

(모바일 콘텐츠 - 기말 미션5 보고서)

유니티 3D 생존 게임\_ 송응열 교수님

# Mysterious-Forest

문화테크노학과 20160241 김미소

문화테크노학과 20160247 오은주

## ▶ 구성원

- 문화테크노학과 20160241 김미소
- 문화테크노학과 20160247 오은주

## ▶ 기획 의도

- '탈출'을 컨셉으로 한 많은 모바일 게임들이 흔히 말하는 '방탈출 게임'과 같은 포맷을 가지고 있다.
- 이러한 이유로 '탈출'이라는 목표 지향적인 게임들 뿐, 스토리 중심적인 게임을 찾기 힘들다.
- 배경 또한 밀폐된 방, 외딴 섬 등 어두운 공간을 주로 사용

=> '탈출'을 컨셉으로 하되 유저들의 감성을 자극할 수 있는 스토리와 밝은 분위기를 가진 새로운 유형의 게임을 제작해보고자 하였다.

## ▶ 게임 시나리오

- 사랑하던 아내를 일찍 떠나보냈지만 그런 아내를 꼭 닮은 딸과 행복하게 살아가던 아빠. 하지만 어느날 딸이 불치병에 걸리고만다. 이 병을 치료할 수 있는 방법은 '신비의 숲'에 있는 신비의 약물을 가져오는 것. 하지만 '신비의 숲'은 한번 들어가며 살아나오기 힘든 악명 높은 숲으로 유명하다. 과연 아빠는 소중한 딸을 위해 무사히 '신비의 약물'을 획득할 수 있을까...?

## ▶ 게임 컨셉

- '신비의 숲'에서 생존 & 탈출의 컨셉을 지닌 < Mysterious Forest >는 캐주얼 풍의 3D 모바일 게임이다. 아름다운 강과 나무, 꽃이 있는 '신비의 숲'을 배경으로 플레이어들은 '신비의 약물'을 얻기 위해 사냥과 채집 등 다양한 미션을 수행해야 한다.

## ▶ 주요 콘텐츠

### 1. Survive

- 강 : 미션을 깨기 위해서는 강을 건너야만 한다. 강에 들어가면 강의 색이 변하고, Player의 Hp가 깎인다.
- 무기 : 각 상황별 알맞은 무기를 사용할 수 있다.  
( ~ : 맨손 / 1 : 칼 / 2 : 방망이 / 3 : 총 / 4 : 곡괭이 )
- 과일 : 강 건너의 과일나무를 베어 과일을 획득하면 다음 단계로 넘어갈 수 있다.
- 바위 : 곡괭이로 바위를 부숴 신비의 물약 제조에 필요한 재료를 얻을 수 있다.

- 꽃 : 일정 개수 이상의 꽃을 획득해 신비의 물약 제조에 필요한 재료를 얻을 수 있다.
- 여우 : 꽃을 가로막는 여우를 방망이로 죽여야 한다. 닿으면 HP가 깎인다.
- 좀비 : 밤이 되면 좀비가 나타난다. 총을 이용해 좀비를 죽여야 한다.
- 약물 : 미션을 다 완수한 후 신비의 물약을 찾으면 최종 탈출을 할 수 있다.

## 2. Game

- GUI 버튼 : 씬이 전환될 때, 버튼을 눌러 자연스럽게 게임을 시작할 수 있다.
- GUI 박스 : 왼쪽 상단에 위치하여 HP와 획득한 꽃, 과일, 바위의 수를 나타낸다.
- 레이블 텍스트 : 레이블 텍스트를 통해 유저가 게임을 원활하게 진행하도록 돕는다.
- 효과음 : 플레이어가 사냥 및 채집을 할 때 효과음이 나온다.
- 인벤토리 : 과일, 물약, 꽃을 얻으면 인벤토리 안에 저장되고, 획득 개수를 알려준다.
- 낮과 밤 : 낮에 신비의 물약 제조에 필요한 꽃과 물약을 모두 모으면 밤이 된다.

### ▶ 게임 조작 방법

- 캐릭터 움직임 조절 : A / S / D / W
- 캐릭터 방향키 조절 : 마우스
- 달리기 : Shift
- 점프 : Space
- 무기 교체 : ~(기본) / 1(칼) / 2(방망이) / 3(총) / 4(곡괭이)
- 인벤토리 창 : I

## □ Scenes 종류

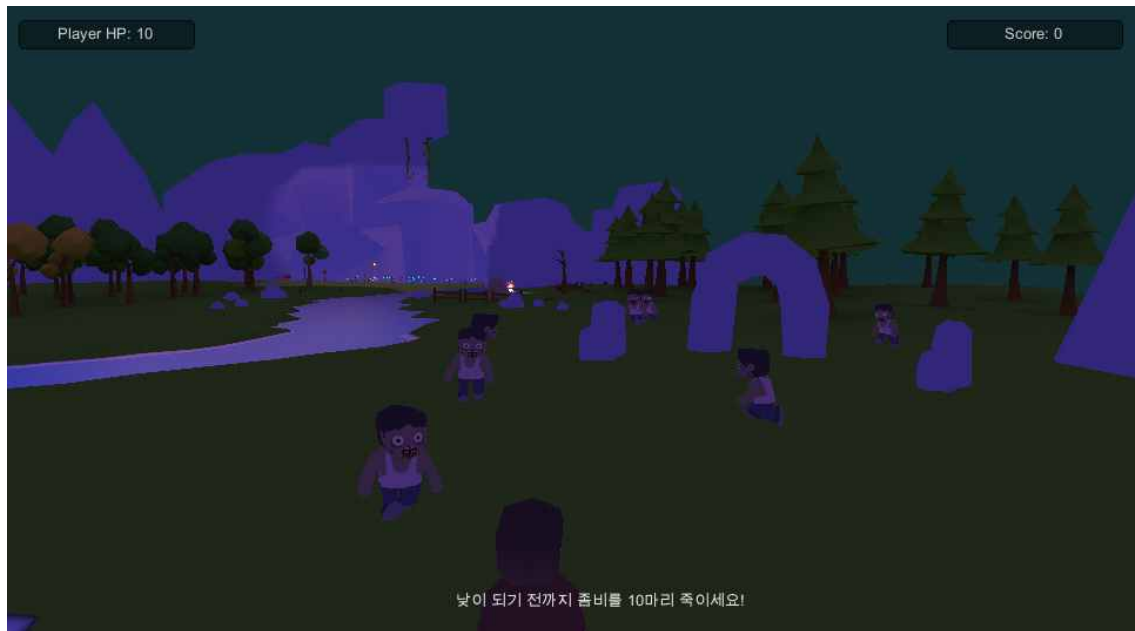
-> 게임 시작 씬 (StartScenes)



-> 메인 게임 씬 낮 (DemoDay)



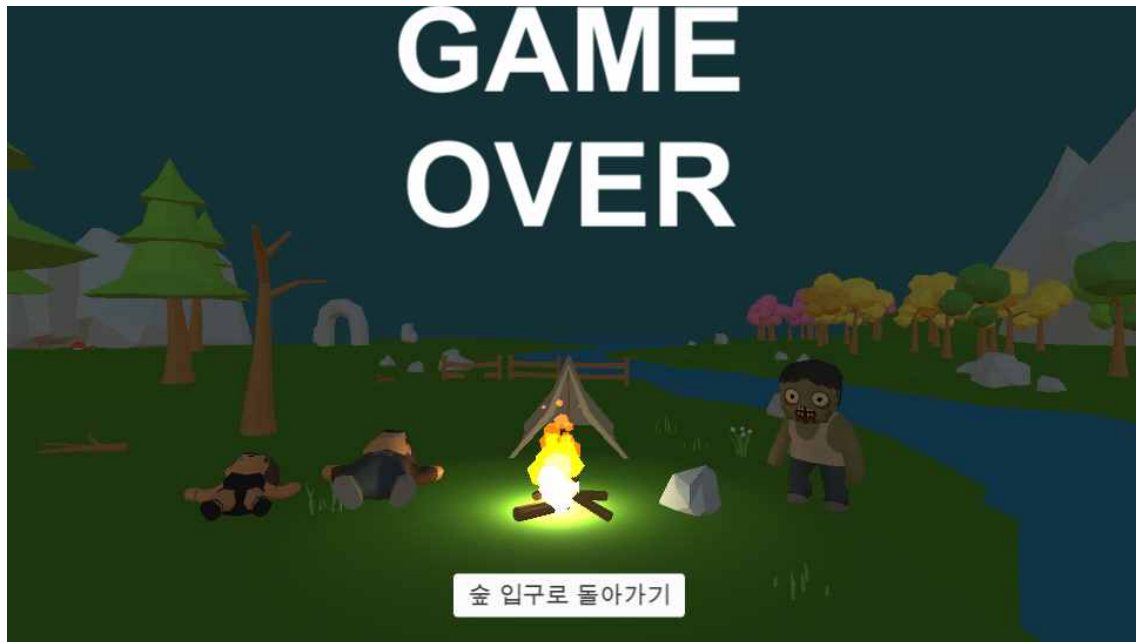
-> 메인 게임 씬 밤 (DemoNight)



-> 게임 종료 씬 낮(OverScene)



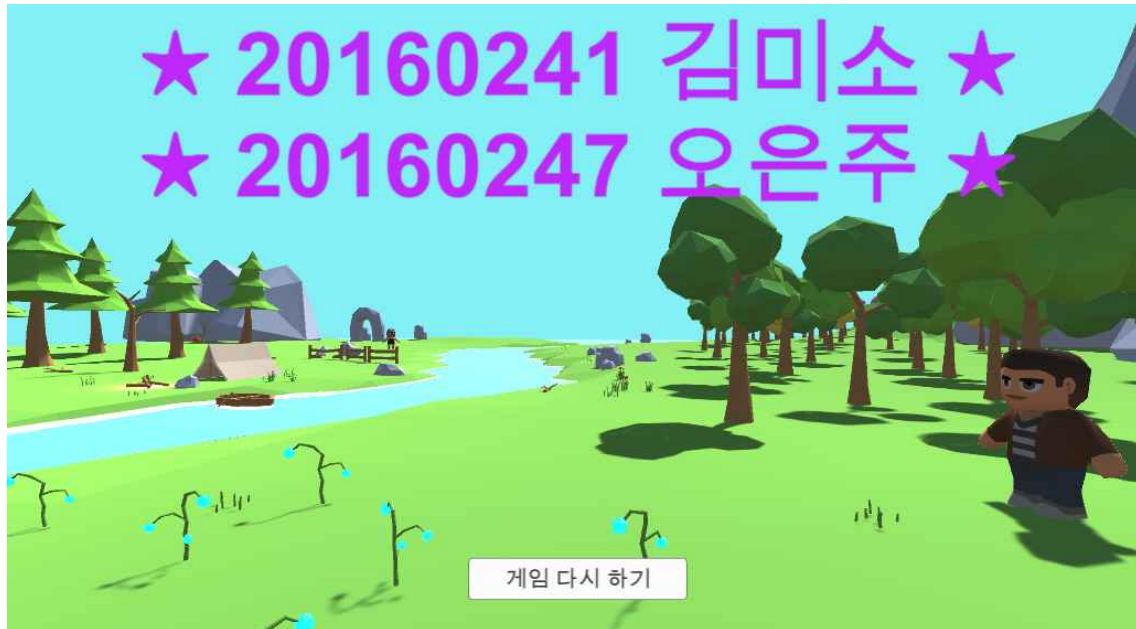
-> 게임 종료 씬 밤 (OverNightScene)



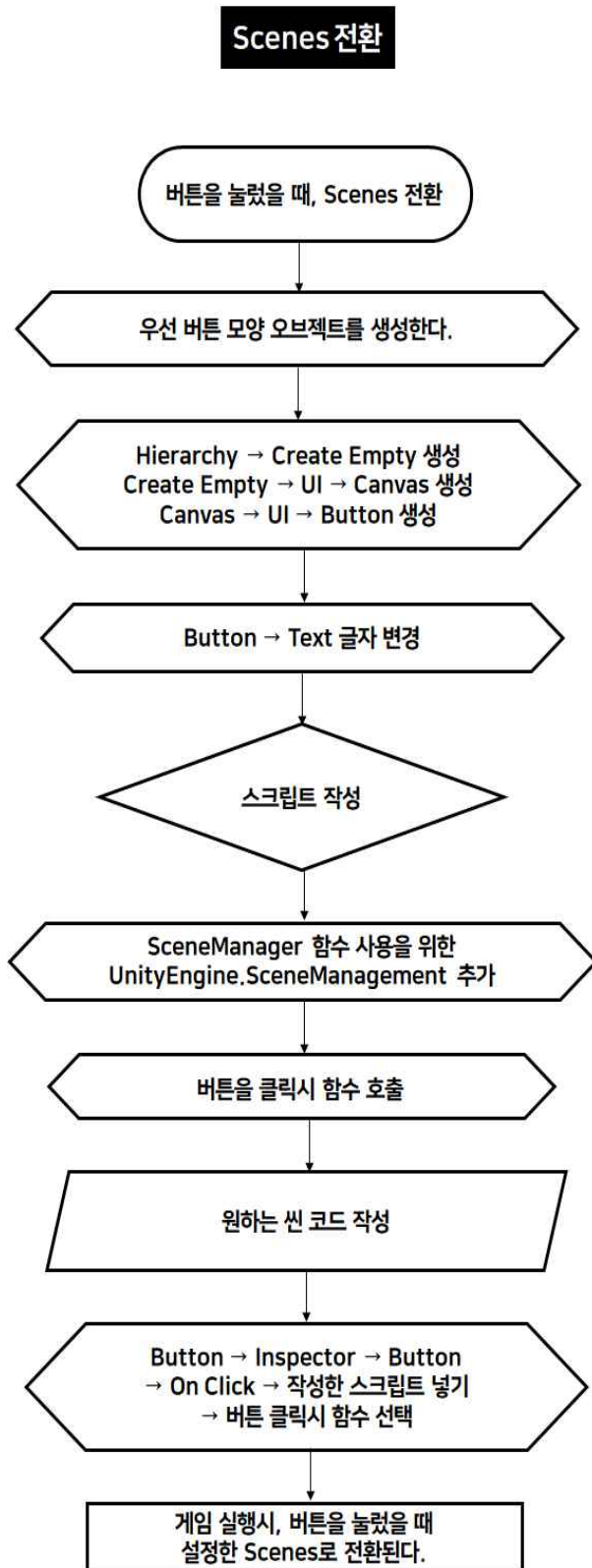
-> 신비의 물약 씬 (PortionScene)



-> 게임 최종 성공 씬 (WinScene)



□ 순서도 (버튼을 눌렀을 때, Scenes 전환)



- 버튼을 눌렀을 때, Scenes 전환
  - StartSystem.cs (시작씬으로 전환)
  - OverSystem.cs(메인 게임 씬으로 전환)



## □ 설명 (버튼을 눌렀을 때, Scenes이 전환된다.)

- ▶ 버튼을 눌렀을 때, 원하는 Scenes으로 전환되도록 구현하려 한다.
- ▶ 먼저 버튼을 만들어야 한다.
- ▶ 버튼 모양 오브젝트를 생성한다.
- ▶ Hierarchy창에서 Create Empty를 생성한다.
- ▶ Create Empty 생성 후, Create Empty에서 UI를 통해 Canvas를 생성한다.
- ▶ 그리고, Canvas에서 UI를 통해 Button을 생성한다.
- ▶ Button에서 Text 글자를 변경한다.
- ▶ 이제 씬 전환을 적용시킬 스크립트를 작성한다.
  - ▶ 씬 전환할 때 필요한 SceneManager 함수 사용을 위한 UnityEngine.SceneManagement 을 추가한다.
  - ▶ 버튼을 클릭할 수 있도록 하는 함수를 호출한다.
  - ▶ 버튼을 클릭할 수 있도록 하는 함수에 원하는 씬 입력한다.
- ▶ 작성한 스크립트를 Hierarchy창에서 생성한 Create Empty에 넣어준다.
- ▶ 그리고, Canvas에서 만든 Button의 Inspector창에서 Button을 찾아 버튼을 클릭시 함수를 선택한다.
- ▶ 게임 실행시, 버튼을 눌렀을 때 원하는 Scenes으로 전환된다.

