# C# JellyFarm

지원자 : 김현지

작성일 : 2022년 11월 15일

# 프로젝트 설명

**게임 장르** : 클리커 방치형 게임
**엔진 버전** : Unity 2021.3.12f1

**게임 설명** :
젤리를 클릭하여 젤라틴을 얻고 매매하여 골드를 얻습니다.
2D 게임을 공부하고자 유튜브의 강의 영상을 보면서 시작하게 되었습니다.
Bolt 로 구현된 게임을 Script를 작성하여 기능을 구현하였습니다.

**프로젝트 소스 링크** : https://github.com/asterism1030/2D-JellyFarm

# 플레이 화면


5) 젤리 구매 및 해금


6)젤리 수용량 및 생산량 업그레이드


2) 다른 젤리를 해금할 수 있는 젤라틴을 얻음

4) 방치를 해도 골드를 얻음

1) 생성 된 젤리를 누르면

3) 젤리를 아이콘 위로 드래그 드랍을 하면 골드를 얻음

# Script 구조

<Manager> : 싱글톤
- GameManager.cs : 게임 상에서의 동작
- GameInfo.cs : 게임 상에서의 정보
- UserInfo.cs : 플레이 유저에 대한 정보
- SoundManager.cs : 사운드

<Event> : EventTrigger 등에서 사용
- JellyEvent.cs : 젤리에 대한 전반적 이벤트
- JellyPanelEvent.cs : 특정 판넬 동작 구현
- PanelEvent.cs : 판넬의 기본적인 동작 (숨기기, 보이기)

<Object> : GameObject 의 직접적인 동작에 사용
- Scrolling.cs : 배경의 구름 이동
- **AI.cs** : **젤리의 동작**

(Project 상에서의 Script 분류는 씬에서의 부모 오브젝트 이름을 따라감)

# AI

**작성 과정** :

젤리의 동작 구현(대기 – 걷기 의 반복) 에 대해 IEnumerator 를 최대한 사용 안하고 가독성이 좋게 구현할 지 고민함.

자신의 머리에서 나온 결과물과 유사한 실제 상태 머신 패턴이 있다는 것을 알고 깊은 인상을 받았음.

**기능** :

GameObject 의 생성 및 파괴 시에 대한 동작 처리
젤리의 상태머신(대기 – 걷기) 동작
젤리에 대한 정보(가격, 레벨) 처리
이벤트 처리(드래그 & 드롭) 구현

# AI.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.EventSystems;
using UnityEngine.UI;
#if UNITY_EDITOR
using UnityEditor;
#endif
using EnumManager;

namespace EnumManager {
   public enum State
   {
      doNothing,  // 아무것도 안함
      doWaiting, // 일반 대기
      doWalking,
      doCounting, // 동작 멈추고 시간 잼
   }
}

  void Update()
   {

      DoAI();

   }

   bool AITimer(float setTime)
   {
      timer += Time.deltaTime;
      if(timer <= waitTime) {
         return true;
      }

      startTime = Time.deltaTime;
      timer = 0.0f;
      waitTime = setTime;

      return false;
   }
```

```
   void DoAI()
   {
      // AI 동작
      switch(CURSTATE) {
         case State.doNothing:
            break;
         case State.doWaiting:
            {
               bool isTimerRunning = AITimer(walkingDuration);

               if(isTimerRunning == true) {
                  break;
               }

               speedX = Random.Range(-0.2f, 0.2f);
               speedY = Random.Range(-0.2f, 0.2f);

               animator.SetBool(walkingTrigger, true);
               CURSTATE = State.doWalking;
            }
            break;
         case State.doWalking:
            {
               bool isTimerRunning = AITimer(Random.Range(3.0f, 5.0f));

               if(isTimerRunning == true) {
                  Walking();
                  break;
               }

               animator.SetBool(walkingTrigger, false);
               CURSTATE = State.doWaiting;
            }
            break;
         case State.doCounting:
            {
               AITimer(0.0f);
               CURSTATE = State.doNothing;
            }
            break;
         default:
            break;
      }
   }
```

전체 코드 :
https://github.com/asterism1030/2D-JellyFarm/blob/main/Assets/Script/Jelly/AI.cs

# 후기

클린한 코드를 위해서 다양한 패턴을 알아두는 것도 좋다고 생각했음

(상태머신 패턴을 알았다면 시간이 많이 단축됐을 것)

개인적으론 1인 플젝(TimeAttack)을 하던 때에 비해 게임의 장르와 기능이 명확하고 에셋이 제공되어 있어서 개발하는데 더 집중할 수 있어서 좋았음