Contents

	Language	Lib/SDK	TOOL	-
Adeinc.	C#	EMGU	Visual Studio 2015	정규화 데이터 변환, 각도 검출
	C#	EMGU	Visual Studio 2015	전봇대 특정 객체 검출
Final Project	C#	OpenCVSharp	Visual Studio 2017	Macro
	C#	ARCore, ADMob	Unity, Visual Studio 2017	AR 박스 쌓기
Semi Project	C#		Unity, Visual Studio 2017	3D Simulation
Mimi Project	C++	OpenCV	Visual Studio 2017	차량 번호판 & 번호 검출
	C#		Visual Studio 2017	비행기 슈팅 게임

입사 희망자 : 홍 길 동

[정규화 데이터 변환, 각도 검출_1]

Toolid	TCWFS11											
Sublotid	E0SAA0101A01											
Recipe	₩₩10.150.14.66₩FabVision₩Storage₩DimRecipes₩Recipes-WVSEC-57-P1_pm4-TCWFS12-1.pm4											
Wafer	1											
Slot	1											
Cassette	UA2844											
ScanDate	06/14/2018 13:11:18											
RawFilena	1806141311-969045.wsfv											
Location(r	0	1	2	3	4	5	6	7	8	9	10	11
-150	-	-	-	-	-	-	-	-	-	-	-	-
-149	-	-	-	-	-	-	-	-	-	-	-	-
-148	-	-	-	-	-	-	-	-	-	-	-	-
-147	-	-	-	-	-	-	-	-	-	-	-	-
-146	0.329591	-	-	-	-	0.611305	0.630737	-	-	0.614115	-	0.537522
-145	0.341293	0.418999	0.484233	0.539983	0.585214	0.618559	0.640633	0.651419	0.64612	0.627678	0.598866	0.553532
-144	0.356237	0.429457	0.49482	0.551113	0.597589	0.632627	0.655777	0.666312	0.662888	0.647061	0.617459	0.577064
-143	0.370124	0.442605	0.507595	0.56376	0.610358	0.645837	0.670025	0.681655	0.679599	0.665305	0.637823	0.599178
-142	0.382815	0.454657	0.519198	0.575392	0.622262	0.658128	0.683141	0.695741	0.695395	0.682477	0.65715	0.62002
-141	0.394008	0.465321	0.52944	0.58572	0.633025	0.669366	0.695115	0.708568	0.709961	0.698286	0.674973	0.63966
-140	0.403385	0.47422	0.538128	0.594385	0.642144	0.678982	0.705608	0.719852	0.722743	0.712414	0.690663	0.658043
-139	0.410862	0.481253	0.545022	0.601246	0.649282	0.686664	0.714147	0.729375	0.733151	0.724792	0.704684	0.674679
-138	0.416415	0.486443	0.549971	0.606348	0.65435	0.692414	0.72051	0.737064	0.741683	0.735457	0.717101	0.688821
-137	0.419936	0.489697	0.552934	0.609439	0.657356	0.696179	0.724906	0.742689	0.748443	0.743984	0.727512	0.700763
-136	0.421278	0.490714	0.553687	0.610111	0.65832	0.697751	0.7271	0.745895	0.753161	0.749976	0.73557	0.71058
-135	0.420325	0.489328	0.552018	0.608311	0.657057	0.6968	0.726813	0.746541	0.755585	0.75355	0.741091	0.71813
-134	0.417027	0.485628	0.547991	0.604186	0.653308	0.693275	0.724069	0.744727	0.755432	0.754644	0.743857	0.723172
-133	0.411413	0.479576	0.541653	0.597751	0.646906	0.687446	0.719001	0.74055	0.752616	0.753301	0.744174	0.725567
-132	0.403465	0.471034	0.532957	0.588914	0.638091	0.67942	0.711592	0.733984	0.747158	0.749629	0.742192	0.725514
-131	0.392977	0.460001	0.521891	0.577743	0.627086	0.66897	0.701774	0.725089	0.73945	0.743598	0.737908	0.723059
-130	0.379946	0.446696	0.508485	0.564377	0.613863	0.656202	0.689735	0.713961	0.729374	0.735091	0.731318	0.718277
-129	0.364779	0.431377	0.492907	0.548886	0.598462	0.641295	0.675625	0.70082	0.717085	0.724472	0.722361	0.711327
-128	0.347822	0.414126	0.475393	0.531358	0.58112	0.624479	0.659401	0.685709	0.702872	0.711676	0.711169	0.702157
-127	0.329287	0.395158	0.456232	0.512046	0.562126	0.605968	0.641316	0.668539	0.686858	0.696853	0.697955	0.690665
-126	0.309497	0.374825	0.435722	0.491413	0.541699	0.585786	0.621572	0.649511	0.669186	0.680359	0.682978	0.677244
-125	0.288613	0.353227	0.414058	0.469697	0.519043	0.564104	0.600497	0.629156	0.649886	0.663471	0.666535	0.662205