□ **결과물** (버튼을 눌렀을 때, Scenes이 전환된다.)

▶ “숲으로 들어가기” 버튼을 누르면



▶ 메인 게임 Scenes 씬으로 전환된다.



## □ 소스코드

(StartSystem.cs - 시작씬으로 전환/OverSystem.cs - 메인 게임 씬으로 전환)

```
StartSystem.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class StartSystem : MonoBehaviour
7 {
8     public void OnClickGame()
9     {
10         SceneManager.LoadScene("DemoDay"); //시작 버튼을 누를시 메인 게임씬으로 간다.
11     }
12 }

OverSystem.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class OverSystem : MonoBehaviour
7 {
8     public void OnClickGame()
9     {
10         SceneManager.LoadScene("StartScenes"); //버튼을 누를시 게임 시작 씬으로 간다.
11     }
12 }
```



## □ 설명

(Player의 움직임, 다양한 무기 교체 및 아이템 섭취, 애니메이션과 카메라 설정)

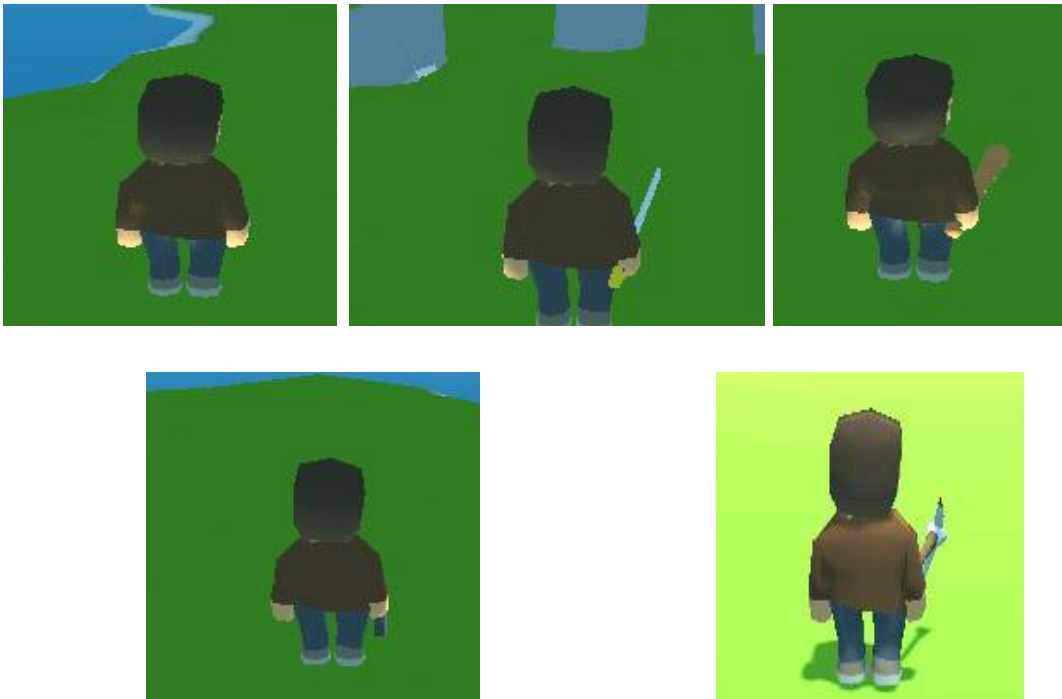
- ▶ Player의 움직임과 전체적인 기능을 첨가한다.
- ▶ 기본적으로 Player 오브젝트 Inspector 창에서 Rigidbody, Capsule Collider, Animator, Audio Source를 넣어준다.
- ▶ 여기서 Rigidbody의 회전 값은 전부 고정시켜준다.(X,Y,Z)
- ▶ 이제 Player의 움직임과 다양한 기능들을 적용시킬 스크립트를 만들어준다.
  - ▶ 먼저 Player가 걷거나 달리는 것을 구현하기 위해 걷는 속도와 달리는 속도 변수를 설정하고 걷는 속도와 달리는 속도 서로 전환될 수 있도록 하는 변수도 설정한다.
  - ▶ 점프를 위해 Player가 땅에 있는지 없는지 상태를 확인해주는 변수도 설정한다.
  - ▶ 마우스를 이용하여 Player의 시선을 처리할 수 있도록 카메라를 조절하는 변수를 설정한다.
  - ▶ Player에서 총알이 나갈 수 있도록 총알 오브젝트 변수와 총알 속도 및 소리 변수를 설정한다.
  - ▶ 무엇인가를 섭취했을 때 소리가 나도록 하는 변수를 설정한다.
  - ▶ 소리를 Player 오브젝트 Inspector 창에서 생성할 수 있도록 코드 변수를 작성한다.
  - ▶ 무기 교체를 위해 키를 불러올 변수를 설정한다. 그리고 키에 넣어줄 무기 오브젝트 변수를 설정한다.
  - ▶ 카메라를 Inspector 창에서 생성할 수 있도록 코드 변수를 작성한다.
  - ▶ Player의 Collider에 물리적인 기능을 하도록 하는 코드 변수를 작성한다.
  - ▶ Player의 애니메이션을 사용하기 위한 애니메이션 변수를 작성한다.
  - ▶ 만약 마우스 오른쪽을 누르면,(ture) ▶만약 누르지 않으면(false)
  - ▶ 총알의 Rigidbody를 불러온다. ▶총알이 나오지 않는다.
  - ▶총알이 Player 위치에서 등장 한다.
  - ▶총알이 Player 앞쪽에서 나온다.
  - ▶키보드를 눌렀을 때 함수를 호출한다.
  - ▶설정한 키의 변수 이름을 설정한다.
  - ▶만약 누른 키보드 숫자가 ( )이면,(ture) ▶만약 키보드 숫자를 누르지 않으면(false)
  - ▶( )로 설정한 오브젝트가 등장한다. ▶오브젝트가 등장하지 않는다.
  - ▶만약 스페이스바를 누르고 땅에 착지해 있다면(ture)
  - ▶점프한다.
  - ▶스페이스바를 누르지 않고 땅에 착지하지 않으면(false)
  - ▶점프하지 않는다.
  - ▶만약 Shift 키를 누르면(ture)
  - ▶달린다.
  - ▶만약 Shift 키를 누르지 않으면(false)
  - ▶달리지 않는다.
  - ▶만약 ( )키를 누르면(ture)

- ▶ 설정한 애니메이션을 실행시킨다.(뛰는 모션, 공격모션, 총 모션)
- ▶ 만약 ( )키를 누르지 않으면(false)
- ▶ 설정한 애니메이션이 실행되지 않는다.
- ▶ 만약 충돌한 물체가 ( )라면(true)
- ▶ 상단바에 설정한 각각의 GUI박스가 증가 및 감소한다.
- ▶ 만약 충돌한 물체가 ( )가 아니면(false)
- ▶ 상단바에 설정한 각각의 GUI박스가 증가 및 감소하지 않는다.
- ▶ Player 땅 착지 여부를 알 수 있는 함수 코드를 작성한다.
- ▶ 걷는 속도가 달리는 속도로 바뀌도록 함수 코드를 작성한다.
- ▶ 달리는 속도가 걷는 속도로 바뀌도록 함수 코드를 작성한다.
- ▶ A,D,W,S를 누르면 Player의 움직임 X,Z축을 이용하여 상,하,좌,우로 이동하도록 정의한다.
- ▶ 마우스를 상,하,좌,우로 움직이면 카메라가 상,하,좌,우로 움직이도록 하는 함수 코드를 작성한다.
- ▶ 작성한 스크립트를 Player에 넣는다.
- ▶ 게임 실행시, Player가 움직이고 무기 교체 및 아이템을 섭취하고 설정한 애니메이션이 실행된다.

## □ 결과물

(Player의 움직임, 다양한 무기 교체 및 아이템 섭취, 애니메이션과 카메라 설정)

### ▶ 무기 교체(~,1,2,3,4)



## ▶아이템 섭취



## ▶움직임 애니메이션



## □ 소스코드

(PlayerControl.cs - 메인 게임 밤씬에 나오는 Player의 소스 코드)

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerControl : MonoBehaviour
6  {
7      //스피드 조절 변수
8      [SerializeField] //walkSpeed를 Inspector창에서 수정 가능하도록 하는 기능
9      private float walkSpeed; //걸는 속도 변수 설정
10
11      [SerializeField] //runSpeed를 Inspector창에서 수정 가능하도록 하는 기능
12      private float runSpeed; //달리는 속도 변수 설정
13
14      private float applySpeed; //걸는 속도와 달리는 속도를 전환할 수 있도록 하는 변수 설정
15
16      [SerializeField] //jumpForce를 Inspector창에서 수정 가능하도록 하는 기능
17      private float jumpForce; //점프 변수 설정
18
19      private bool isGround = true; //땅에 있는지 없는지 상태를 확인해주는 변수 코드
20
21      private CapsuleCollider capsuleCollider; //발 착지 여부
22
23      //민감도
24      [SerializeField] //lookSensitivity를 Inspector창에서 수정 가능하도록 하는 기능
25      private float lookSensitivity; //카메라의 민감도 변수를 설정한다.
26
27      //카메라 한계
28      [SerializeField]
29      private float cameraRotationLimit; //마우스로 카메라를 조절할 때 일정 각도까지만 움직이도록 제한을 준다.
30      private float currentCameraRotationX = 0; //카메라가 정면을 바라볼 수 있도록 설정
31
32      public GameObject bullet; //총알 오브젝트 변수
33      public float bulletSpeed = 10.0f; //총알 속도
34      private AudioSource audioBullet; //총알 소리 변수 설정
35      private GameManager gameManager; //채택을 각기위해 gameManager 변수를 설정한다.
36
37      [SerializeField]
38      private Camera theCamera; //Camera의 GetComponent를 불러온다.
39
40      private Rigidbody myRigid; //Player의 Collider에 물리적인 기능을 하도록 하는 기능 //Inspector창에 Rigidbody추가 후 작성
41
42      private Animator playerAnia; //Player의 애니메이션을 사용하기 위해 애니메이션 변수를 설정한다.
43
44
45      // Use this for initialization
46      void Start()
47      {
48          capsuleCollider = GetComponent<CapsuleCollider>(); //Inspector창에서 CapsuleCollider를 불러온다.
49          myRigid = GetComponent<Rigidbody>(); //Inspector창에서 Rigidbody를 불러온다.
50          audioBullet = GetComponent<AudioSource>(); //총알 소리를 불러온다.
51          applySpeed = walkSpeed; //달리기 전까지 걷는 상태
52          playerAnia = GameObject.Find("Player").GetComponent<Animator>(); //Inspector창에서 Animator를 불러온다.
53          gameManager = GameObject.Find("GameManager").GetComponent<GameManager>(); //GameManager를 불러온다.
54      }
55
56
57
58
59
60      // Update is called once per frame
61      void Update()
62      {
63          isGround(); //땅에 착지하고 있는지 하지 않는지 제어하기 위한 코드 작성 전 함수 정의
64          TryRun(); //뛰는지 걷는지 제어하기 위한 코드 작성 전 함수 정의(움직임 제어)
65          Move(); //움직임을 직접 실시간으로 움직이도록 하는 코드를 작성하기 위한 함수 정의
66          Quaternion.LookRotation(transform.forward); //카메라 움직임을 함수 정의
67          CharacterRotation(); //캐릭터 움직임을 함수 정의
68
69          if (Input.GetMouseButtonDown(1))
70          {
71              GameObject newBullet = Instantiate(bullet, transform.position + transform.forward, transform.rotation) as GameObject; //총알이 Player에서 나오도록하기
72              Rigidbody rbBullet = newBullet.GetComponent<Rigidbody>(); //총알의 Rigidbody를 불러오기
73              rbBullet.velocity = transform.forward * bulletSpeed; //총알이 Player 앞에 나오도록 하기
74          }
75
76
77
78
79      }
80
81      // 지면 체크.
82      private void isGround()
83      {
84          isGround = Physics.Raycast(transform.position, Vector3.down, capsuleCollider.bounds.extents.y * 0.1f); //Player의 Y축 방향으로 0.1 거리 만큼 레이저를 쏘아 Player가 땅에 닿는지 닿지 않는지 여부 인식(땅에 착지한 순간)
85      }
86
87      // 점프 시도
88      private void TryJump()
89      {
90          if (Input.GetKeyDown(KeyCode.Space) && isGround) //Space키가 눌려지고 Player가 땅에 착지하고 있으면
91          {
92              Jump();
93          }
94      }
95
96
97
98
99
100

```



