

목차

1번 프로젝트(C++) - 자판기 시스템

2번 프로젝트(C++) - 미로 찾기 게임

3번 프로젝트(Unity) - Save The War Ship(슈팅게임)

4번 프로젝트(Unity + OpenCV Sharp) - AR 3D 모형 세우기

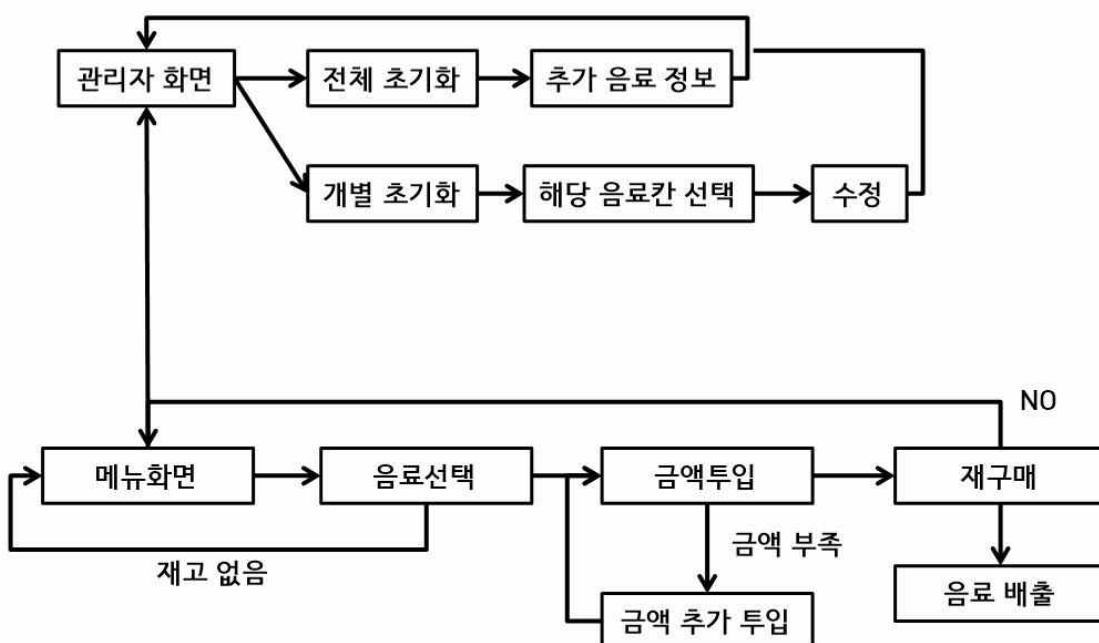
1번 프로젝트(C++) - 자판기 시스템

자판기 시스템	
언어	C++
개발 도구	Visual Studio 2019
개발 기간	2019.07.03. - 07.06
개발 인원	3명(역할: 결제 기능, UI)

프로젝트 개요

일상생활에서 사용하는 기본적인 시스템을 C++ 스타일로 구현

시스템 구성도



2번 프로젝트(C++) - 미로 찾기 게임


미로 찾기 게임		
언어		C++
개발 도구		Visual Studio 2019
개발 기간		2019.07.18. - 07.22
개발인원 (2명)	정승욱(팀장)	플레이어 기능, 미로 및 게임 구현
	이우민	미로구현 및 오류수정

프로젝트 개요

난이도 조절이 가능하고 WSAD 버튼으로 방향을 조절하며 스테이지 종료 후 점수크기 별로 저장해 보여준다.

