록 clearCheck 버튼을 Off한다. |

* + 1. **금고 열기**



|  |  |
| --- | --- |
| 코드 | public class Coffer : MonoBehaviour  {  public Image[] before;  public Sprite[] number;  public Sprite[] alpha;  public int numcnt1;  public int numcnt2;  public int alphacnt1;  public int alphacnt2;  private void Update()  {  Check();  } |

|  |  |
| --- | --- |
| 코드 | public void Numberdial1()  {  GameManager.Instance.PressKey();  numcnt1 += 1;  if (numcnt1 % 6 == 0)  {  before[1].sprite = number[0];  }  else if (numcnt1 % 6 == 1)  {  before[1].sprite = number[1];  }  else if (numcnt1 % 6== 2)  {  before[1].sprite = number[2];  }  else if (numcnt1 % 6 == 3)  {  before[1].sprite = number[3];  }  else if (numcnt1 % 6 == 4)  {  before[1].sprite = number[4];  }  else if (numcnt1% 6 == 5)  {  before[1].sprite = number[5];  }  } |

|  |  |
| --- | --- |
| 코드 | public Button cofferbutton;  public void Check()  {  if (alphacnt1 % 6 == 2)  {  if (numcnt1 % 6 == 2)  {  if (alphacnt2 % 6 == 5)  {  if (numcnt2 % 6 == 3)  {  cofferbutton.interactable = true;  before[0].gameObject.SetActive(false);  before[1].gameObject.SetActive(false);  before[2].gameObject.SetActive(false);  before[3].gameObject.SetActive(false);  }  }  }  }  }  } |
| 설명 | 알파벳과 숫자 이미지를 함수로 하나씩 묶어서 if문으로 해당하는 숫자에 맞는 이미지를 배열에 넣어 교체하는 방식. (구현 방식은 alphacnt1,2, numcnt1,2 모두 동일하여 하나만 작성 후 나머지 생략.) |

* + 1. **열린 금고문\_스프라이트 체인지**



|  |  |
| --- | --- |
| 코드 | public void coffer()  {  before[4].sprite = GameManager.Instance.changeImage[5];  before[4].GetComponent<RectTransform>().sizeDelta = new Vector2(555, 345);  before[4].GetComponent<RectTransform>().anchoredPosition = new Vector3(125, 18,0);  before[5].sprite = GameManager.Instance.changeImage[5];  before[5].GetComponent<RectTransform>().sizeDelta = new Vector2(555, 345);  before[5].GetComponent<RectTransform>().anchoredPosition = new Vector3(121, 22, 0);  } |
| 설명 | anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. |