```

91     }
92 }
93
94 // 점프
95 private void Jump()
96 {
97     myRigid.velocity = transform.up * jumpForce; //Player의 움직임에서 Y축으로 설정한 점프 힘만큼 증가하도록 한다.
98 }
99
100 //달리기 시도
101 private void TryRun()
102 {
103     if (Input.GetKey(KeyCode.LeftShift)) //왼쪽 Shift 키를 누르면
104     {
105         Running(); //달리기가 실행됨
106     }
107     if (Input.GetKey(KeyCode.LeftShift)) //왼쪽 Shift키가 올라올라왔을 때
108     {
109         RunningCancel(); //달리지 않는다.
110     }
111 }
112
113 // 달리기 실행
114 private void Running()
115 {
116     applySpeed = runSpeed; //같은 속도가 달리는 속도로 바뀐다.
117 }
118 // 달리기 취소
119 private void RunningCancel()
120 {
121     applySpeed = walkSpeed; //달리는 속도가 걷는 속도로 바뀐다.
122 }
123
124 private void Move()
125 {
126     float _moveDirX = Input.GetAxisRaw("Horizontal"); //A,D를 눌렀을 때
127     float _moveDirZ = Input.GetAxisRaw("Vertical"); //W,S를 눌렀을 때
128
129     Vector3 _moveHorizontal = transform.right * _moveDirX; //Player의 transform X축을 이용하여 좌, 우 이동하도록 정의
130     Vector3 _moveVertical = transform.forward * _moveDirZ; //Player의 transform Z축을 이용하여 앞, 뒤 이동하도록 정의
131     Vector3 _velocity = (_moveHorizontal + _moveVertical).normalized * applySpeed; //속도는 방향이 움직이면 속도값을 곱하여 얼마나 빨리 이동할 것인지 설정한다.
132     myRigid.MovePosition(transform.position + _velocity * Time.deltaTime); //Rigidbody에 움직일 함수를 정의하여 Player의 위치에 속도를 더해주고, 1초동안 속도만큼 움직이게 하겠다.
133 }
134
135
136
137 if (_moveDirZ > 0.05f || _moveDirZ < -0.05f) //만약 A,D를 누르고 있으면
138 {
139     playerAnia.SetBool("Run", true); //뛰어라
140 }
141 else
142 {
143     playerAnia.SetBool("Run", false); //멈춰라
144 }
145
146
147 if (Input.GetMouseButtonDown(0)) //왼쪽 마우스 버튼을 누르면
148 {
149     playerAnia.SetBool("Hit", true); //때리는 애니메이션을 실행
150 }
151 else
152 {
153     playerAnia.SetBool("Hit", false); //때리는 애니메이션을 멈춰라
154 }
155
156
157 if (Input.GetMouseButtonDown(1)) //오른쪽 마우스 버튼을 누르면
158 {
159     playerAnia.SetBool("Dig", true); //굴보는 애니메이션을 실행
160     audioBullet.Play();
161 }
162 else
163 {
164     playerAnia.SetBool("Dig", false); //굴보는 애니메이션을 멈춰라
165 }
166
167
168
169
170 private void CharacterRotation()
171 {
172     // 좌우 캐릭터 회전
173     float _yRotation = Input.GetAxisRaw("Mouse X"); //마우스를 좌 우로 움직였을 때, 카메라가 좌 우(Y축)으로 움직인다.
174     Vector3 _characterRotation = new Vector3(0f, _yRotation, 0f) * lookSensitivity; //카메라가 천천히 움직이도록 설정한다.
175     myRigid.MoveRotation(myRigid.rotation * Quaternion.Euler(_characterRotation)); //Player의 회전값에 천천히 움직이도록 설정한 카메라 값을 곱하여 회전하도록 한다.
176 }
177
178
179 private void CameraRotation()
180 {
181     //상 하 카메라 회전
182     float _yRotation = Input.GetAxisRaw("Mouse Y"); //마우스를 앞 뒤로 움직였을 때, 카메라가 상 하(X축)으로 움직인다.
183     float _cameraRotationX = _yRotation * lookSensitivity; //카메라가 천천히 움직이도록 설정한다.
184     currentCameraRotationX -= _cameraRotationX; //현재 카메라 움직임은 마우스를 앞 뒤로 움직이는 값에 따라 움직인다.
185     currentCameraRotationX = Mathf.Clamp(currentCameraRotationX, -cameraRotationLimit, cameraRotationLimit); //cameraRotationLimit의 최대, 최소값을 설정한다.(설정된 값도만큼 이상 움직이지 못하도록 제한을 둔다.)
186
187     theCamera.transform.localEulerAngles = new Vector3(currentCameraRotationX, 0f, 0f); //현재 카메라에 적용시키기 위한 설정을 한다. 마우스를 앞 뒤로 움직였을 때 카메라가 상 하(X축)로만 움직이도록 설정한다.
188 }
189
190 void OnCollisionEnter(Collision collision)
191 {
192     if (collision.gameObject.name == "zombie(Clone)") //충돌한 대상이 좀비라면
193     {
194         gameManager.HP -= 1; //체력이 줄어든다.
195     }
196 }
197

```

## (PlayerControll2.cs - 메인 게임 낫씬에 나오는 Player의 소스 코드)

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PlayerControll2 : MonoBehaviour
6  {
7      //스피드 조정 변수
8      [SerializeField] //walkSpeed를 Inspector창에서 수정 가능하도록 하는 기능
9      private float walkSpeed; //걸는 속도 변수 설정
10
11      [SerializeField] //runSpeed를 Inspector창에서 수정 가능하도록 하는 기능
12      private float runSpeed; //달리는 속도 변수 설정
13
14      private float applySpeed; //걸는 속도와 달리는 속도를 대입할 수 있도록 하는 변수 설정
15
16      [SerializeField] //jumpForce를 Inspector창에서 수정 가능하도록 하는 기능
17      private float jumpForce; //점프 변수 설정
18
19      //상태 변수
20      private bool isGround = true; //땅에 있는지 없는지 상태를 확인해주는 변수 코드
21
22
23      private CapsuleCollider capsuleCollider; //땅 착지 여부
24
25      //민감도
26      [SerializeField] //lookSensitivity를 Inspector창에서 수정 가능하도록 하는 기능
27      private float lookSensitivity; //카메라의 민감도 변수를 설정한다.
28
29      //카메라 한계
30      [SerializeField]
31      private float cameraRotationLimit; //마우스로 카메라를 조절할 때 일정한 각도까지만 움직이도록 제한을 두는 변수 설정
32      private float currentCameraRotationX = 0; //카메라가 정면을 바라볼 수 있도록 설정
33
34      public GameObject bullet; //총알 오브젝트 변수
35      public float bulletSpeed = 10.0f; //총알 속도
36      private AudioSource audioBullet; //총알 소리 변수 설정
37      private GameManager2 gameManager; //제어를 갖기 위해 gameManager 변수를 설정한다.
38
39      public AudioClip flowerSource; //무엇인가를 설치했을 때 소리가 나도록 하는 변수 설정
40      AudioSource flowerSource01; //소리를 Inspector창에 생성할 수 있도록 코드 변수 설정
41
42
43      public GameObject kal; //칼 오브젝트 설정
44      public GameObject mang; //방망이 오브젝트 설정
45      public GameObject gun; //총 오브젝트 설정
46      public GameObject goi; //곡괭이 오브젝트 설정
47
48
49      private bool hand; //맨손 카를 불러올 변수 설정
50      private bool pon1; //카를 불러올 오브젝트 변수 설정(칼)
51      private bool pon2; //카를 불러올 오브젝트 변수 설정(방망이)
52      private bool pon3; //카를 불러올 오브젝트 변수 설정(총)
53      private bool pon4; //카를 불러올 오브젝트 변수 설정(곡괭이)
54
55      [SerializeField]
56      private Camera theCamera; //Camera의 GetComponent를 불러온다.
57
58      private Rigidbody myRigid; //Player의 Collider에 물리적인 기능을 하도록 하는 기능 //Inspector창에 Rigidbody추가 후 작성
59
60      private Animator playerAnia; //Player의 애니메이션을 사용하기 위한 애니메이션 변수를 설정한다.
61      void GetInput() //키보드를 눌렀을 때
62      {
63          hand = Input.GetButtonDown("Hand");
64          pon1 = Input.GetButtonDown("Pon1");
65          pon2 = Input.GetButtonDown("Pon2");
66          pon3 = Input.GetButtonDown("Pon3");
67          pon4 = Input.GetButtonDown("Pon4");
68
69          if (hand) //키보드 숫자 1 누르면
70          {
71              kal.SetActive(false);
72              mang.SetActive(false);
73              gun.SetActive(false);
74              goi.SetActive(false);
75          }
76
77          if (pon1) //키보드 숫자 1 누르면
78          {
79              kal.SetActive(true); //칼 오브젝트 등장
80              mang.SetActive(false);
81              gun.SetActive(false);
82              goi.SetActive(false);
83          }
84
85          if (pon2) //키보드 숫자 2 누르면
86          {
87              mang.SetActive(true); //방망이 오브젝트 등장
88              kal.SetActive(false);
89              gun.SetActive(false);
90              goi.SetActive(false);
91          }
92
93          if (pon3) //키보드 숫자 3 누르면
94          {
95              mang.SetActive(false);
96              kal.SetActive(false);
97              gun.SetActive(true); //총 오브젝트 등장
98              goi.SetActive(false);
99          }
100
101          if (pon4) //키보드 숫자 4 누르면
102          {
103              mang.SetActive(false);
104              kal.SetActive(false);
105              gun.SetActive(false);
106              goi.SetActive(true); //곡괭이 오브젝트 등장
107          }
108      }
109
110      // Use this for initialization
111      void Start()
112      {
113          capsuleCollider = GetComponent<CapsuleCollider>(); //Inspector창에서 CapsuleCollider를 불러온다.
114          myRigid = GetComponent<Rigidbody>(); //Inspector창에서 Rigidbody를 불러온다.
115          audioBullet = GetComponent<AudioSource>(); //총알 소리를 불러온다.
116          applySpeed = walkSpeed; //달리기 전까지 걸는 상태
117          playerAnia = GameObject.Find("Player").GetComponent<Animator>(); //Inspector창에서 Animator를 불러온다.
118          gameManager = GameObject.Find("GameManager2").GetComponent<GameManager2>(); //GameManager2를 불러온다.
119          flowerSource01 = GetComponent<AudioSource>(); //소리를 불러온다.
120          kal.SetActive(false); //시작할 때 무기를 안보이게 설정
121          mang.SetActive(false); //시작할 때 무기를 안보이게 설정
122          gun.SetActive(false); //시작할 때 무기를 안보이게 설정
123          goi.SetActive(false); //시작할 때 무기를 안보이게 설정
124      }
125
126      // Update is called once per frame
127      void Update()
128      {
129          isGround(); //땅에 착지하고 있는지 하지 않는지 제어하기 위한 코드 작성 전 함수 정의
130          TryJump(); //점프를 실행 및 제어하기 위한 코드 작성 전 함수 정의
131          TryRun(); //미는지 걸는지 제어하기 위한 코드 작성 전 함수 정의(움직임 제어)
132          Move(); //움직임을 하면 실시간으로 움직이도록 하는 코드를 작성하기 위한 함수 정의
133          CameraRotation(); //카메라 움직임 함수 정의
134          CharacterRotation(); //캐릭터 움직임 함수 정의
135          GetInput(); //키를 눌렀을 때 함수 정의
136      }

```

```

137 if (Input.GetMouseButtonDown(1)) //왼쪽 마우스 오른쪽을 누르면
138 {
139     GameObject newBullet = Instantiate(bullet, transform.position + transform.forward, transform.rotation) as GameObject; //총알이 Player에서 나오도록 한다.
140     Rigidbody rbBullet = newBullet.GetComponent<Rigidbody>(); //총알의 Rigidbody를 얻어오기
141     rbBullet.velocity = transform.forward * bulletSpeed; //총알이 Player 밑에 나오도록 하기
142 }
143
144 }
145
146 private void IsGround() //지면을 체크하는 함수 코드
147 {
148     isGround = Physics.Raycast(transform.position, Vector3.down, capsuleCollider.bounds.extents.y * 0.1f); //Player의 Y축 방향으로 0.1 거리 만큼 레이저를 쏘아 Player가 땅에 닿는지 알지 않는지 여부 인식(땅에 착지한 순간)
149 }
150
151 // 점프 시도
152 private void TryJump()
153 {
154     if (Input.GetKeyDown(KeyCode.Space) && isGround) //Space키가 눌리고 Player가 땅에 착지하고 있으면
155     {
156         jump();
157     }
158 }
159
160 // 점프
161 private void Jump()
162 {
163     myRigid.velocity = transform.up * jumpForce; //Player의 움직임에서 Y축으로 설정한 점프 힘만큼 증가하도록 한다.
164 }
165
166 //달리기 시도
167 private void TryRun()
168 {
169     if (Input.GetKey(KeyCode.LeftShift)) //왼쪽 Shift 키를 누르면
170     {
171         Running(); //달리기가 실행됨
172     }
173     if (Input.GetKeyUp(KeyCode.LeftShift)) //왼쪽 Shift키가 올라올때까지
174     {
175         RunningCancel(); //달리지 않는다.
176     }
177 }
178
179 // 달리기 실행
180 private void Running()
181 {
182     applySpeed = runSpeed; //같은 속도가 달리는 속도로 바뀐다.
183 }
184
185 // 달리기 취소
186 private void RunningCancel()
187 {
188     applySpeed = walkSpeed; //달리는 속도가 걷는 속도로 바뀐다.
189 }
190
191 private void Move()
192 {
193     float _moveDirX = Input.GetAxisRaw("Horizontal"); //A,D를 눌렀을 때
194     float _moveDirZ = Input.GetAxisRaw("Vertical"); //W,S를 눌렀을 때
195
196     Vector3 _moveHorizontal = transform.right * _moveDirX; //Player의 transform X축을 이용하여 좌, 우 이동하도록 정의
197     Vector3 _moveVertical = transform.forward * _moveDirZ; //Player의 transform Z축을 이용하여 앞, 뒤 이동하도록 정의
198
199     Vector3 _velocity = (_moveHorizontal + _moveVertical).normalized * applySpeed; //속도는 방향이 움직이면 속도값을 곱하여 얼마나 빨리 이동할 것인지 설정한다.
200
201     myRigid.MovePosition(transform.position + _velocity * Time.deltaTime); //Rigidbody에 움직일 함수를 정의하여 Player의 위치에 속도를 더해주고, 1초동안 속도만큼 움직이게 하겠다.
202
203     if (_moveDirZ > 0.05f || _moveDirZ < -0.05f) //안막 A,D를 누르고 있으면
204     {
205         playerAnia.SetBool("Run", true); //뛰어라
206     }
207     else
208     {
209         playerAnia.SetBool("Run", false); //멈춰라
210     }
211
212     if (Input.GetMouseButtonDown(0)) //왼쪽 마우스 버튼을 누르면
213     {
214         playerAnia.SetBool("Hit", true); //패리는 애니메이션을 실행
215         audioBullet.Play();
216     }
217
218     if (Input.GetMouseButton(0)) //왼쪽 마우스 버튼이 올라오면
219     {
220         playerAnia.SetBool("Hit", false); //패리는 애니메이션을 멈춰라
221     }
222 }
223
224
225 if (Input.GetMouseButtonDown(1)) //오른쪽 마우스 버튼을 누르면
226 {
227     playerAnia.SetBool("Dig", true); //총포는 애니메이션을 실행
228 }
229
230 if (Input.GetMouseButton(1)) //오른쪽 마우스 버튼이 올라오면
231 {
232     playerAnia.SetBool("Dig", false); //총포는 애니메이션을 멈춰라
233 }
234
235 }
236
237 private void CharacterRotation()
238 {
239     // 좌우 캐릭터 회전
240     float _yRotation = Input.GetAxisRaw("Mouse X"); //마우스를 좌,우로 움직였을 때, 카메라가 좌,우(Y축)으로 움직인다.
241     Vector3 _characterRotation = new Vector3(0f, _yRotation, 0f) * lookSensitivity; //카메라가 완전히 움직이도록 설정한다.
242     myRigid.MoveRotation(myRigid.rotation * Quaternion.Euler(_characterRotation)); //Player의 회전값에 완전히 움직이도록 설정한 카메라 값을 곱하여 회전하도록 한다.
243 }
244
245 private void CameraRotation()
246 {
247     // 상하 카메라 회전
248     float _yRotation = Input.GetAxisRaw("Mouse Y"); //마우스를 앞,뒤로 움직였을 때, 카메라가 상,하(X축)으로 움직인다.
249     float _cameraRotation = _yRotation * lookSensitivity; //카메라가 완전히 움직이도록 설정한다.
250     currentCameraRotation -= _cameraRotation; //현재 카메라 움직임은 마우스를 앞,뒤로 움직이는 값에 따라 움직인다.
251     currentCameraRotation = Mathf.Clamp(currentCameraRotation, -cameraRotationLimit, cameraRotationLimit); //cameraRotationLimit의 최대, 최소값을 결정한다. (설정된 값도만큼 이상 움직이지 못하도록 제한을 둔다.)
252     theCamera.transform.localEulerAngles = new Vector3(currentCameraRotation, 0f, 0f); //현재 카메라에 적용시키기 위한 설정을 한다. 마우스를 앞,뒤로 움직였을 때 카메라가 상,하(X축)로만 움직이도록 설정한다.
253 }
254
255 void OnCollisionEnter(Collision collision)
256 {
257     if (collision.gameObject.tag == "Fox") //만약 충돌한 오브젝트가 여우면
258     {
259         gameManager.FP -- 1; //체력이 감소한다.
260     }
261
262     if (collision.gameObject.tag == "blue") //만약 충돌한 오브젝트가 꽃이면
263     {
264         gameManager.Flower ++ 1; //꽃 개수가 증가한다.
265         FlowerSource01.PlayOneShot(FlowerSource); //설정된 소리가 재생된다.
266     }
267 }
268

```

```

269
270     if (collision.gameObject.tag == "food") //만약 충돌한 오브젝트가 과일이면
271     {
272         gameManager.HP += 1; //체력이 증가한다.
273         gameManager.food += 1; //음식이 증가한다.
274         FlowerSource01.PlayOneShot(FlowerSource); //설정한 소리가 재생된다.
275     }
276
277     if (collision.gameObject.tag == "Potion") //만약 충돌한 오브젝트가 물약이면
278     {
279         gameManager.rock += 1; //물약이 증가한다.
280         FlowerSource01.PlayOneShot(FlowerSource); //설정한 소리가 재생된다.
281     }
282 }
283
284
285
286