- 전공정 이후 컷팅된 웨이퍼의 정규화 데이터
- 해당 데이터를 기반으로 웨이퍼 이미지를 생성 후 각도 검출
- 각도 0도 ~ 180도
- 위치 값 -150 ~ +150

[정규화 데이터 변환, 각도 검출_2]

```
    Console.WriteLine("최소값 : " + minValue);
    Console.WriteLine("최대값 : " + maxValue);

    // 최대값 - 최소값 /255 = x.xx~ -x.xx 로 이루어진 실수값의 분포를 255
    double colorField = (maxValue - minValue) / 255;
    double valueCompare = 0;
    int colorValueConvert = 0;

    for (int column = 1; column < BasicConfig.ColumnCount; column++)
    {
        for (int row = 1; row < BasicConfig.RowCount; row++)
        {
            valueCompare = realData[column, row];
            // 최소값을 기준으로 colorField를 더하다 보면, cell에 위치한 값
            for (double oneStep = 0; oneStep <= Math.Abs(minValue) + maxVa
            {
                if (colorField == 0)
                {
                    break;
                }
                if (valueCompare >= minValue + oneStep)
                {
                    realData[column, row] = colorValueConvert;
                }
                colorValueConvert++;
            }
            colorValueConvert = 0;
        }
    }
    minValue = 0;
    maxValue = 0;

    return realData;
}
```

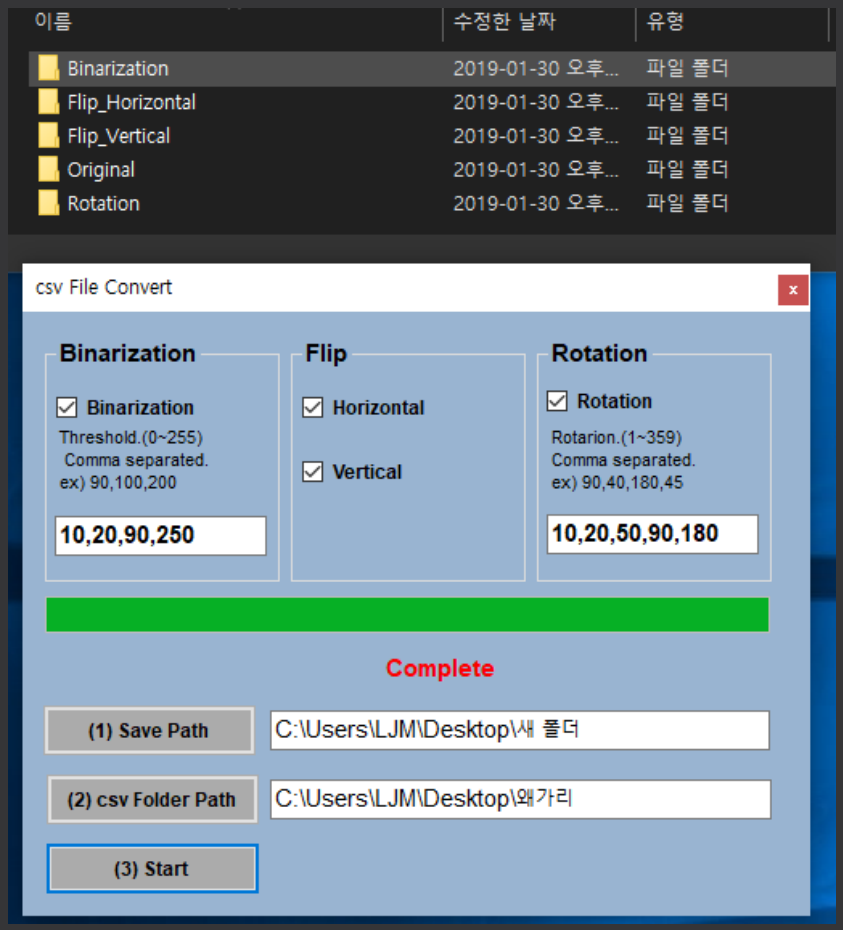
- Csv의 데이터 – 0.XXX ~ -X.XXXX 로 이루어진 실수 분포에서, 최소, 최대값을 찾은 후, 255로 나누어
- 0 ~ 255에 해당하는 GrayColor 값을 찾아낸다.