프로젝트 목적

1번 프로젝트 보다 조금 더 C++ 같이 클래스와 cpp 분리 코딩 연습을 위해 프로젝트 진행

맵	
1. 게임화면	2. 맵 생성 함수
	<pre>//난이도 Easy 맵 void Map_Print_Easy(char Map_Easy[EASY_X][EASY_Y]) { Player p; p.SetEnd(19, 19); clock_t start = 0, end = 0; start = clock(); //타이머 시작 strcpy_s(Map_Easy[0], "00000000000000000000"); strcpy_s(Map_Easy[1], "020111111111110111110"); strcpy_s(Map_Easy[2], "010100000000010100010"); strcpy_s(Map_Easy[3], "010101111101110111010"); strcpy_s(Map_Easy[4], "010101010001000101010"); strcpy_s(Map_Easy[5], "010101010111011101010"); strcpy_s(Map_Easy[6], "010101010100000001010"); strcpy_s(Map_Easy[7], "011101010101111111010"); strcpy_s(Map_Easy[8], "00000101010100000010"); strcpy_s(Map_Easy[9], "011101010101011111010"); strcpy_s(Map_Easy[10], "000101010101000100010"); strcpy_s(Map_Easy[11], "011101011101110111010"); strcpy_s(Map_Easy[12], "010101000000010101010"); strcpy_s(Map_Easy[13], "010101011111110101110"); strcpy_s(Map_Easy[14], "010101000101000100000"); strcpy_s(Map_Easy[15], "010101110101011111110"); strcpy_s(Map_Easy[16], "010100010101000000010"); strcpy_s(Map_Easy[17], "010101110101110111110"); strcpy_s(Map_Easy[18], "010001000000010100000"); strcpy_s(Map_Easy[19], "011111111111110111130"); strcpy_s(Map_Easy[20], "00000000000000000000");</pre>
2-2. 맵생성 함수	설명
	1. 문자열 2차원 배열 EASY_X, EASY_Y를 생성해 그림 2.2 반복문 안에서 0,1,2,3 숫자에 맞춰 문자를 생성해 맵 생성 및 색상을 입힌다.

```

while (1)
{
    system("cls");

    for (int i = 0; i < EASY_X; i++)
    {
        for (int j = 0; j < EASY_Y; j++)
        {
            if (p.GetNow_X() == j && p.GetNow_Y() == i)
            {
                setcolor(10, 0);
                cout << "▼";
            }
            else if (Map_Easy[i][j] == '0')
            {
                setcolor(15, 0);
                cout << "■";
            }
            else if (Map_Easy[i][j] == '1')
            {
                cout << " ";
            }
            else if (Map_Easy[i][j] == '2')
            {
                setcolor(10, 0);
                cout << "▽";
            }
            else if (Map_Easy[i][j] == '3')
            {
                setcolor(12, 0);
                cout << "◎";
            }
        }
    }
}

```

플레이어 이동

1. 이동키 인식

```

void MovePlayer_Hard(char Map_difficulty[HARD_X][HARD_Y], Player& p, char input) //플레이어를 움직이는 함수
{
    switch (input)
    {
        case 'w':
        case 'W':
            MoveUp_Hard(Map_difficulty, p);
            break;
        case 's':
        case 'S':
            MoveDown_Hard(Map_difficulty, p);
            break;
        case 'a':
        case 'A':
            MoveLeft_Hard(Map_difficulty, p);
            break;
        case 'd':
        case 'D':
            MoveRight_Hard(Map_difficulty, p);
            break;
    }
}

```

설명

2. 이동 제한 사항

```

void MoveUp_Hard(char Map_difficulty[HARD_X][HARD_Y], Player& p)
{
    if (p.GetNow_Y() - 1 >= 0)
    {
        //벽인지 체크
        if (Map_difficulty[p.GetNow_Y() - 1][p.GetNow_X()] != '0')
        {
            p.SetNow_Y(-1);
        }
        if (Map_difficulty[p.GetNow_Y()][p.GetNow_X()] == '4')
        {
            p.Attack();
            p.MonsterKill();
            Map_difficulty[p.GetNow_Y()][p.GetNow_X()] = '1';
        }
        if (Map_difficulty[p.GetNow_Y()][p.GetNow_X()] == '6')//벽일때
        {
            if (p.GetKey() > 0)
            {
                Map_difficulty[p.GetNow_Y()][p.GetNow_X()] = '1';
                p.SetKey(-1);
            }
            if (p.GetKey() == 0)
            {
                p.SetNow_Y(-1);
            }
        }
        if (Map_difficulty[p.GetNow_Y()][p.GetNow_X()] == '5')//키를 획득했을때
        {
            p.SetKey(1);
            Map_difficulty[p.GetNow_Y()][p.GetNow_X()] = '1';
        }
    }
}

```

WSAD 키로 이동을 하고 switch 문을 이용해 입력하는 키를 _getchar 함수로 입력과 동시에 실행하도록 처리, 크기를 해당 난이도 크기만큼 설정해 주고 벽과 같은 장애물을 만났을 시에 진행하지 못하게 처리