```

## (PlayerControll3.cs - 신비의 물약썩에 나오는 Player의 소스 코드)

```

PlayerControll3.cs
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerControll3 : MonoBehaviour
{
    //스피드 조정 변수
    [SerializeField] //walkSpeed를 Inspector창에서 수정 가능하도록 하는 기능
    private float walkSpeed; //걸는 속도 변수 설정
    [SerializeField] //runSpeed를 Inspector창에서 수정 가능하도록 하는 기능
    private float runSpeed; //달리는 속도 변수 설정
    private float applySpeed; //걸는 속도와 달리는 속도를 대입할 수 있도록 하는 변수 설정
    [SerializeField] //jumpForce를 Inspector창에서 수정 가능하도록 하는 기능
    private float jumpForce; //점프 변수 설정
    private bool isGround = true; //땅에 있는지 없는지 상태를 확인해주는 변수 코드
}

private CapsuleCollider capsuleCollider; //땅 착지 여부
//민감도
[SerializeField] //lookSensitivity를 Inspector창에서 수정 가능하도록 하는 기능
private float lookSensitivity; //카메라의 민감도 변수를 설정한다.
//카메라 한계
[SerializeField]
private float cameraRotationLimit; //마우스로 카메라를 조절할 때 일정 각도까지만 움직이도록 제한을 준다.
private float currentCameraRotationX = 0; //카메라가 평면을 바라볼 수 있도록 설정
public GameObject bullet; //총알 오브젝트 변수
public float bulletSpeed = 10.0f; //총알 속도
private AudioSource audioBullet; //총알 소리 변수 설정
private GameManager3 gameManager; //체력을 적기 위해 gameManager 변수를 설정한다.

public AudioClip PotionSource;
private AudioSource PotionSource01;

public GameObject Kal;
public GameObject Mang;
public GameObject Gun;

private bool hand;
private bool pon1;
private bool pon2;
private bool pon3;

//필요한 컴포넌트
[SerializeField]
private Camera theCamera; //Camera의 GetComponent를 불러온다.
private Rigidbody myRigid; //Player의 Collider에 물리적인 기능을 하도록 하는 기능 //Inspector창에 Rigidbody추가 후 작성
private Animator playerAnia; //Player의 애니메이션을 사용하기 위해 애니메이션 변수를 설정한다.

private void GetInput()
{
    hand = Input.GetButtonDown("Hand");
    pon1 = Input.GetButtonDown("Pon1");
    pon2 = Input.GetButtonDown("Pon2");
    pon3 = Input.GetButtonDown("Pon3");

    if (hand) //키보드 숫자 1 누르면
    {
        Kal.SetActive(false);
        Mang.SetActive(false);
        Gun.SetActive(false);
    }

    if (pon1) //키보드 숫자 1 누르면
    {
        Kal.SetActive(true);
        Mang.SetActive(false);
        Gun.SetActive(false);
    }

    if (pon2) //키보드 숫자 2 누르면
    {
        Mang.SetActive(true);
        Kal.SetActive(false);
        Gun.SetActive(false);
    }
}

```



```

98     if (don3) //키보드 숫자 2 누르면
99     {
100         Mang.SetActive(false);
101         Kai.SetActive(false);
102         Gun.SetActive(true);
103     }
104 }
105
106 // Use this for initialization
107 void Start()
108 {
109     capsuleCollider = GetComponent<CapsuleCollider>(); //Inspector창에서 CapsuleCollider를 불러온다.
110     rigidbody = GetComponent<Rigidbody>(); //Inspector창에서 Rigidbody를 불러온다.
111     audioBullet = GetComponent<AudioSource>(); //총알 소리를 불러온다.
112     applySpeed = walkSpeed; //달리기 전까지 걷는 상태
113     playerAnia = GameObject.Find("Player").GetComponent<Animator>(); //Inspector창에서 Animator를 불러온다.
114     gameManager = GameObject.Find("GameManager").GetComponent<GameManager3>(); //GameManager를 불러온다.
115     PotionSource01 = GetComponent<AudioSource>(); //소리를 불러온다.
116 }
117
118 // Update is called once per frame
119 void Update()
120 {
121     isGround(); //땅에 착지하고 있는지 하지 않는지 제어하기 위한 코드 작성 전 함수 정의
122     TryJump(); //점프를 실행 및 제어하기 위한 코드 작성 전 함수 정의
123     TryRun(); //뛰는지 않는지 제어하기 위한 코드 작성 전 함수 정의(움직임 제어)
124     Move(); //움직임을 하면 실시간으로 움직이도록 하는 코드를 작성하기 위한 함수 정의
125     CameraRotation(); //카메라 움직임 함수 정의
126     CharacterRotation(); //캐릭터 움직임 함수 정의
127
128     if (Input.GetMouseButtonDown(1)) //왼쪽 마우스 오른쪽 버튼을 누르면
129     {
130         GameObject newBullet = Instantiate(bullet, transform.position + transform.forward, transform.rotation) as GameObject; //총알이 Player에서 나오도록 한다.
131         Rigidbody rbBullet = newBullet.GetComponent<Rigidbody>(); //총알의 Rigidbody 불러오기
132         rbBullet.velocity = transform.forward * bulletSpeed; //총알이 Player 앞에 나오도록 하기
133     }
134 }
135
136 // 지면 체크
137 private void isGround()
138 {
139     isGround = Physics.Raycast(transform.position, Vector3.down, capsuleCollider.bounds.extents.y * 0.1f); //Player의 Y축 방향으로 0.1 거리 만큼 레이저를 쏘아 Player가 땅에 닿는지 닿지 않는지 여부 인식(땅에 착지한 순간)
140 }
141
142 // 점프 시도
143 private void TryJump()
144 {
145     if (Input.GetKeyDown(KeyCode.Space) && isGround) //Space키가 눌리고 Player가 땅에 착지하고 있으면
146     {
147         Jump();
148     }
149 }
150
151 // 점프
152 private void Jump()
153 {
154     rigidbody.velocity = transform.up * jumpForce; //Player의 움직임에서 Y축으로 설정한 점프 힘을 증가하도록 한다.
155 }
156
157 //달리기 시도
158 private void TryRun()
159 {
160     if (Input.GetKeyDown(KeyCode.LeftShift)) //왼쪽 Shift 키를 누르면
161     {
162         Running(); //달리기가 실행됨
163     }
164     if (Input.GetKeyUp(KeyCode.LeftShift)) //왼쪽 Shift키가 올라올때까지
165     {
166         RunningCancel(); //달리지 않는다.
167     }
168 }
169
170 // 달리기 실행
171 private void Running()
172 {
173     applySpeed = runSpeed; //걷는 속도가 달리는 속도로 바뀐다.
174 }
175
176 // 달리기 취소
177 private void RunningCancel()
178 {
179     applySpeed = walkSpeed; //달리는 속도가 걷는 속도로 바뀐다.
180 }
181
182 private void Move()
183 {
184     float _moveDirX = Input.GetAxisRaw("Horizontal"); //A,D를 눌렀을 때
185     float _moveDirZ = Input.GetAxisRaw("Vertical"); //W,S를 눌렀을 때
186
187     Vector3 _moveHorizontal = transform.right * _moveDirX; //Player의 transform X축을 이용하여 좌, 우 이동하도록 정의
188     Vector3 _moveVertical = transform.forward * _moveDirZ; //Player의 transform Z축을 이용하여 앞, 뒤 이동하도록 정의
189
190     Vector3 _velocity = (_moveHorizontal + _moveVertical).normalized * applySpeed; //속도는 방향이 움직이면 속도값을 곱하여 얼마나 빨리 이동할 것인지 설정한다.
191
192     rigidbody.MovePosition(transform.position + _velocity * Time.deltaTime); //Rigidbody에 움직임 함수를 정의하여 Player의 위치에 속도를 더해주고, 1초동안 속도만큼 움직이게 하겠다.
193
194     if (_moveDirZ > 0.05f || _moveDirZ < -0.05f) //안쪽 A,D를 누르고 있으면
195     {
196         playerAnia.SetBool("Run", true); //뛰어라
197     }
198     else
199     {
200         playerAnia.SetBool("Run", false); //멈춰라
201     }
202
203     if (Input.GetMouseButtonDown(0)) //왼쪽 마우스 버튼을 누르면
204     {
205         playerAnia.SetBool("Hit", true); //패러는 애니메이션을 실행
206     }
207     if (Input.GetMouseButtonUp(0)) //왼쪽 마우스 버튼이 올라오면
208     {
209         playerAnia.SetBool("Hit", false); //패러는 애니메이션을 멈춰라
210     }
211
212     if (Input.GetMouseButtonDown(1)) //오른쪽 마우스 버튼을 누르면
213     {
214         playerAnia.SetBool("Dig", true); //총알은 애니메이션을 실행
215         audioBullet.Play();
216     }
217     if (Input.GetMouseButtonUp(1)) //오른쪽 마우스 버튼이 올라오면
218     {
219         playerAnia.SetBool("Dig", false); //총알은 애니메이션을 멈춰라
220     }
221 }

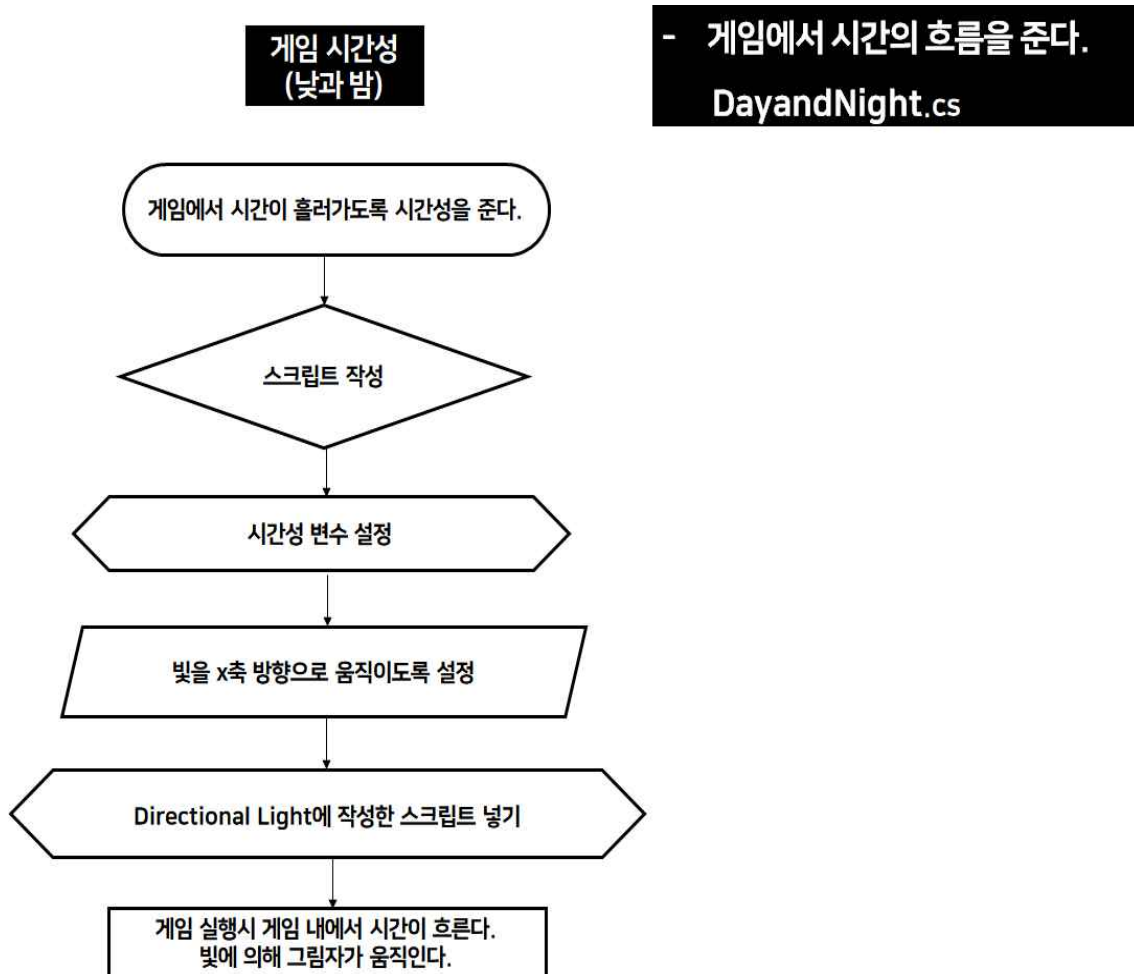
```

```

223 private void CharacterRotation()
224 {
225     // 좌우 카메라 회전
226     float _yRotation = Input.GetAxisRaw("Mouse X"); //마우스를 좌 우로 움직였을 때, 카메라가 좌 우(Y축)으로 움직인다.
227     Vector3 _characterRotationY = new Vector3(0f, _yRotation, 0f) * lookSensitivity; //카메라가 천천히 움직이도록 설정한다.
228     myRigid.MoveRotation(myRigid.rotation * Quaternion.Euler(_characterRotationY)); //Player의 회전값에 천천히 움직이도록 설정한 카메라 값을 곱하여 회전하도록 한다.
229 }
230
231 private void CameraRotation()
232 {
233     //상 하 카메라 회전
234     float _yRotation = Input.GetAxisRaw("Mouse Y"); //마우스를 앞 뒤로 움직였을 때, 카메라가 상 하(X축)으로 움직인다.
235     float _cameraRotationX = _yRotation * lookSensitivity; //카메라가 천천히 움직이도록 설정한다.
236     currentCameraRotationX += _cameraRotationX; //현재 카메라 움직임은 마우스를 앞 뒤로 움직이는 값에 따라 움직인다.
237     currentCameraRotationX = Mathf.Clamp(currentCameraRotationX, -cameraRotationLimit, cameraRotationLimit); //cameraRotationLimit의 최대, 최소값을 결정한다.(설정한 각도만큼 이상 움직이지 못하도록 제한을 둔다.)
238
239     theCamera.transform.localEulerAngles = new Vector3(currentCameraRotationX, 0f, 0f); //현재 카메라에 적용시키기 위한 설정을 한다. 마우스를 앞 뒤로 움직였을 때 카메라가 상 하(X축)로만 움직이도록 설정한다.
240 }
241
242 void OnCollisionEnter(Collision collision)
243 {
244     if (collision.gameObject.name == "potion_yellow") //충돌한 물체가 물약이라면
245     {
246         gameManager._MysteriousPotion += 1; //신비의 물약이 증가한다.
247         PotionSource01.PlayOneShot(PotionSource); //소리가 재생된다.
248     }
249 }
250
251 }
252

```

## □ 순서도 (게임 시간성)



## □ 설명 (게임 시간성)

- ▶ 게임에서 시간이 흘러가도록 시간성을 준다.
- ▶ Directional Light에 넣을 스크립트 작성한다.
  - ▶ 시간성 변수를 설정한다.
  - ▶ 빛을 x축 방향으로 움직이도록 설정한다.
- ▶ Directional Light에 작성한 스크립트를 넣어준다.
- ▶ 게임 실행시 게임 내에서 시간이 흐른다.  
(빛에 의해 그림자가 움직이는 것으로 알 수 있다.)