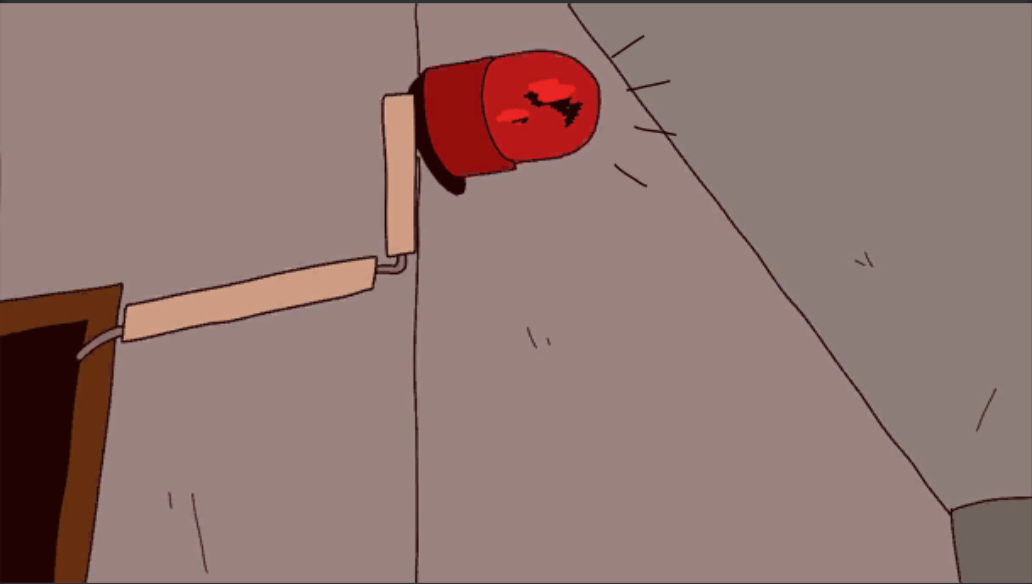
* + 1. **스위치 켜기\_스프라이트 체인지**

|  |  |
| --- | --- |
| 코드 | public void Switch()  {  before[2].sprite = GameManager.Instance.changeImage[6];  before[2].GetComponent<RectTransform>().sizeDelta = new Vector2(9, 30);  before[2].GetComponent<RectTransform>().anchoredPosition = new Vector3(-48, 13, 0);  } |
| 설명 | anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. -값을 주어 에셋을 뒤집어서 스위치가 켜진 컷처럼 연출. |
|  |  |

1. **3F\_Aisle**

이곳 4부에서는 3층 복도에서의 플로우를 설명한다.

* 1. **3층 복도\_다급한 효과내기**



* + 1. **사이렌 사운드**

MainCamera에 Loop로 걸어 패널 이동에도 계속해서 사운드가 실행되도록 함.

* + 1. **빨간 배경**

Fade와 Animator은 동일하게 설정하였으나 Loop로 설정값을 주어 반복적으로 애니메이션 동작이 실행되도록 함.

* + 1. **사운드 퀴즈**

|  |  |
| --- | --- |
| 코드 | #Aisle3F  public class Aisle3F : MonoBehaviour  {  private void Update()  {  Onclicked\_OK();  }  public Text numbering;  public int num;  public string passward = "1241";  public bool IsOK = false;  public void Onclicked\_Num(int num)  {  if (IsOK == true)  {  numbering.text = "";  }  else  {  numbering.text = num.ToString();  }  if (numbering.text == "")  {  numbering.text = num.ToString();  }  else  {  numbering.text += num;  }  }  public GameObject C; |

|  |  |
| --- | --- |
| 코드 | public GameObject D;  public GameObject E;  public GameObject G;  public GameObject A;  public void Cbutton()  {  C.GetComponent<Animation>().Play("calpha");  GameManager.Instance.DoSound();  numbering.text += "0";  }  public void Dbutton()  {  D.GetComponent<Animation>().Play("dalpha");  GameManager.Instance.ReSound();  numbering.text += "1";  }  public void Ebutton()  {  E.GetComponent<Animation>().Play("ealpha");  GameManager.Instance.MiSound();  numbering.text += "2";  }  public void Gbutton()  {  G.GetComponent<Animation>().Play("galpha");  GameManager.Instance.SolSound();  numbering.text += "3";  }  public void Abutton()  {  A.GetComponent<Animation>().Play("aalpha");  GameManager.Instance.RaSound();  numbering.text += "4";  }  public GameObject Clear;  public void Onclicked\_OK()  { |

|  |  |
| --- | --- |
| 코드 | if (numbering.text.Length == 4)  {  if (numbering.text == passward)  {  GameManager.Instance.JailSound();  Clear.SetActive(true);  }  else  {  GameManager.Instance.WrongSound();  numbering.text = "";  }  }  }  }  #GameManager.cs  public void DoSound()  {  audiosource.clip = Do;  audiosource.Play();  }  public void ReSound()  {  audiosource.clip = Re;  audiosource.Play();  }  public void MiSound()  {  audiosource.clip = Mi;  audiosource.Play();  }  public void SolSound()  {  audiosource.clip = Sol;  audiosource.Play();  }  public void RaSound()  {  audiosource.clip = Ra;  audiosource.Play();  } |
| 설명 | 각 버튼에는 순서대로 도, 레, 미, 솔, 라 함수가 걸려있다. 버튼 클릭 시, 해당하는 음이 재생되며, 숫자를 부여해 알맞은 순서대로 입력하면 다음 단계로 넘어갈 수 있도록 구현하였다. |

1. **3F**

이곳 5부에서는 3층의 플로우를 설명한다.