플레이어 클래스

1. 플레이어 클래스

설명

```
#pragma once
#include <iostream>

using namespace std;

class Player
{
public:
    Player();
    ~Player();

public:
    void SetHP(int add_HP); //체력 셋팅
    int GetHP(); //체력 받아오기

    void SetScore(int add_score); //점수 셋팅하기
    int GetScore(); //점수얻어오기

    int GetStart_X(); //시작점 X좌표
    int GetStart_Y(); //시작점 Y좌표

    void SetEnd(int add_pos_x, int add_pos_y); //종료점 셋팅하기
    int GetEnd_X(); //종료점 X좌표
    int GetEnd_Y(); //종료점 Y좌표

    void SetNow_X(int add_pos_x); //현재위치 x셋팅
    void SetNow_Y(int add_pos_y); //현재위치 y셋팅
    int GetNow_X(); //현재위치 x
    int GetNow_Y(); // 현재위치 y

    int GetMonsterKill(); //몬스터 처치수
    void MonsterKill(); //몬스터 셋팅

    int GetKey(); //키 갯수
    void SetKey(int add_key); //키 획득

    void clear(); //Player 클래스 초기화

public:
    void Attack(); //공격시 Set함수에 -10을 넣어줌

private:
    int HP; //체력
    int nowPos_X, nowPos_Y; //플레이어 현재 위치
    int startPos_X, startPos_Y; //미로 이동시 플레이어의 시작위치
    int endPos_X, endPos_Y; //미로 이동시 플레이어의 끝 위치
    int score; //플레이어의 점수
    int Monster; //몬스터 클리어 수
    int Key; // 문을 열기위한 키 갯수
};
```

플레이어 클래스를 만들어 체력, 점수, 시작점 및 끝점, 현재위치 몬스터, 키, 공격 ,초기화 함수 등 플레이어와 연관된 기능을 작성

점수 저장

1. 점수 저장 화면

2. 저장 함수

도착했습니다. 완료시간은 : 26 초 입니다.
점수는 3600 점 입니다.
점수 입력을 위해 이름을 입력해 주세요 : seungwook2
저장 되었습니다.

```

*           1. 게임 시작           *
*           2. 점수 화면           *
*           0. 종 료               *
*

```

```

if (p.GetNow_X() == p.GetEnd_X() && p.GetNow_Y() == p.GetEnd_Y()) //종료지점 도착
{
    end = clock();
    int endLine = (end - start) / QLOCKS_PER_SEC;
    int scoreResult = (endLine * 100) + (p.GetHP() * 10) + (p.GetMonsterKill() * 1000); //점수 계산은 (걸린 시간 * 100) + (체력 * 10) + (몬스터 처리수 * 1000) 입니다.

    string name; //플레이어 이름

    p.SetScore(scoreResult);

    cout << "도착했습니다. 완료시간은 : " << endLine << " 초. 입니다." << endl;
    cout << "점수는 " << p.GetScore() << " 점 입니다." << endl;
    cout << "점수 입력을 위해 이름을 입력해 주세요 : ";
    cin >> name;

    scoreFile.open("score.txt"); //파일을 열어준다.

    if (!scoreFile) //만약 파일에 없을경우
    {
        ofstream scoreOut("score.txt");

        scoreOut << name << " " << p.GetScore() << " HARD" << endl;
        scoreOut.close();
        cout << "저장 되었습니다." << endl;
    }
    else //있는경우 뒤에 붙여서 작성
    {
        ofstream scoreOut("score.txt", ios::app);
        scoreOut << name << " " << p.GetScore() << " HARD" << endl;
        scoreOut.close();
        cout << "저장 되었습니다." << endl;
    }

    p.clear(); //Player 클래스 초기화
    break;
}

```

설명

스테이지 게임 종료 후 시간 함수 종료 후 점수 입력을 위해 이름을 받고 이름과 점수, 난이도 등을 텍스트 파일에 저장한다. 저장 시에 파일이 있을 때와 없을 때 예외 처리를 해준다.

점수 출력

1. 점수출력 화면

```

*           스코어를 표시합니다.
*
이름           점수           난이도
seungwook      3700          EASY
seungwook2     3600          EASY
-----
*           1. 게임 시작           *
*           2. 점수 화면           *
*           0. 종 료               *
*