□ 결과물 (게임 시간성)



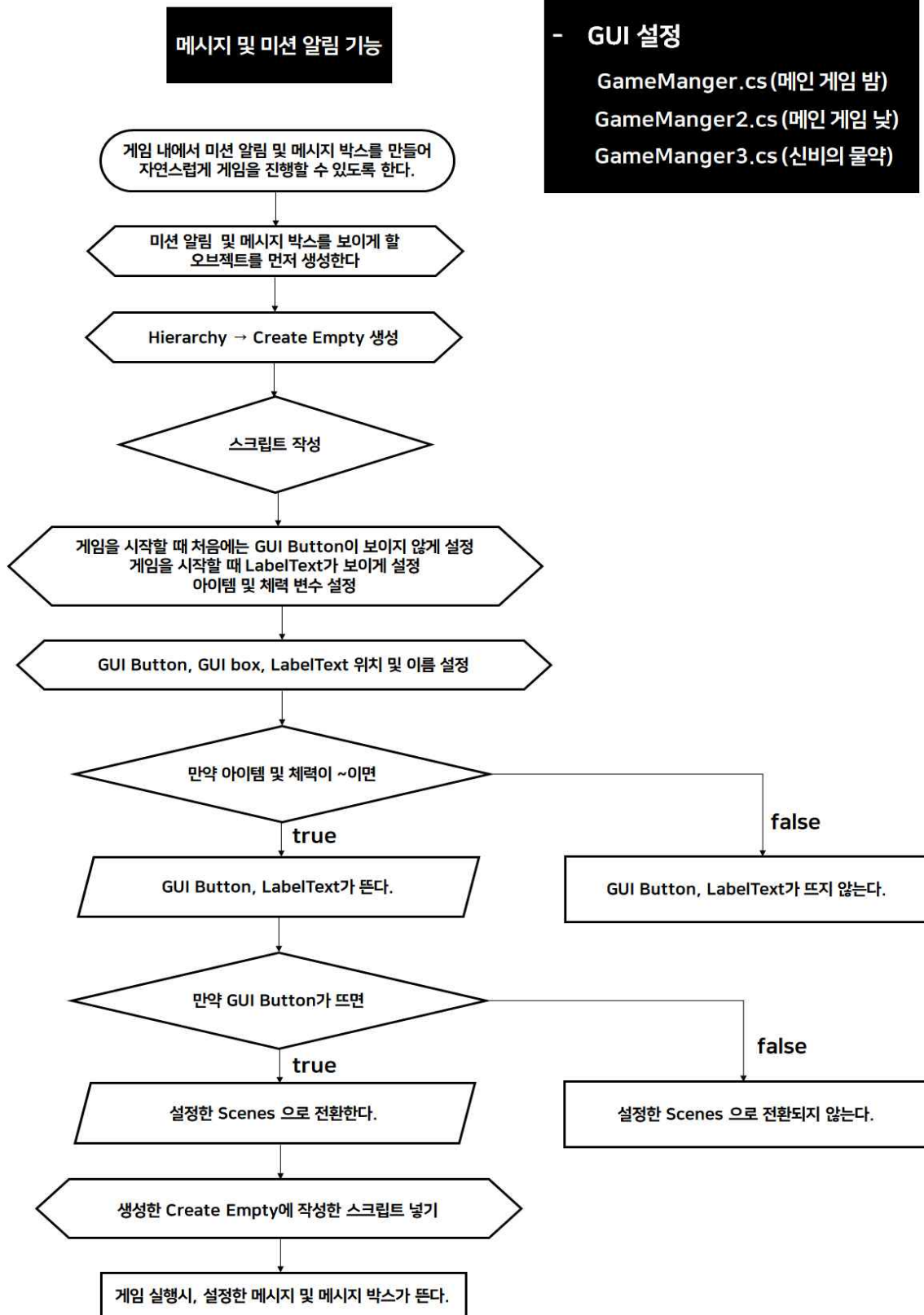


## □ 소스코드

(DayandNight.cs - 게임 내 시간성 소스 코드)

```
DayandNight.cs ×
기타 파일 DayandNight
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class DayandNight : MonoBehaviour
6 {
7     [SerializeField] private float isTime; //게임 시간과 현실 시간이 달라야 하기 때문에 변수 지정
8
9
10    // Start is called before the first frame update
11
12    // Update is called once per frame
13    void Update()
14    {
15        transform.Rotate(Vector3.right, 0.1f * isTime * Time.deltaTime); //빛을 x축 방향으로 움직이도록 설정
16    }
17 }
18
```

## □ 순서도 (GUI 설정)



## □ 설명 (GUI 설정)

- ▶ 게임 내에서 미션 알림 및 메시지 박스를 만들어 자연스럽게 게임을 진행할 수 있도록 한다.
- ▶ 먼저 미션 알림 및 메시지 박스를 보이게 할 오브젝트를 먼저 생성한다.
- ▶ 게임 내 Hierarchy창에서 Create Empty를 생성한다.
- ▶ 생성한 Create Empty에 넣어줄 스크립트를 작성한다.
  - ▶ 게임을 시작할 때 처음에는 GUI Button이 보이지 않게 설정한다.
  - ▶ 게임을 시작할 때 LabelText가 보이게 설정한다.
  - ▶ 아이템 및 체력 변수를 설정한다.
  - ▶ GUI Button, GUI Box, LabelText 위치 및 이름을 설정한다.
  - ▶ 만약 아이템 및 체력이 ( )이면(true)
  - ▶ GUI Button과 LabelText가 뜬다.
  - ▶ 만약 아이템 및 체력이 ( )이 아니면 (false)
  - ▶ GUI Button과 LabelText가 뜨지 않는다.
  - ▶ 만약 GUI Button이 뜨면 (true)
  - ▶ 설정한 Scenes로 전환된다.
  - ▶ 만약 GUI Button이 뜨지 않으면 (false)
  - ▶ 설정한 Scenes로 전환되지 않는다.
- ▶ 생성한 Create Empty에 작성한 스크립트를 넣어준다.
- ▶ 게임 실행시, 설정한 메시지 및 메시지 박스가 뜬다.

## □ 결과물

(게임 내에서 미션 알림 및 메시지 박스를 만들어 자연스럽게 게임을 진행할 수 있도록 한다.)







## □ 소스코드

### GameManger.cs - 메인 게임 밤

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class GameManger : MonoBehaviour
7  {
8      public string labelText = "낮이 되기 전까지 준비를 10마리 죽이자!"; //labelText가 보이도록 설정
9      private int playerHP = 10; //Player 체력 설정
10     public bool dieWindow = false; //처음에는 보이지 않게 설정
11     public bool winWindow = false; //처음에는 보이지 않게 설정
12
13     public int HP //체력 함수
14     {
15         get { return playerHP; } //Player 체력값 다시 반복
16         set
17         {
18             playerHP = value; //player체력 값 저장
19             if (playerHP <= 0) //player 체력이 0이면
20             {
21                 dieWindow = true; //설정된 GUI 창이 나온다.
22                 labelText = ""; //labelText가 보이지 않게 한다.
23             }
24         }
25     }
26
27     private int playerScore = 0; //player가 준비를 죽이면 스코어가 올라가는 함수
28     public int Score //스코어 함수
29     {
30         get { return playerScore; } //스코어 값 다시 반복
31         set
32         {
33             playerScore = value; //스코어 함수 값 저장
34             if (playerScore >= 7) //스코어가 10이상이면
35             {
36                 winWindow = true; //설정된 GUI 창이 나온다.
37                 labelText = ""; //처음에 설정한 labelText 안보이게 설정
38             }
39         }
40     }
41 }
42
43

```

```

43
44
45 // Update is called once per frame
46
47 void OnGUI()
48 {
49     GUI.Box(new Rect(20, 20, 150, 25), "Player HP: " + playerHP); //GUI 박스 위치 및 이름 설정
50     GUI.Box(new Rect(750, 20, 150, 25), "Score: " + playerScore); //GUI 박스 위치 및 이름 설정
51     GUI.Label(new Rect(Screen.width / 2 - 100, Screen.height - 50, 350, 50), labelText); //GUI 박스 위치 및 이름 설정
52
53     if (dieWindow) //만약 dieWindow가 true면
54     {
55         if (GUI.Button(new Rect(Screen.width / 2 - 100, Screen.height / 2 - 50, 200, 100), "DIE")) //GUI Button 위치 및 이름 설정
56         {
57             SceneManager.LoadScene("OverNightScene"); //밤을 배경으로 하는 게임오버씬으로 전환
58         }
59     }
60
61     if (winWindow) //만약 winWindow가 true면
62     {
63         if (GUI.Button(new Rect(Screen.width / 2 - 100, Screen.height / 2 - 50, 200, 100), "곧 날이 밝아옵니다!")) //GUI Button 위치 및 이름 설정
64         {
65             SceneManager.LoadScene("PortionScene"); //물약씬으로 전환
66         }
67     }
68 }
69
70

```

## GameManger2.cs - 메인 게임 낫

```

GameManger2.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class GameManger2 : MonoBehaviour
7 {
8     public string labelText = "우측에 과일나무들이 있다. 들고있는 칼(1번키)로 나무를 베어보자!"; //labelText가 보이도록 설정
9
10     public bool showLoseWindow = false; //처음에는 보이지 않게 설정
11     public bool showFlowerWindow = false; //처음에는 보이지 않게 설정
12     public bool hungryWindow = true; //처음에 보이게 설정
13
14     private int playerHP = 5; //Player 체력 설정
15     public int HP //체력 함수
16     {
17         get { return playerHP; } //체력 값을 다시 받아온다.
18         set
19         {
20             playerHP = value; //체력 값을 저장한다.
21             if (playerHP <= 0) //만약 체력이 0 이면
22             {
23                 showLoseWindow = true; //GUI 버튼이 뜨도록 한다.
24             }
25         }
26     }
27
28     private int MysteriousFlower = 0; //신비의 꽃 갯수 설정
29     public int Flower //신비의 꽃 함수
30     {
31         get { return MysteriousFlower; } //신비의 꽃 값을 다시 받아온다.
32         set
33         {
34             MysteriousFlower = value; //신비의 꽃 값을 저장한다.
35             if (MysteriousFlower >= 20) //만약 신비의 꽃을 20개 이상 얻으면
36             {
37                 labelText = ""; //labelText가 보이지 않게 한다.
38                 showFlowerWindow = true; //GUI 버튼이 뜨도록 한다.
39             }
40         }
41     }
42
43     private int Food = 0; //과일 체력 설정
44     public int food //과일 체력 함수
45     {
46         get { return Food; } //과일 체력 값을 다시 받아온다.
47         set
48         {
49             Food = value; //과일 값을 저장한다.
50             if (Food >= 5) //만약 과일을 5개 이상 얻으면
51             {
52                 labelText = "이제 신비의 물약 재료에 필요한 물약 2개를 모으자! 곡괭이(4번키)로 바위를 깨면 물약이 나온다."; //labelText가 보이도록 한다.
53             }
54         }
55     }
56
57     private int Rock = 0; //바위 체력 설정
58     public int rock //바위 함수
59     {
60         get { return Rock; } //바위 체력 값을 다시 받아 온다.
61         set
62         {
63             Rock = value; //바위 값을 저장한다.
64             if (Rock >= 2) //바위에서 나온 아이템(물약)이 2개 이상이면
65             {
66                 labelText = "마지막으로 신비의 물약을 얻기 위한 재료인 꽃을 20송이 모으자!"; //labelText가 보이도록 한다.
67             }
68         }
69     }
70
71     // Update is called once per frame
72     void OnGUI()
73     {
74         GUI.Box(new Rect(20, 20, 150, 25), "Player HP: " + playerHP); //GUI 박스 위치 및 이름 설정
75         GUI.Box(new Rect(20, 50, 150, 25), "Food: " + Food); //GUI 박스 위치 및 이름 설정
76         GUI.Box(new Rect(20, 80, 150, 25), "Rock: " + Rock); //GUI 박스 위치 및 이름 설정
77         GUI.Box(new Rect(20, 110, 150, 25), "Mysterious Flower: " + MysteriousFlower); //GUI 박스 위치 및 이름 설정
78         GUI.Label(new Rect(Screen.width / 3, Screen.height - 40, 600, 100), labelText); //GUI Label 위치 설정
79
80         if (showLoseWindow) //만약 showLoseWindow가 true면
81         {
82             if (GUI.Button(new Rect(Screen.width / 2 - 100, Screen.height / 2 - 50, 200, 100), "DIE")) //GUI Button 위치 및 이름 설정
83             {
84                 SceneManager.LoadScene("OverScene"); //낫 게임 오버 신드로 전환
85             }
86         }
87     }
88
89
90

```

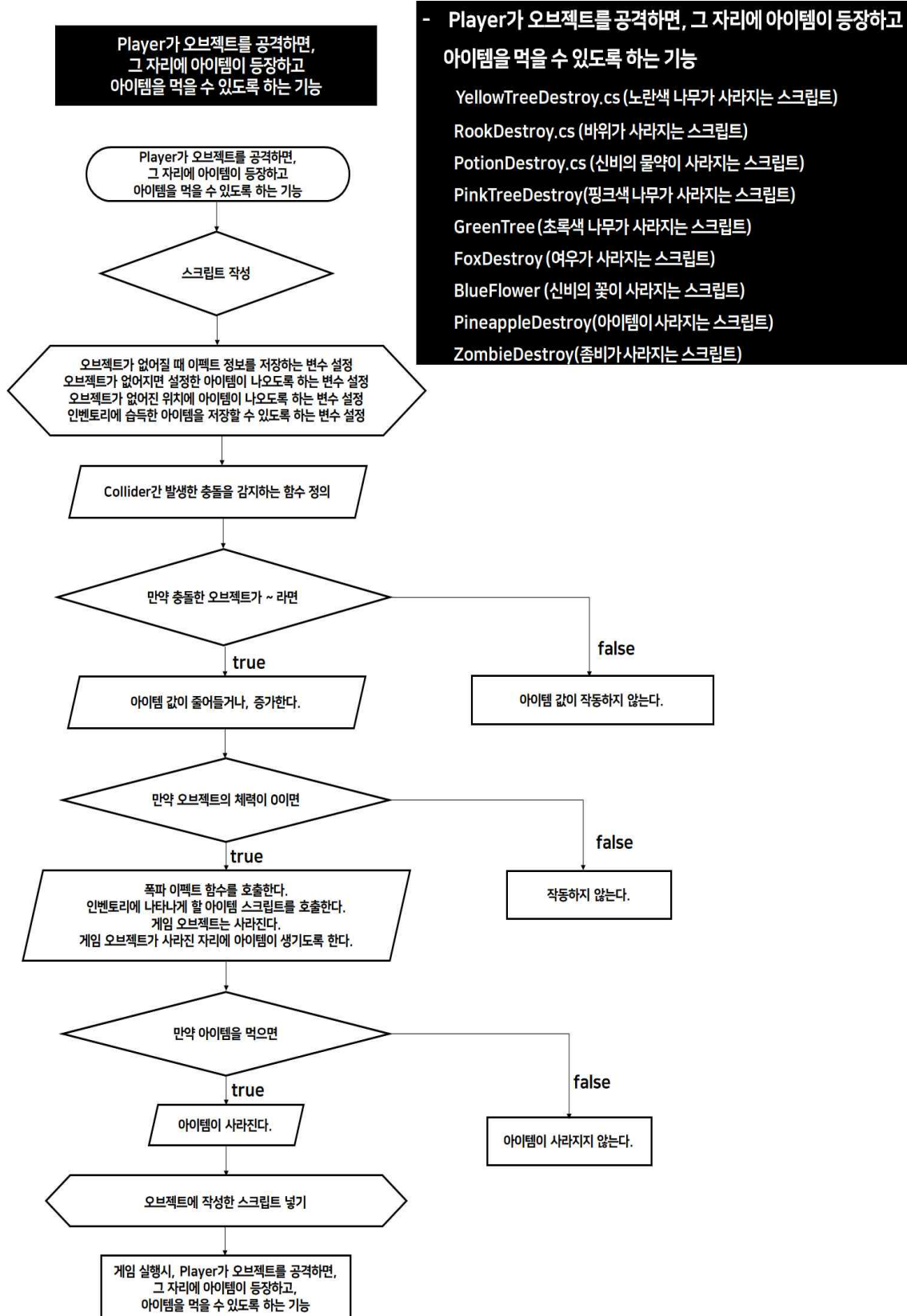
## GameManger3.cs - 신비의 물약