[정규화 데이터 변환, 각도 검출_3]

```
List<List<double>> standardDeviation = new List<List<double>>();

for (int j = 1; j <= 181; j++)
{
    double variance = 0d;
    double average = 0d;
    double sum = 0;
    standardDeviation.Add(new List<double>(j));
    //145
    //144
    //143
    //
    for (int i = 6; i <= 286; i++)
    {
        sum += realData[i, j];
    }

    average = sum / 281;
    for (int i = 6; i <= 286; i++)
    {
        variance += Math.Pow(realData[i, j] - average, 2);
    }

    standardDeviation[j - 1].Add(realData[0, j]);

    standardDeviation[j - 1].Add(Math.Sqrt(variance / 281));
}

double minDeviation = 0;
double angle = 0;
for (int i = 0; i < standardDeviation.Count; i++)
{
    if (i == 0)
    {
        minDeviation = standardDeviation[i][1];
        angle= standardDeviation[i][0];
    }
   else if(minDeviation>standardDeviation[i][1])
    {
        minDeviation = standardDeviation[i][1];
        angle = standardDeviation[i][0];
    }
}



double realAngle = 180 - angle;

Console.WriteLine("각도 : "+realAngle);
Console.WriteLine("최소 표준편차 : " + minDeviation);
```

- 표준 편차 이용
- 직선상의 Pixel Color 값을 기반으로 180도에 해당하는 180개의 변화량을 측정
- 해당 데이터의 표준 편차량이 가장 적은 것을 기준 각도로 선정.

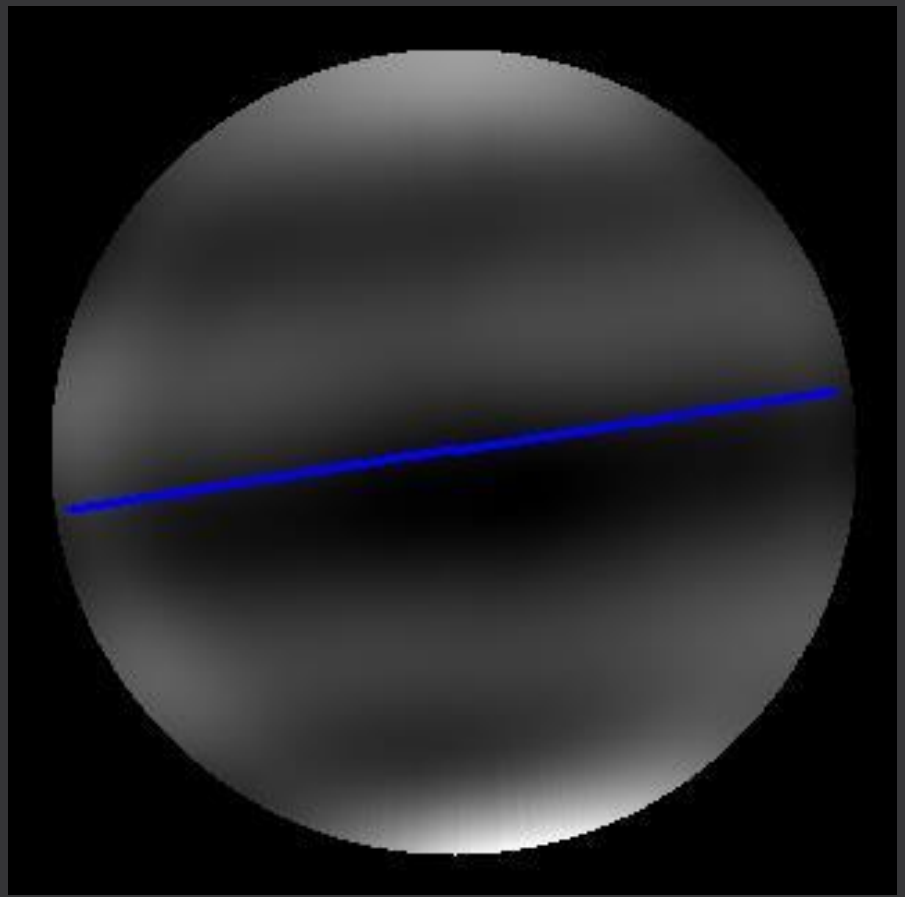
[정규화 데이터 변환, 각도 검출_4]