```

2-1. 점수출력 함수

```

//점수 출력 함수
void MenuScore()
{
    system("cls");

    cout << "스코어를 표시합니다.\n\n";

    cout << "이름 점수 난이도\n" << endl;

    vector<PlayerScore> clas;

    PlayerScore ps;

    fstream op;

    string temp_name;
    int temp_score = 0;
    string temp_difficulty;

    string in_line;

    op.open("score.txt", ios::in);
    if (!op.is_open())
    {
        cout << "저장된 기록이 없습니다.\n";
        return;
    }

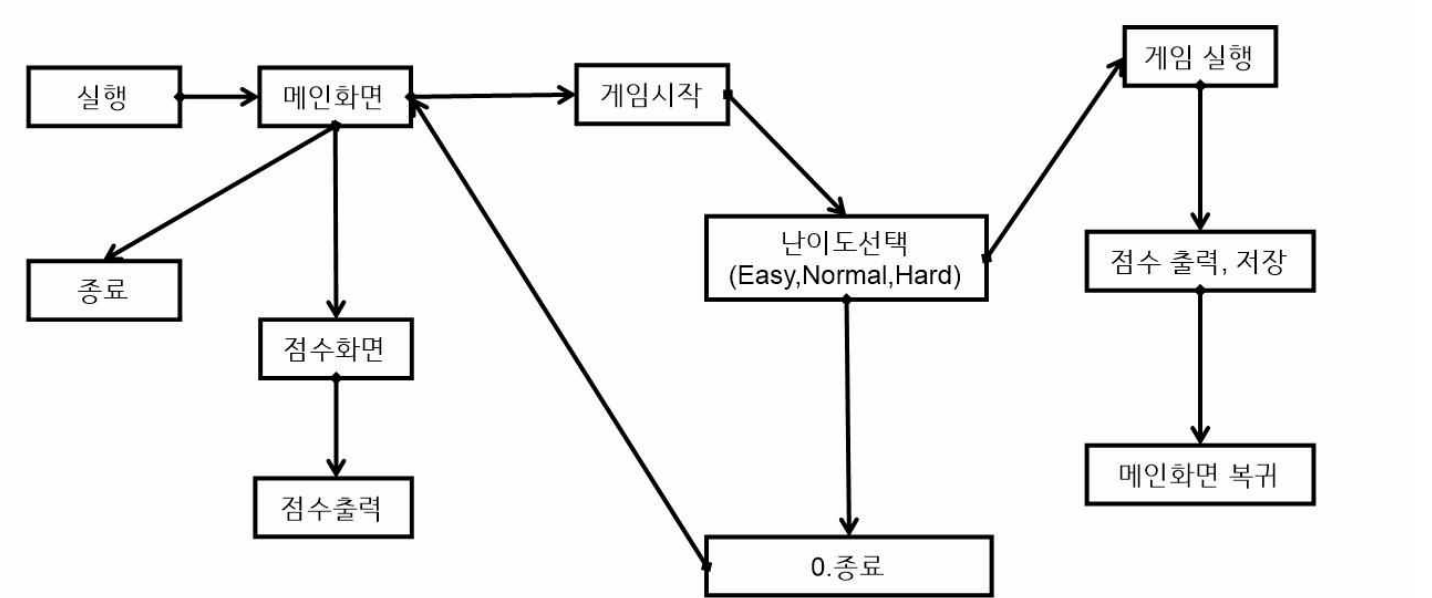
    while (op >> ps.name)
    {
        op >> ps.score;
        op >> ps.difficulty;
        clas.push_back(ps);
    }

    op.close();
}

```

2-2. 점수출력 함수	설명
<pre> for (int i = 0; i < size(clas)-1; i++) { for (int j = i + 1; j < size(clas); j++) { if (clas[i].score < clas[j].score) { temp_name = clas[i].name; temp_score = clas[i].score; temp_difficulty = clas[i].difficulty; clas[i].name = clas[j].name; clas[i].score = clas[j].score; clas[i].difficulty = clas[j].difficulty; clas[j].name = temp_name; clas[j].score = temp_score; clas[j].difficulty = temp_difficulty; } } } op.open("score.txt", ios::out); for (int i = 0; i < size(clas); i++) { op << clas[i].name << "ㅍㅍㅍ" << clas[i].score << "ㅍㅍㅍ" << clas[i].difficulty << endl; } op.close(); ifstream in("score.txt"); while (getline(in, in_line)) { cout << "ㅍㅍㅍ" << in_line << endl; } in.close(); </pre>	<ol style="list-style-type: none"> 1. 텍스트 파일에서 이름, 난이도 , 점수를 가져와 PlayerScore 구조체 배열에 저장한다. 2. 파일이 없을 경우 예외처리를 해준다. 3. 이름, 난이도, 점수를 텍스트 파일에서 불러와 점수를 기준으로 오름차순으로 정렬해준다. 4. 정렬해준 값을 텍스트 파일에 기록해준다. 5. 점수 화면을 출력 후 파일 입출력을 종료해준다.