```
GameManger3.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class GameManger3 : MonoBehaviour
7 {
8     public string labelText = "신비의 물약이 완성되었습니다!"; //LabelText가 보이도록 설정
9
10    public bool showInWindow = false; //처음에는 보이지 않게 설정
11
12    private int Mysteriouspotion = 0; //신비의 물약 갯 설정
13    public int Mysteriouspotion //신비의 물약 갯 합수
14    {
15        get { return Mysteriouspotion; } //신비의 물약 갯을 다시 불러온다.
16        set
17        {
18            Mysteriouspotion = value; //신비의 물약 갯을 저장한다.
19
20            if (Mysteriouspotion >= 1) //만약 신비의 물약을 획득했을 경우
21            {
22                labelText = ""; //LabelText가 보이지 않게 한다.
23                showInWindow = true; //GUI 버튼이 뜨도록 한다.
24            }
25        }
26    }
27
28    void OnGUI()
29    {
30        GUI.Label(new Rect(Screen.width / 2 - 100, Screen.height - 50, 350, 50), labelText); //GUI Label 위치 설정
31
32        if (showInWindow)
33        {
34            if (GUI.Button(new Rect(Screen.width / 2 - 100, Screen.height / 2 - 50, 380, 100), "신비한 물약을 획득했습니다. 서둘러 밍에게 돌아가세요!")) //GUI Button 위치 및 이름 설정
35            {
36                SceneManager.LoadScene("In Scene"); //게임 성공 반으로 전환
37            }
38        }
39    }
40
41 }
42
```



## □ 순서도

(Player가 오브젝트를 공격하면, 그 자리에 아이템이 등장하고 아이템을 먹을 수 있도록 하는 기능)



## □ 설명

(Player가 오브젝트를 공격하면, 그 자리에 아이템이 등장하고 아이템을 먹을 수 있도록 하는 기능)

▶ Player가 오브젝트를 공격하면, 그 자리에 아이템이 등장하고, 아이템을 먹을 수 있도록 하는 기능을 추가한다.

▶ 다양한 오브젝트에 적용할 스크립트를 작성한다.

▶ 오브젝트가 없어질 때 이펙트 정보를 저장하는 변수를 설정한다.

▶ 오브젝트가 없어지면 설정한 아이템이 나오도록 하는 변수를 설정한다.

▶ 오브젝트가 없어진 위치에 아이템이 나오도록 하는 변수를 설정한다.

▶ 인벤토리에 습득한 아이템을 저장할 수 있도록 하는 변수를 설정한다.

▶ Collider간 발생한 충돌을 감지하는 함수를 정의한다.

▶ 만약 충돌한 오브젝트가 ( )라면

▶ 아이템 값이 줄어들거나, 증가한다(true)

▶ 아이템 값이 작동하지 않는다(false)

▶ 만약 오브젝트의 체력이 0이면

▶ 폭파 이펙트 함수를 호출한다.(true)

▶ 인벤토리에 나타나게 할 아이템 스크립트를 호출한다.(true)

▶ 게임 오브젝트는 사라진다.(true)

▶ 게임 오브젝트가 사라진 자리에 아이템이 생기도록 한다.(true)

▶ 게임 오브젝트가 사라지지 않는다. 아이템이 생기지 않고 이펙트도 생기지 않는다.(false)

▶ 만약 아이템을 먹으면

▶ 아이템이 사라진다.(true)

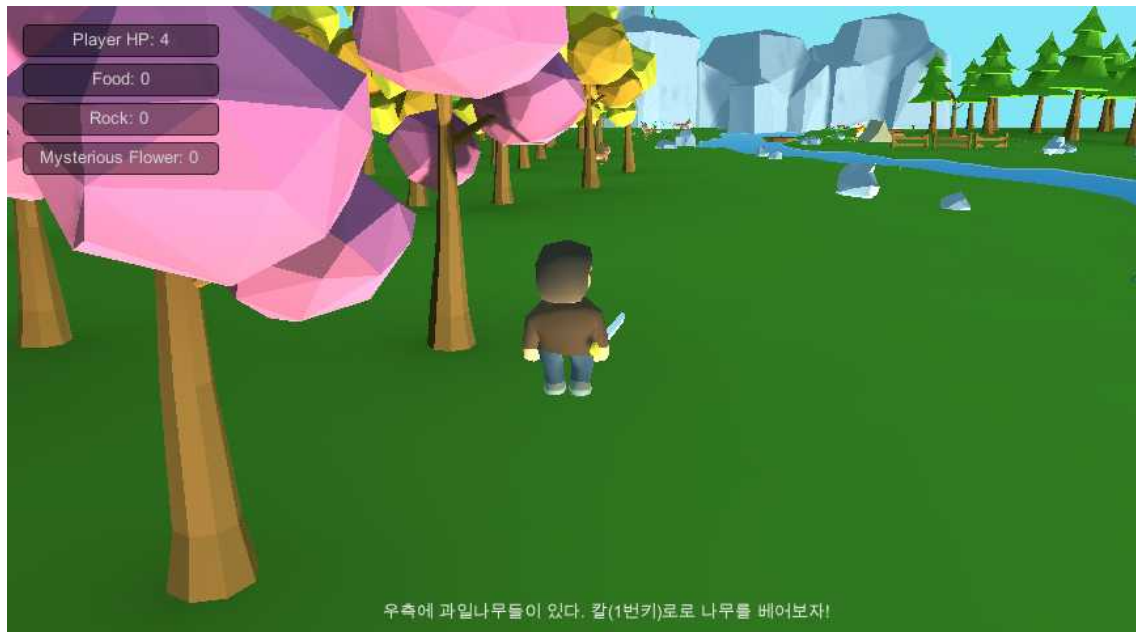
▶ 아이템이 사라지지 않는다.(false)

▶ 다양한 오브젝트에 작성한 스크립트를 넣는다.

▶ 게임 실행시 Player가 오브젝트를 공격하면, 그 자리에 아이템이 등장하고, 아이템을 먹을 수 있게 된다.

## □ 결과물

▶ Player가 오브젝트를 공격하면



▶아이템이 등장한다.



▶아이템을 섭취하면 아이템이 사라진다.



▶섭취한 아이템은 인벤토리에 저장된다.

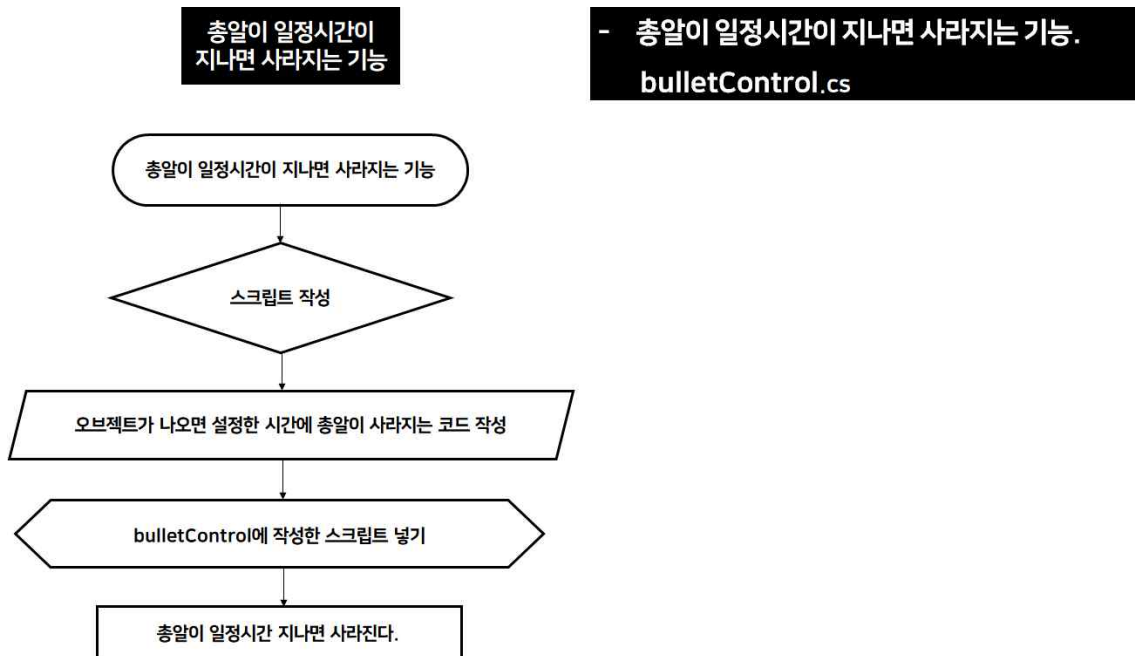


## □ 소스코드

(YellowTreeDestroy.cs - 노란색 나무가 사라지는 스크립트 외 9개)

```
YellowTreeDestroy.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class YellowTreeDestroy : MonoBehaviour
6 {
7     public ParticleSystem yellowtree; //나무가 없어질 때, 이펙트 정보를 저장하는 변수를 설정한다.
8     public GameObject PineApple; //나무가 없어지면 설정한 파인애플이 나오도록 하는 변수 설정
9     public Transform TreePost; //나무가 없어진 위치에 파인애플이 나오도록 하는 변수 설정
10
11     int YellowTreeHp = 3; //나무의 체력변수 설정한다.
12
13     [SerializeField]
14     private Inventory theInventory; //인벤토리에 습득한 아이템을 저장할 수 있도록 하는 변수 설정
15
16     void OnCollisionEnter(Collision col) //Collider간 발생한 충돌을 감지하는 함수 설정
17     {
18         if (col.gameObject.name == "e_sword_B") //만약 충돌하는 물체가 칼이라면
19         {
20             YellowTreeHp -- 1; //나무의 체력이 1 줄어든다.
21         }
22     }
23
24     // Update is called once per frame
25     void Update()
26     {
27         if (YellowTreeHp == 0) //만약 나무의 체력이 0이라면
28         {
29             ParticleSystem fire = Instantiate(yellowtree, transform.position, Quaternion.identity); //폭파 이펙트 함수 호출한다.
30             fire.Play();
31             theInventory.AcquireItem(GetComponent<ItemPickUp>().item); // ItemPickUp안에 item을 인벤토리에 저장한다.
32             Destroy(gameObject); //나무는 사라진다.
33             Instantiate(PineApple, TreePost.position, TreePost.rotation); //나무가 없어진 자리에 파인애플이 뜨는 함수 코드
34         }
35     }
36
37 }
```

## □ 순서도 (총알이 일정시간이 지나면 사라지는 기능)



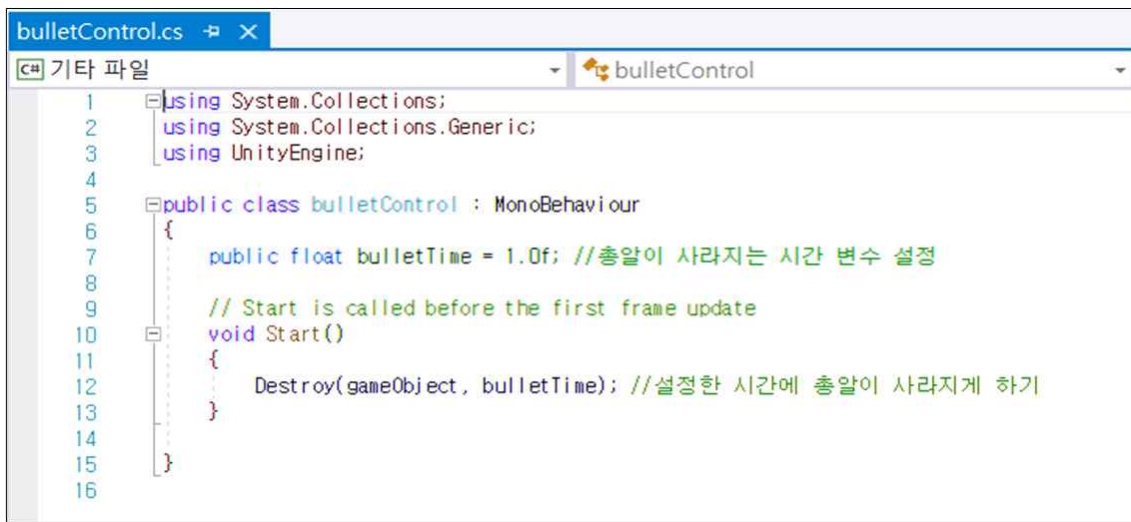
## □ 설명 (총알이 일정시간이 지나면 사라지는 기능)

- ▶ 총알이 일정시간이 지나면 사라지는 기능을 넣는다.
- ▶ 스크립트를 작성한다.
  - ▶ 오브젝트가 나오면 설정한 시간에 총알이 사라지는 코드 작성
- ▶ 작성한 스크립트를 총알에 넣는다.
- ▶ 총알을 발사했을 때, 총알이 일정시간이 지나면 사라진다.

☐ **결과물** (총알이 일정시간이 지나면 사라지는 기능)



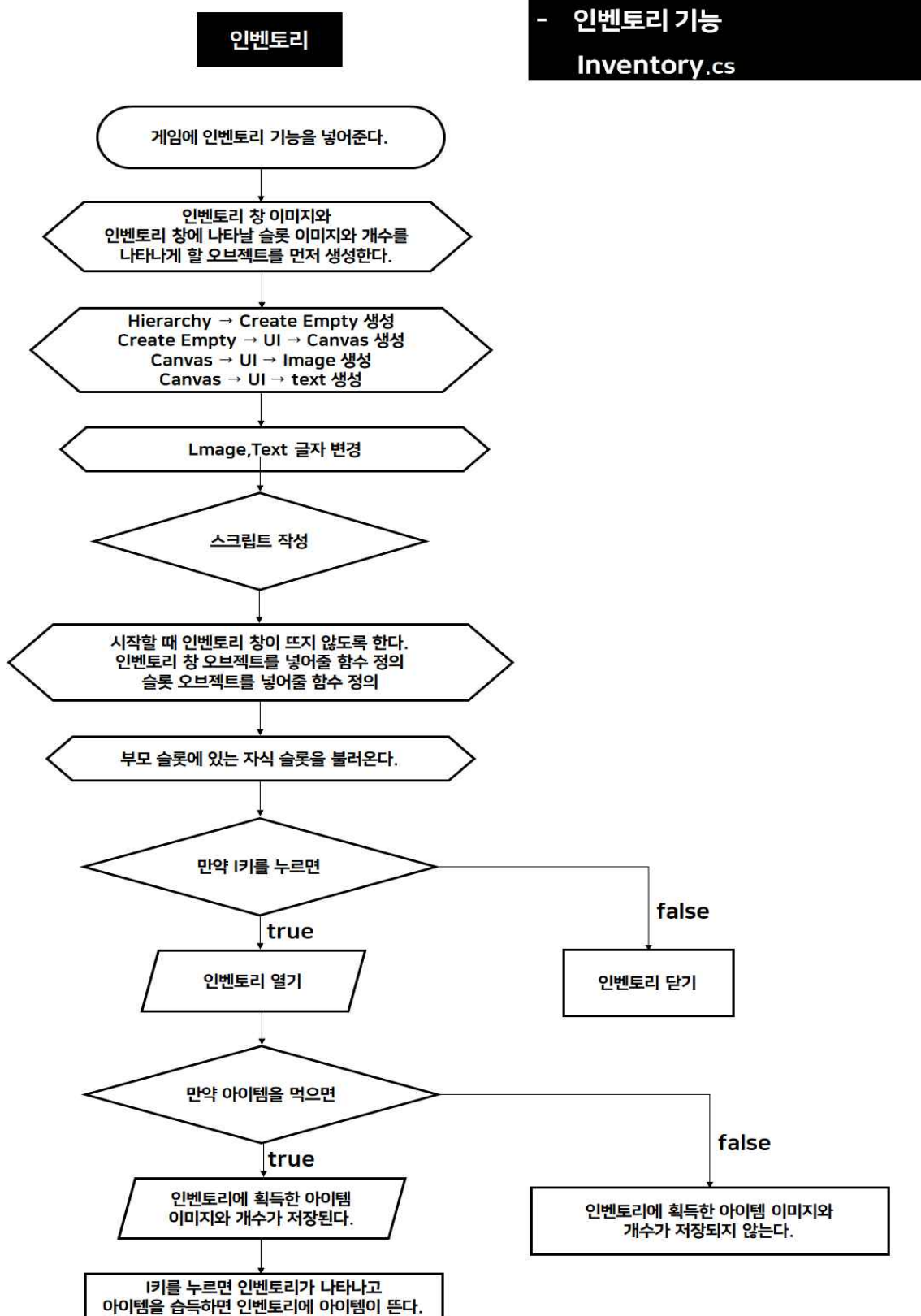
- ☐ 소스코드 (총알이 일정시간이 지나면 사라지는 기능)