* 1. **3층 방 안**
     1. **목걸이 습득**

|  |  |
| --- | --- |
| 코드 |  |
| 설명 | 카드키와 구현 방식은 동일하다. |

* + 1. **목걸이 사용(S.O.S 신호 보내기)**

|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  .  .  .  case 11:  Pendant();  break;  }  public void Pendant()  {  ImageOn.SetActive(true);  } |
| 설명 | 목걸이 클릭 시 IsItemSelected가 체크되고, 목걸이에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. Pendant함수를 실행한다. |

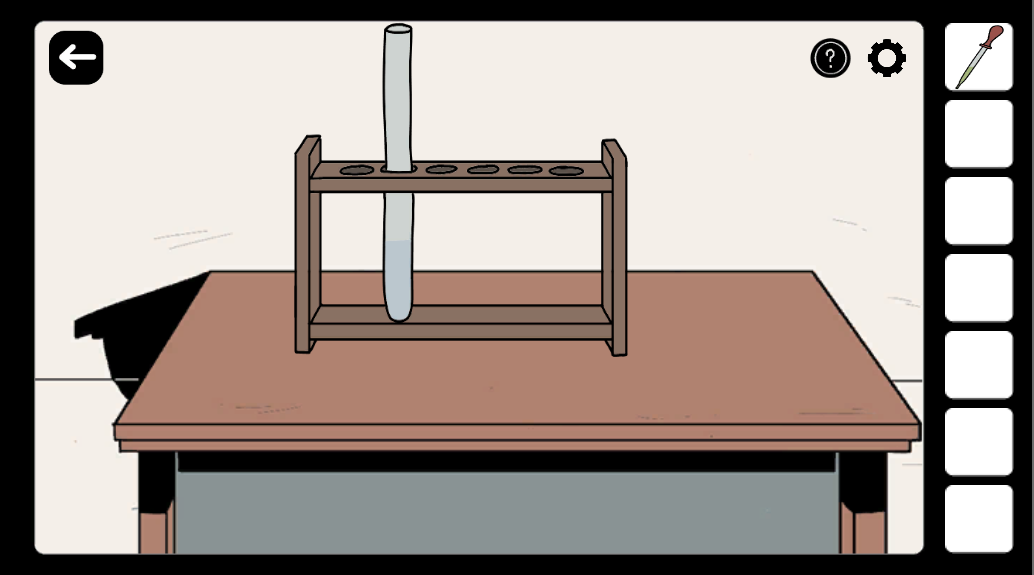
* + 1. **스포이드 습득**

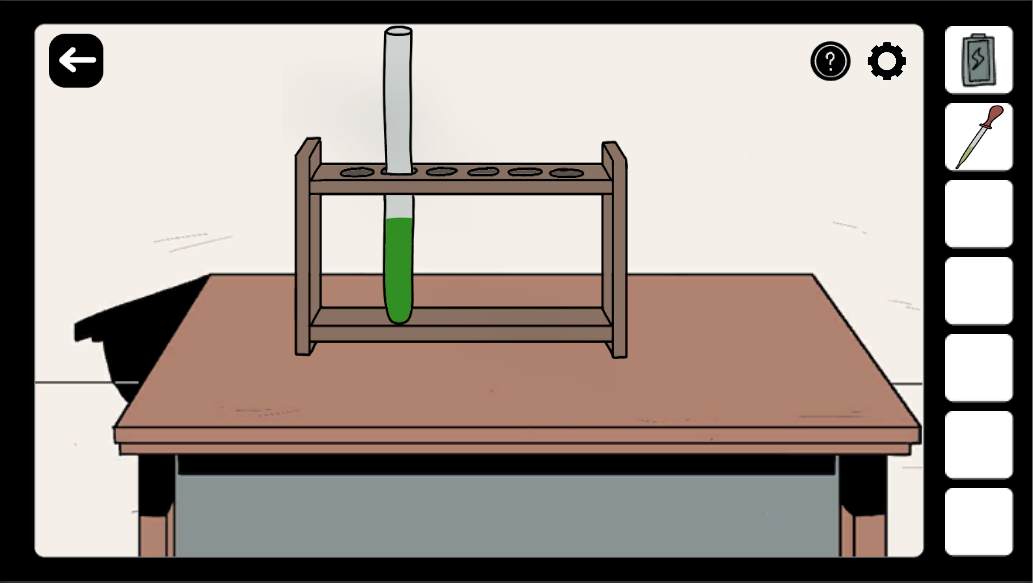
|  |  |
| --- | --- |
| 코드 |  |
| 설명 | 카드키와 구현 방식은 동일하다. |
|  |  |