게임구성도



3번 프로젝트(Unity) - Save The War Ship(슈팅게임)

Save The War Ship		
언어		C#
Assets		warship ,enemy, font, mech, bullet, title
개발 기간		2019.08.28. - 09.03
개발 도구		Unity 2019.2, Visual Studio 2017
개발인원 (3명)	박웅배(팀장)	전체 애니메이션 삽입, 미니 맵 제작, UI제작 및 수정, 게임전체 기획 및 총괄
	정영훈	적 생성 및 AI제작, 기록저장구현, 전체 오브젝트 상태 관리 , 오류수정
	정승욱	오브젝트 스크립트 기본 틀 제작, 씬 전환, 슈팅부분 담당, UI기본 틀 제작, 오류수정

프로젝트 개요

슈팅 생존 게임의 일종. 매 턴(30초)마다 적 기체가 생성 되고 플레이어의 주위를 선회한다. 그 후 랜덤 한 시간이 지나면 플레이어를 향해 적기가 돌진한다. 돌진하는 적을 마우스 회전 및 클릭을 통해 미사일을 을 발사해 적 기체를 격추 시키고 버티는 게임.

프로젝트 목적

유니티 엔진 사용 능력을 키움. 프로그램 동작에 필요한 코드를 적정한 스크립트에 배열하고 컴포넌트 사용을 능숙하게 함.

실행 화면

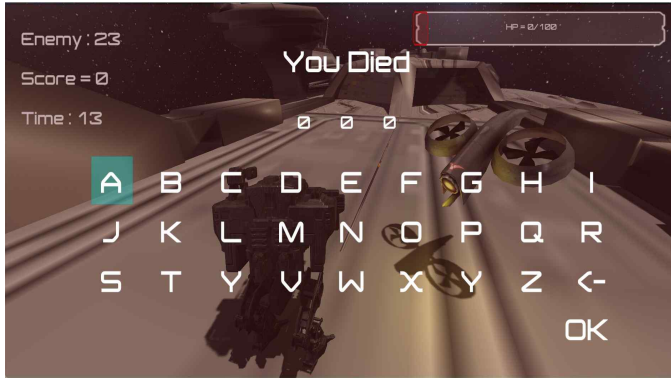
1. 메인화면



2. 게임 화면



3. 게임 종료 화면



4. 점수 화면



코드

1. 총알

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class Bullet : MonoBehaviour
6  {
7      void Start()
8      {
9          //총알이 발사된 후 2초가 지나면 총알 오브젝트 제거
10         Destroy(gameObject, 2f);
11     }
12
13     void Update()
14     {
15         //mechtarget 이름을 가진 오브젝트의 mechstatus의 death 값이 false이면 작동
16         if (GameObject.Find("mechtarget").GetComponent<MechStatus>().death == false)
17         {
18             //프레임마다 오브젝트를 로컬좌표상에서 앞으로 1의 힘만큼 날아가라
19             transform.Translate(Vector3.up * 0.8f);
20         }
21     }
22
23     void OnCollisionEnter(Collision other)
24     {
25         Destroy(gameObject); //물체와 충돌시 총알 제거
26     }
27 }
    
```

2. 슈팅 기본 틀

```

void Update()
{
    // 플레이어 캐릭터가 죽지 않았을때의 if구문
    if (GameObject.Find("mechtarget").GetComponent<MechStatus>().death == false)
    {
        if (Input.GetMouseButton(0))
        {
            expoR.Play(); // 좌측과 우측의 발사효과 이펙트 플레이
            expoL.Play();

            if (flag == true)
            {
                StartCoroutine("Flag"); // 발사시의 사운드 플레이. 총알은 코루틴구문에서 생성한다.
            }
        }
    }
}

IEnumerator Flag()
{
    AudioSource.clip = audioBigCanon;
    AudioSource.Play();
    flag = false;
    yield return new WaitForSeconds(0.1f);
    // 발사되는 총알을 생성하는 코드.
    Instantiate(bullet, FirePosLeft.transform.position, FirePosLeft.transform.rotation);
    Instantiate(bullet, FirePosRight.transform.position, FirePosRight.transform.rotation);
    flag = true;
}
    