```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class bulletControl : MonoBehaviour
6  {
7      public float bulletTime = 1.0f; //총알이 사라지는 시간 변수 설정
8
9      // Start is called before the first frame update
10     void Start()
11     {
12         Destroy(gameObject, bulletTime); //설정 한 시간에 총알이 사라지게 하기
13     }
14
15 }
16
```



## □ 순서도 (인벤토리 기능)



## □ 설명 (인벤토리 기능)

- ▶ 게임에 인벤토리 기능을 넣어준다.
- ▶ 우선 인벤토리 창 이미지와 인벤토리 창에 나타날 슬롯 이미지와 개수를 나타나게 할 오브젝트를 먼저 생성한다.
- ▶ Hierarchy창에 Create Empty을 생성한다.
- ▶ Create Empty에서 UI를 통해 Canvas 생성한다.
- ▶ Canvas에서 UI를 통해 Image 생성한다.
- ▶ Canvas에서 UI를 통해 text 생성한다.
- ▶ Limage,Text 글자 변경한다.
- ▶ 스크립트를 작성한다.
  - ▶ 시작할 E0 인벤토리 창이 뜨지 않도록 한다.
  - ▶ 인벤토리 창 오브젝트를 넣어줄 함수를 정의한다.
  - ▶ 슬롯 오브젝트를 넣어줄 함수를 정의한다.
  - ▶ 부모 슬롯에 있는 자식 슬롯을 불러온다.
  - ▶ 만약 I키를 누르면
  - ▶ 인벤토리를 연다(true)
  - ▶ 인벤토리가 열리지 않는다(false)
  - ▶ 만약 아이템을 먹지 않으면
  - ▶ 인벤토리에 획득한 아이템 이미지와 개수가 저장된다.(true)
  - ▶ 인벤토리에 획득한 아이템 이미지와 개수가 저장되지 않는다.(false)
  - ▶ 게임을 실행하면 인벤토리가 나타나고 아이템을 습득하면 인벤토리에 아이템이 뜬다.

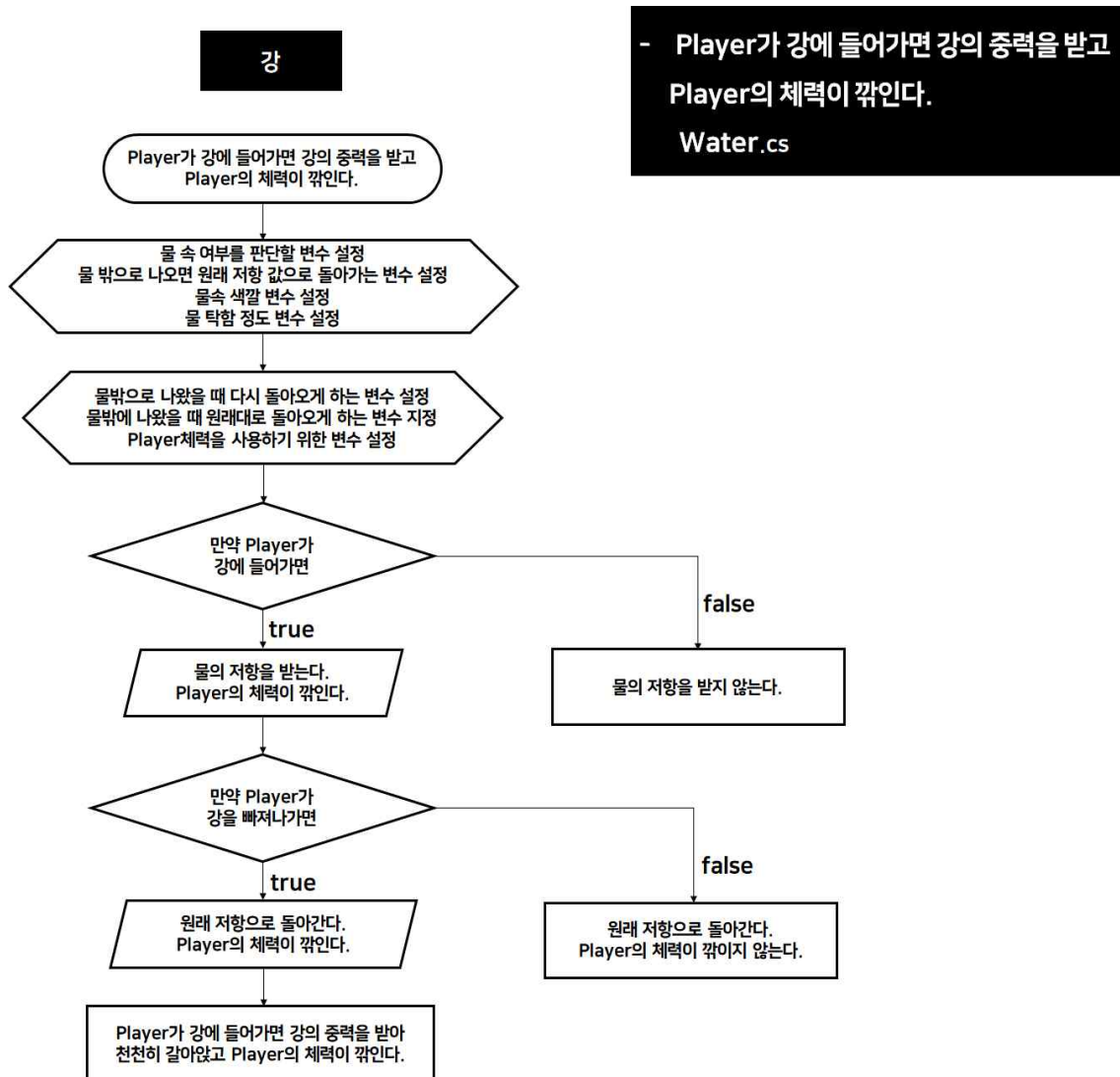
## □ 결과물 (인벤토리 기능)



## 소스코드 Inventory.cs

```
Inventory.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Inventory : MonoBehaviour
6 {
7     public static bool inventoryActivated = false; //시작할 때 인벤토리 창이 뜨지 않도록 한다.
8
9     // 필요한 컴포넌트
10    [SerializeField]
11    private GameObject go_InventoryBase; //인벤토리 창 오브젝트를 넣을 함수 정의
12    [SerializeField]
13    private GameObject go_SlotsParent; //슬롯 오브젝트를 넣을 함수 정의
14
15    private Slot[] slots; //슬롯들
16
17    // Start is called before the first frame update
18    void Start()
19    {
20        slots = go_SlotsParent.GetComponentInChildren<Slot>(); //부모 슬롯에 있는 자식 슬롯을 불러온다.
21    }
22
23    // Update is called once per frame
24    void Update()
25    {
26        TryOpenInventory();
27    }
28
29    private void TryOpenInventory()
30    {
31        if (Input.GetKeyDown(KeyCode.I)) //만약 i키를 누르면
32        {
33            inventoryActivated = !inventoryActivated;
34            if (inventoryActivated)
35                OpenInventory(); //인벤토리 열기
36            else
37                CloseInventory(); //인벤토리 닫기
38        }
39    }
40
41    private void OpenInventory()
42    {
43        go_InventoryBase.SetActive(true); //인벤토리 보이게 하기
44    }
45
46    private void CloseInventory()
47    {
48        go_InventoryBase.SetActive(false); //인벤토리 보이지 않게 하기
49    }
50
51    public void AcquireItem(Item _item, int _count = 1) //아이템을 불러온다.
52    {
53        for (int i = 0; i < slots.Length; i++) //아이템을 먹으면 슬롯에 계속해서 추가한다.
54        {
55            if (slots[i].item != null) //만약 아이템의 슬롯이 비었으면
56            {
57                if (slots[i].item.itemName == _item.itemName) //만약 아이템의 이름 슬롯이 비었으면
58                {
59                    slots[i].SetSlotCount(_count); //슬롯에 나타나는 아이템 갯수를 증가하거나 감소시켜준다.
60                    return;
61                }
62            }
63
64            for (int i = 0; i < slots.Length; i++) //아이템을 먹으면 슬롯에 계속해서 추가한다.
65            {
66                if (slots[i].item == null) //만약 아이템의 슬롯이 비었으면
67                {
68                    slots[i].AddItem(_item, _count); //슬롯에 나타나는 아이템 이미지를 증가하거나 감소시켜준다.
69                    return; //값을 되돌린다.
70                }
71            }
72        }
73    }
74
75
76
77 }
```

□ **순서도** (Player가 강에 들어가면 강의 중력을 받고 Player의 체력이 깎인다.)



□ **설명** (Player가 강에 들어가면 강의 중력을 받고 Player의 체력이 깎인다.)

- ▶ Player가 강에 들어가면 강의 중력을 받고 Player의 체력이 깎이도록 한다.
- ▶ 강 오브젝트에 넣어줄 스크립트를 작성한다.
  - ▶ 우선 물속 여부를 판단할 변수를 설정한다.
  - ▶ 물 밖으로 나오면 원래 저항 값으로 돌아가는 변수를 설정한다.
  - ▶ 물속 색 변수를 설정한다.
  - ▶ 물 탁함 정도 변수를 설정한다.
  - ▶ 물 밖으로 나왔을 때 다시 돌아오게 하는 변수를 설정한다.
  - ▶ 물 밖으로 나왔을 때 원래대로 돌아오게 하는 변수를 지정한다.
- ▶ 만약 Player가 강에 들어가면

- ▶ 물의 저항을 받는다.(true)
- ▶ Player의 체력이 깎인다.(true)
- ▶ 물의 저항을 받지 않는다.(false)
- ▶ 만약 Player가 강을 빠져나가면
- ▶ 원래 저항으로 돌아간다.
- ▶ Player의 체력이 깎인다.(true)
- ▶ 원래 저항으로 돌아간다.(true)
- ▶ Player의 체력이 깎이지 않는다.(false)
- ▶ 작성한 스크립트를 강 오브젝트에 넣어준다.
- ▶ 결과적으로 Player가 강에 들어가면 강의 중력을 받아 천천히 갈아 앉고 Player의 체력이 깎인다.

☐ **결과물** (Player가 강에 들어가면 강의 중력을 받고 Player의 체력이 깎인다.)





## □ 소스코드

Water.cs (Player가 강에 들어가면 강의 중력을 받고 Player의 체력이 깎인다.)

```

Water.cs -> zombieManager.cs
Assembly-CSharp -> Water

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  @Unity 스크립트 | 참조 0개
7  public class Water : MonoBehaviour
8  {
9
10     public static bool isWater = false; //물속여부 판단
11
12     [SerializeField] private float waterDrag; //물속 중력
13     private float orignDrag; //물밖으로 나오면 원래 저항값으로 돌아가기
14
15     [SerializeField] private Color WaterColor; //물속 색깔
16     [SerializeField] private float waterFogDensity; //물 탁함 정도
17
18     private Color originColor; //물밖에 나왔을 때 원래대로 돌아오게 하는 변수 지정
19     private float originFogDensity; //물밖에 나왔을 때 원래대로 돌아오게 하는 변수 지정
20
21     private GameManager2 gameManager; //Player체력을 사용하기 위한 변수 설정
22
23     // Start is called before the first frame update
24     @Unity 메시지 | 참조 0개
25     void Start()
26     {
27         originColor = RenderSettings.fogColor; //처음 색 값
28         originFogDensity = RenderSettings.fogDensity; //처음 색 값
29         gameManager = GameObject.Find("GameManager2").GetComponent<GameManager2>(); //GameManager2를 불러온다.
30         orignDrag = 0; //물의 저항을 0으로 시작한다.
31     }

```

```

38  @Unity 메시지 | 참조 0개
39  private void OnTriggerEnter(Collider other)
40  {
41      if (other.transform.tag == "Player") // 만약 강에 Player가 들어오면
42      {
43          GetWater(other); // GetWater를 불러온다.
44      }
45  }

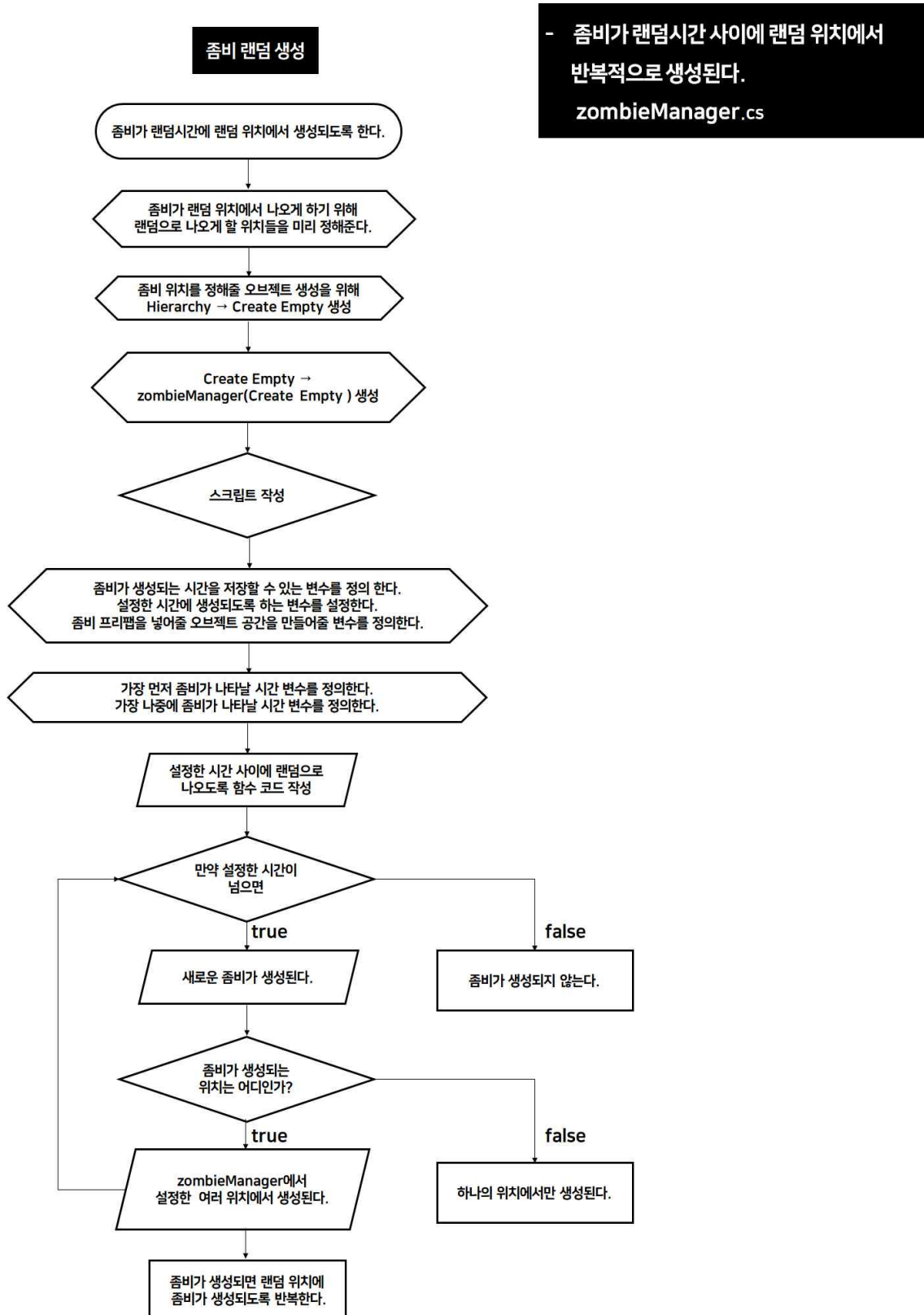
46  @Unity 메시지 | 참조 0개
47  private void OnTriggerExit(Collider other)
48  {
49      if (other.transform.tag == "Player") // 만약 강에서 Player가 나가면
50      {
51          GetOutWater(other); // GetOutWater를 불러온다.
52      }
53  }