* + 1. **스포이드 사용**

|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  .  .  .  case 7:  Experiment();  break;  } |
| 설명 | 시험관 클릭 시 IsItemSelected가 체크되고, 시험관에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. Experiment 함수를 실행한다. |
|  |  |

* + 1. **시험관\_스프라이트 체인지**





|  |  |
| --- | --- |
| 코드 | public void Experiment()  {  before[5].sprite = changeImage[8];  before[5].GetComponent<RectTransform>().sizeDelta = new Vector2(445, 445);  before[5].GetComponent<RectTransform>().anchoredPosition = new Vector3(-67, 125, 0);  BImage.SetActive(true);  }  public void Battery()  {  AImage.SetActive(false);  CImage.SetActive(true);  battery.SetActive(true);  } |
| 설명 | anchoredPosition과 sizeDelta로 설정된 값대로 오브젝트의 위치와 사이즈를 변경한다. 체인지 후 BImage 패널을 On.패널 On/Off로 시험관 안의 내용물이 채워지는 모습을 연출한다. |
|  |  |

* + 1. **배터리 습득**

|  |  |
| --- | --- |
| 코드 |  |
| 설명 | 카드키와 구현 방식은 동일하다. |
|  |  |

* + 1. **배터리 사용**

|  |  |
| --- | --- |
| 코드 | public void InteractCheck(int interactNo)  {  currentInteractNo = interactNo;  if (currentItemNo == currentInteractNo)  {  Debug.Log("currentspaceNo" + currentSpaceNo);  Debug.Log("cureentItemNo" + currentItemNo);  Debug.Log("currentInteractNo" + currentInteractNo);  switch (currentInteractNo)  {  case 0:  break;  case 10:  BatteryOn();  break;  }  }  }  public void BatteryOn()  {  Fill.SetActive(true);  batteryon.SetActive(true);  GoToEnding.SetActive(true);  } |
| 설명 | 배터리 클릭 시 IsItemSelected가 체크되고, 배터리에 걸린 TouchArea스크립트에 설정된 아이템 넘버를 매개변수로 GameManager의 InteractButton함수를 호출한다. BatteryOn함수는 Fill패널을 끄고 batteryon패널을 On하며, batteryon패널의 버튼을 누르면 Animator y값 조절로 블라인드가 올라가며 EndingScene으로 씬체인지한다. |

1. **탈출**

이곳 6부에서는 엔딩 플로우를 설명한다.

* 1. **탈출 스토리**
     1. **탈출 스토리 진행**

클릭 시, 패널을 하나씩 On, Off하여 이미지 교체, 마지막 이미지 패널을 상위 트리(GameObject)로 묶고, 헬기 사운드를 걸어서 출력, Animator로 헬기가 올라오는 듯한 연출 구현.

* + 1. **메인 화면으로 돌아가기**

|  |  |
| --- | --- |
| 코드 | public class SceneChange : MonoBehaviour  {  public void ChangeTitle()  {  SceneManager.LoadScene("TitleScene");  }  } |
| 설명 | TitleScene으로 씬 체인지하는 함수. |
|  |  |