```

3-1. UI 기본 틀

```

// 시간을 재는 수식
time -= Time.deltaTime;
if (time <= 0)
{
    time = 30f;
}

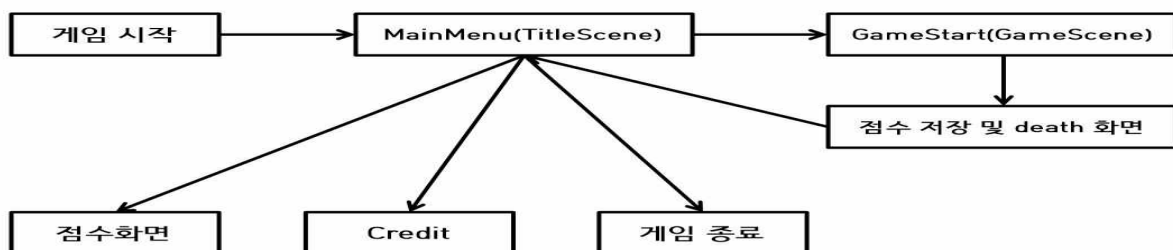
Timer.text = ("Time : " + Mathf.Round(time));
    
```

3-2. UI 기본 틀

```

//slider의 수치를 현재 체력에 빗대어 체력바 표현
hpValue.value = NowHP;
hp.text = ("HP = " + NowHP + "/" + MaxHP);
    
```

게임구성도



4번 프로젝트(Unity + OpenCV Sharp) - AR 3D 모형 세우기

AR 3D 모형 세우기	
언어	C#
개발 도구	Visual Studio 2017, Unity 2019.02
Asset	OpenCV Sharp
개발 기간	2019.10. - 07.06
개발 인원	1명

프로젝트 개요

모바일 카메라로 보이는 평지 빈 공간 위에 3D오브젝트를 띄워 모바일로 가상의 공간을 가늠할 수 있게 만들기 위함

프로젝트 목적

OpenCV의 기본 개념과 AR개발의 시초를 마련하기 위해 프로젝트를 시행 함

코드	
1. 웹 캠 구동	2. 특징 점 추출
<pre>void Awake() { //웹캠 설정 web = new WebCamTexture(320,160,60); red = new Scalar(0, 0, 255); } // Start is called before the first frame update void Start() { //플레이 web.Play(); }</pre>	<pre>void Update() { //웹캠을 Mat 형식으로 변환 image = OpenCvSharp.Unity.TextureToMat(web).Clone(); //gftt방식으로 코너 검출 생성 GFTTDetector gftt = GFTTDetector.Create(2, 0.01,1,3,true,0.01); //코너 검출 시작(gftt) kp = gftt.Detect(image); }</pre>
3. 사각형 예외처리	4. 처리 후 RawImage에 구현
<pre>//검출된 2개의 점으로 사각형 생성 if (kp[0].Pt.X <= kp[1].Pt.X && kp[0].Pt.Y >= kp[1].Pt.Y) { R1 = new Point2f(kp[0].Pt.X, kp[1].Pt.Y); R2 = new Point2f(kp[1].Pt.X, kp[0].Pt.Y); Cv2.Rectangle(image, R1, R2, red); } else if (kp[0].Pt.X <= kp[1].Pt.X && kp[0].Pt.Y <= kp[1].Pt.Y) { R1 = new Point2f(kp[0].Pt.X, kp[0].Pt.Y); R2 = new Point2f(kp[1].Pt.X, kp[1].Pt.Y); Cv2.Rectangle(image, kp[0].Pt, kp[1].Pt, red); } else if (kp[0].Pt.X >= kp[1].Pt.X && kp[0].Pt.Y <= kp[1].Pt.Y) { R1 = new Point2f(kp[1].Pt.X, kp[0].Pt.Y); R2 = new Point2f(kp[0].Pt.X, kp[1].Pt.Y); Cv2.Rectangle(image, R1, R2, red); } else if (kp[0].Pt.X >= kp[1].Pt.X && kp[0].Pt.Y >= kp[1].Pt.Y) { R1 = new Point2f(kp[1].Pt.X, kp[1].Pt.Y); R2 = new Point2f(kp[0].Pt.X, kp[0].Pt.Y); Cv2.Rectangle(image, kp[1].Pt, kp[0].Pt, red); }</pre>	<pre>outTexture = new Texture2D(web.width, web.height); //mat을 텍스처로 변환 outTexture = OpenCvSharp.Unity.MatToTexture(image); //로우 이미지 화면에 띄워줌 raw.texture = outTexture; raw.material.mainTexture = outTexture;</pre>

화면

구동 화면



```
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
[02:40:52] 2 | (x:196 y:62) | (x:236 y:23)
UnityEngine.Debug.Log(Object)
```

시스템 구성도