54  참조 1개
55  private void GetWater(Collider _player) // 물에 들어갔을 때
56  {
57      isWater = true; // 물에 들어갔을 때 실행
58      _player.transform.GetComponent<Rigidbody>().drag = waterDrag; // player의 저항을 물의 저항으로 바꿔준다. (천천히 갈아얕음)
59      GameManager.HP -= 1;
60
61      RenderSettings.fogColor = WaterColor; // 물에 들어갔을 때 주변 색을 바꿔준다.
62      RenderSettings.fogDensity = waterFogDensity; // 물에 들어갔을 때 주변 색을 바꿔준다.
63  }
64
65  참조 1개
66  private void GetOutWater(Collider _player) // 물을 빠져나왔을 때
67  {
68
69      isWater = false; // 물에 나왔을 때
70      _player.transform.GetComponent<Rigidbody>().drag = originDrag; // player의 저항으로 바꿔준다.
71
72      RenderSettings.fogColor = originColor; // 물에서 나왔을 때 주변 색을 다시 원래대로 바꿔준다.
73      RenderSettings.fogDensity = originFogDensity; // 물에서 나왔을 때 주변 색을 다시 원래대로 바꿔준다.
74  }
75

```



□ 순서도 (좀비가 랜덤 시간에 사이에 랜덤 위치에서 반복적으로 생성된다.)



## □ 설명 (좀비가 랜덤 시간에 사이에 랜덤 위치에서 반복적으로 생성된다.)

- ▶ 좀비가 랜덤 시간에 랜덤 위치에서 반복적으로 생성되도록 한다.
- ▶ 좀비가 랜덤 위치에서 나오게 하기 위해 랜덤으로 나오게 할 위치들을 미리 정해준다.
- ▶ 좀비 위치를 정해줄 오브젝트 생성을 위해 Hierarchy창에서 Create Empty 생성한다.
- ▶ Create Empty에서 Create Empty를 하나 더 생성하여 이름을 zombieManager으로 변경한다.
- ▶ 이제 zombieManager에 넣어줄 스크립트를 작성한다.
  - ▶ 우선 좀비가 생성되는 시간을 저장할 수 있는 변수를 정의한다.
  - ▶ 설정한 시간에 생성되도록 하는 변수를 설정한다.
  - ▶ 좀비 프리팹을 넣어줄 오브젝트 공간을 만들어줄 변수를 정의한다.
  - ▶ 가장 먼저 좀비가 나타날 시간 변수를 정의한다.
  - ▶ 가장 나중에 좀비가 나타날 시간 변수를 정의한다.
  - ▶ 설정한 시간 사이에 랜덤으로 나오도록 함수 코드를 작성한다.
  - ▶ 만약 설정한 시간이 넘으면
    - ▶ 새로운 좀비가 생성된다.(true)
    - ▶ 만약 설정한 시간이 넘지 않으면
      - ▶ 새로운 좀비가 생성되지 않는다.(false)
      - ▶ 좀비가 생성되는 위치는 zombieManager에서 설정한 여러 위치에서 생성된다.(true)
      - ▶ 그렇지 않으면 하나의 위치에서만 좀비가 생성된다.(false)
- ▶ 작성한 스크립트를 zombieManager에 넣어준다.
- ▶ 좀비가 생성되면 랜덤 위치에 좀비가 생성되도록 반복한다.

## □ 결과물 (좀비가 랜덤 시간에 사이에 랜덤 위치에서 반복적으로 생성된다.)



## □ 소스코드 - zombieManager.cs

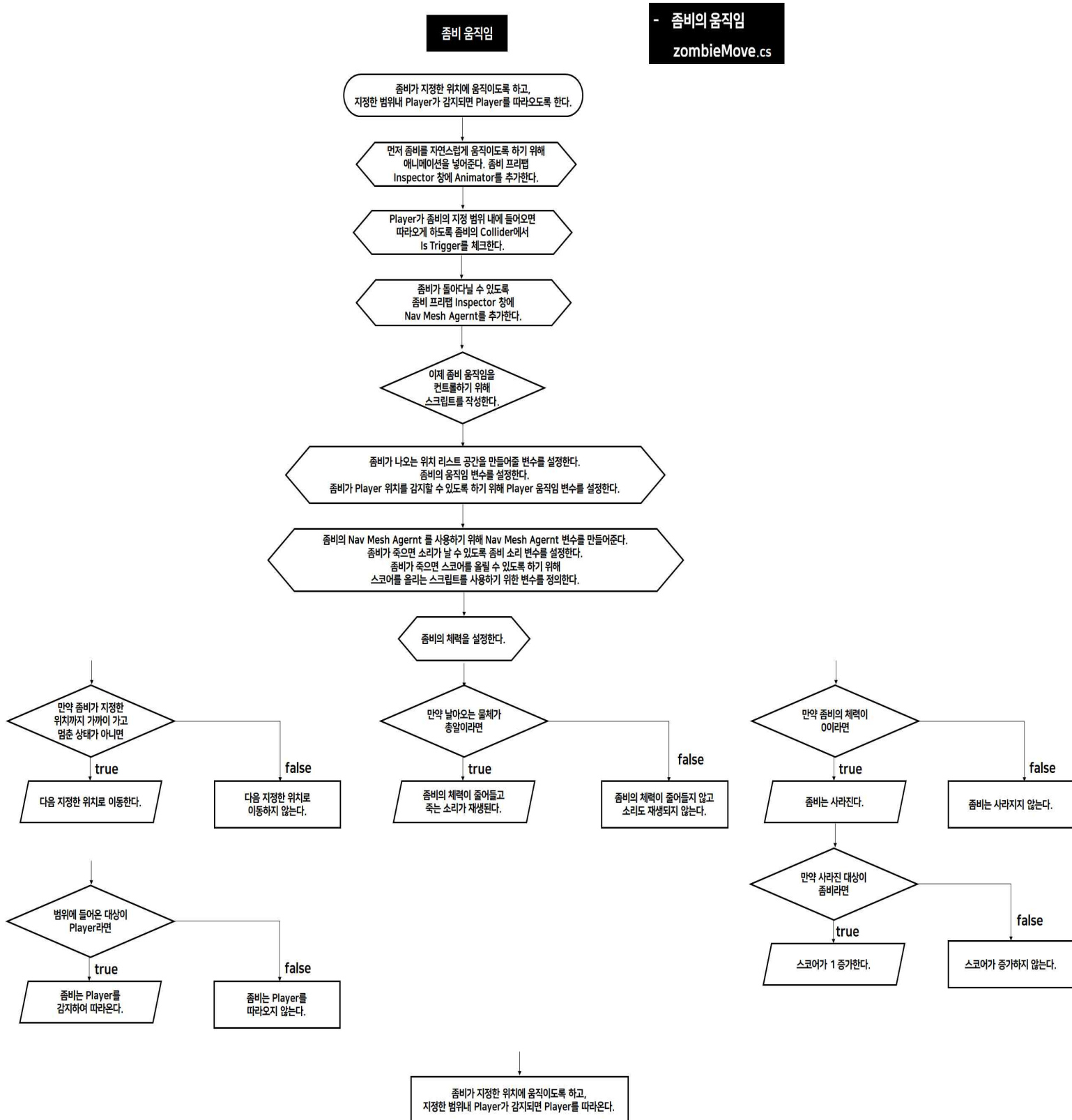
(좀비가 랜덤 시간에 사이에 랜덤 위치에서 반복적으로 생성된다.)

```
zombieManager.cs - X
Assembly-CSharp - zombieManager

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  @Unity 스크립트 |참조 0개
6  public class zombieManager : MonoBehaviour
7  {
8      float currentTime; //시간을 저장할 수 있는 변수 정의
9      public float createTime = 1.0f; //설정된 시간에 생성되도록 설정
10     public GameObject zombie; // 대상은 좀비 게임 오브젝트 생성
11     public float minTime = 5.0f; //가장 먼저 나타나는 시간
12     public float maxTime = 15.0f; //가장 늦게 나타나는 시간
13
14     // Start is called before the first frame update
15     @Unity 메시지 |참조 0개
16     void Start()
17     {
18         currentTime = 0.0f; //처음에는 0부터 시작
19         createTime = UnityEngine.Random.Range(minTime, maxTime); //설정된 시간 사이에 랜덤으로 나오도록 설정
20
21     }
22
23     // Update is called once per frame
24     @Unity 메시지 |참조 0개
25     void Update()
26     {
27         currentTime += Time.deltaTime; //한번 생성되고, 걸리는 시간
28
29         if (currentTime > createTime) //설정된 시간이 넘었을 때,
30         {
31             GameObject newzombie = Instantiate(zombie); //게임오브젝트의 새로운 좀비가 생성된다.
32             newzombie.transform.position = transform.position; //위치 zombieManager에서 설정한 위치로 한다.
33
34             currentTime = 0.0f; //업데이트할 때마다 1초를 세는 방법으로 생성된다.
35             createTime = UnityEngine.Random.Range(minTime, maxTime); //다시 생성하면 랜덤으로 좀비를 생성하도록 반복
36         }
37     }
38 }
```

## □ 순서도

(좀비가 랜덤 시간 사이에 랜덤 위치에서 반복적으로 생성되고 이동중 Player가 감지되면 Player를 따라온다.)



## □ 설명

(좀비가 랜덤 시간 사이에 랜덤 위치에서 반복적으로 생성되고 이동중 Player가 감지되면 Player를 따라온다.)

- ▶ 좀비가 랜덤으로 생성되고 위치도 랜덤으로 계속해서 생성되도록 한다.
- ▶ 먼저 좀비를 자연스럽게 움직이도록 하기 위해 애니메이션을 넣어준다.
- ▶ 좀비 프리팹 Inspector창에서 Animator를 추가한다.
- ▶ Player가 좀비의 지정 범위 내에 들어오면 따라오게 하도록 좀비의 Collider에서 Is Tigger를 체크한다.
- ▶ 다음 좀비가 돌아다닐 수 있도록 좀비 프리팹 Inspector창에서 Nav Mesh Agernt를 추가한다.
- ▶ 이제 좀비 움직임을 컨트롤 하기 위해 스크립트를 작성한다.
  - ▶ 좀비가 나오는 위치 리스트 공간을 만들어줄 변수를 설정한다.
  - ▶ 좀비의 움직임 변수를 설정한다.
  - ▶ 좀비가 Player 위치를 감지할 수 있도록 하기 위해 Player 움직임 변수를 설정한다.
  - ▶ 좀비의 Nav Mesh Agernt를 사용하기 위해 Nav Mesh Agernt 변수를 만들어준다.
  - ▶ 좀비가 죽으면 소리가 날 수 있도록 좀비 소리 변수를 설정한다.
  - ▶ 스코어를 올리는 스크립트를 사용하기 위한 변수를 정의한다.
  - ▶ 좀비의 체력을 설정한다.
  - ▶ 만약 날아오는 물체가 총알이라면
  - ▶ 좀비의 체력이 줄어들고 죽는 소리가 재생된다.(true)
  - ▶ 좀비의 체력이 줄어들지 않고 소리도 재생되지 않는다.(false)
  - ▶ 만약 좀비의 체력이 0이라면
  - ▶ 좀비는 사라진다.(true)
  - ▶ 좀비는 사라지지 않는다.(false)
  - ▶ 만약 사라진 대상이 좀비라면
  - ▶ 스코어가 1 증가한다.(true)
  - ▶ 스코어가 증가하지 않는다.(true)
  - ▶ 만약 좀비가 지정한 위치까지 가까이 가고 멈춘 상태가 아니라면
  - ▶ 다음 지정한 위치로 이동한다.(true)
  - ▶ 다음 지정한 위치로 이동하지 않는다.(false)
  - ▶ 범위에 들어온 대상이 Player라면
  - ▶ 좀비는 Player를 감지하여 따라온다.(true)
  - ▶ 좀비는 Player를 따라오지 않는다.(false)
- ▶ 좀비 움직임을 위해 작성한 스크립트를 좀비 프리팹에 넣어준다.
- ▶ 좀비가 지정한 위치에서 계속 움직이고 지정한 범위 내 Plater가 감지되면 Player를 따라온다.

## □ 결과물

(좀비가 랜덤 시간 사이에 랜덤 위치에서 반복적으로 생성되고 이동중 Player가 감지되면 Player를 따라온다.)



## □ 소스코드

(좀비가 랜덤 시간 사이에 랜덤 위치에서 반복적으로 생성되고 이동중 Player가 감지되면 Player를 따라온다.)

```
zombieMove.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.AI;
5
6 public class zombieMove : MonoBehaviour
7 {
8     public List<Transform> locations; //좀비가 나오는 위치 설정
9     public Transform patrolRoute; //움직임 변수 설정
10    public Transform player; //Player 움직임
11
12    private int locationIndex = 0; //처음에는 0으로 설정
13    private NavMeshAgent zombie; //좀비의 NavMeshAgent 변수를 만들어준다.
14    private AudioSource zombiedie; //좀비 소리 변수 설정
15
16    private GameManager gameManager; //GameManager 스크립트 안에 있는 스코어 함수를 사용하기 위한 변수를 정의한다.
17
18    private int lives = 3;
19
20    void OnCollisionEnter(Collision col) //Collider간 발생한 충돌을 감지하는 함수 설정
21    {
22        if (col.gameObject.tag == "bullet") //만약 날아오는 물체가 총알이라면
23        {
24            lives -= 1; //좀비 체력이 1 줄어든다.
25            zombiedie.Play(); //죽는 소리 재생
26        }
27    }
28
29    // Start is called before the first frame update
30    void Start()
31    {
32        player = GameObject.Find("Player").transform; //player를 불러온다.
33        gameManager = GameObject.Find("GameManager").GetComponent<GameManager>(); //GameManager 스크립트를 사용하기 위해 GameManager 오브젝트를 불러온다.
34        zombiedie = GetComponent<AudioSource>(); //좀비가 가지고 있는 소리를 불러온다.
35        zombie = GetComponent<NavMeshAgent>(); //좀비가 가지고 있는 NavMeshAgent를 불러온다.
36        patrolRoute = GameObject.Find("Route1").transform; //설정된 위치에 좀비가 나오도록 Route1를 불러온다.
37        InitializePatrolRoute(); //좀비가 나오는 위치 리스트를 사용하기 위한 함수
38        MoveToNextPatrolPoint(); //다음 위치로 이동하기 위한 함수
39    }
40}
```

```

42 void Update()
43 {
44     if (zombie.remainingDistance < 0.1f && !zombie.pathPending) //좀비가 일정위치까지 가까이 가고, 좀비가 멈춘 상태가 아닐때
45     {
46         MoveToNextPatrolPoint(); //다음 지정한 포지션으로 이동
47     }
48
49     if (lives <= 0) //만약 체력이 0이라면
50     {
51         GameObject.Find("zombie(Clone)").GetComponent().Play(); //소리가 재생된다.
52         Destroy(this.gameObject); //좀비는 사라진다.
53
54         if (this.gameObject.name == "zombie(Clone)") //만약 사라진 대상이 좀비 프리팹이면
55         {
56             gameManager.Score += 1; //스코어가 1 증가한다.
57         }
58     }
59 }
60
61
62 void InitializePatrolRoute()
63 {
64     foreach (Transform child in patrolRoute) //위치를 설정한 patrolRoute 안에 있는 child 움직임
65     {
66         locations.Add(child); //child를 저장한다.
67     }
68 }
69 void MoveToNextPatrolPoint()
70 {
71     zombie.destination = locations[locationIndex].position; //다음 리스트로 이동하기 위한 할수 코드
72     locationIndex = (locationIndex + 1) % locations.Count; //반복적으로 이동할 수 있게 하는 할수 코드
73 }
74
75
76 private void OnTriggerEnter(Collider collider)
77 {
78     if (collider.name == "Player") //범위안에 들어온 대상이 플레이어라면
79     {
80         zombie.destination = player.position; //좀비는 플레이어를 감지하여 따라온다.
81     }
82 }

```



## □ 생성된 파일들

Unity > MysteriousForest > Assets

Assets 검색

이름	수정된 날짜	유형	크기
Animator	2021-06-07 오전 5:44	파일 폴더	
Explosion	2021-06-05 오후 11:54	파일 폴더	
Fox	2021-06-02 오후 11:07	파일 폴더	
FREE Food Pack	2021-06-04 오전 12:28	파일 폴더	
HuaYao	2021-06-02 오후 10:53	파일 폴더	
Image	2021-06-08 오후 5:26	파일 폴더	
Inventory	2021-06-06 오전 12:40	파일 폴더	
Item	2021-06-07 오후 7:31	파일 폴더	
ItemPrefabs	2021-06-07 오후 7:34	파일 폴더	
Scenes	2021-05-31 오전 11:10	파일 폴더	
SimpleLowPolyNature	2021-05-31 오전 11:15	파일 폴더	
Sripts	2021-06-09 오후 5:19	파일 폴더	
ToonyTinyPeople	2021-06-03 오전 4:26	파일 폴더	
Animator.meta	2021-06-02 오후 8:57	META 파일	1KB
big one	2016-06-30 오후 2:07	WAV 파일	563KB
big one.wav.meta	2021-06-04 오후 2:26	META 파일	1KB
DM-CGS-45	2016-01-26 오전 8:53	WAV 파일	522KB
DM-CGS-45.wav.meta	2021-06-05 오전 1:01	META 파일	1KB
DM-CGS-46	2016-01-26 오전 8:53	WAV 파일	99KB
DM-CGS-46.wav.meta	2021-06-05 오전 1:01	META 파일	1KB
Explosion.meta	2021-06-04 오전 2:47	META 파일	1KB
Fling_1	2016-07-10 오전 3:25	WAV 파일	125KB
Fling_1.wav.meta	2021-06-04 오후 5:36	META 파일	1KB
Fox.meta	2021-06-02 오후 11:07	META 파일	1KB
FRFF Food Pack meta	2021-06-04 오전 12:28	META 파일	1KB