- Csv 데이터를 기반으로,
  이진화 임계 값,
  상하 좌우 반전,
  회전 값

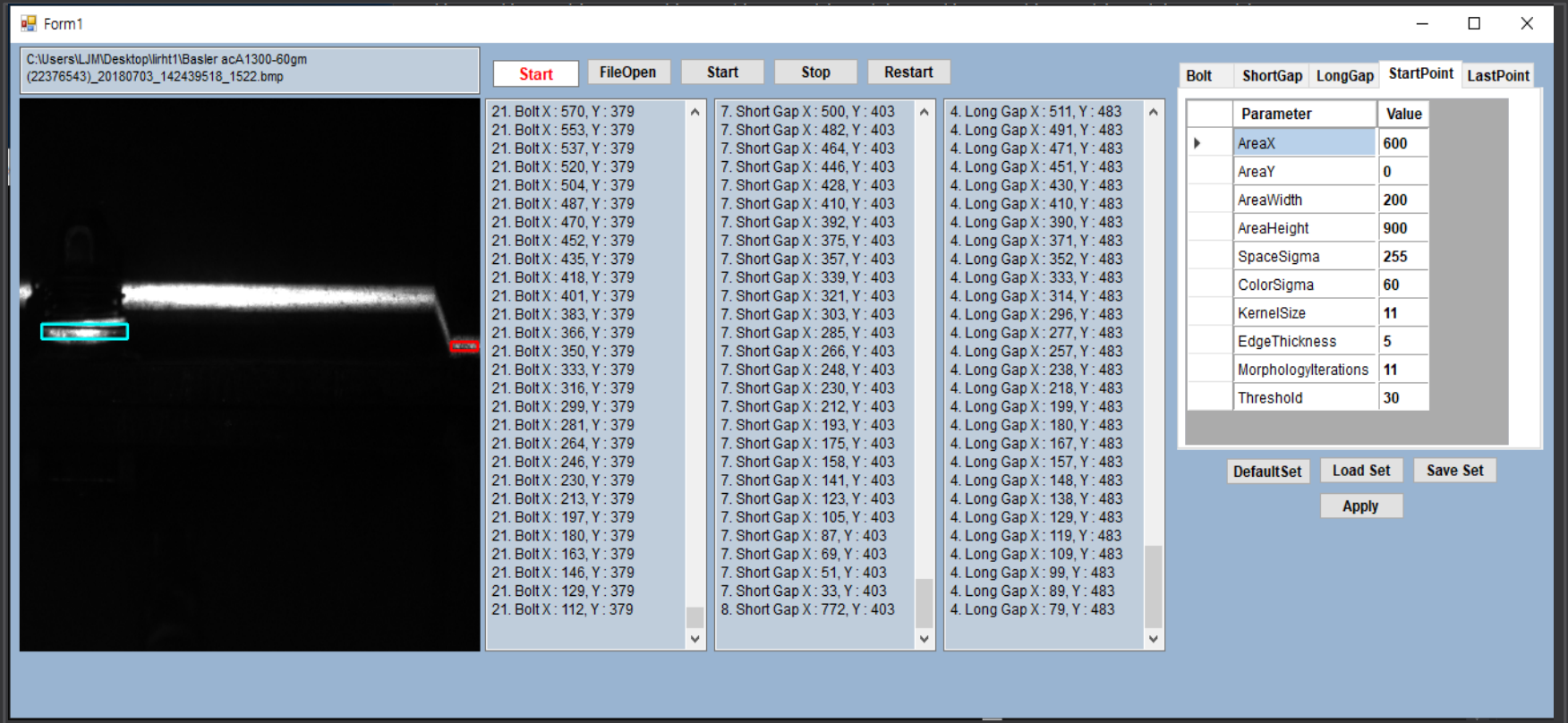
- 입력 값을 기반으로 csv 정규 데이터를
  이미지로 변환.

[정규화 데이터 변환, 각도 검출_5]



- 변환된 이미지와 검출한 각도.
- 컷팅된 웨이퍼의 결에서, 각도를 검출.

[전봇대 특정 객체 검출_1]



- 영상에서 검출된 볼트, 짧은 틈, 긴 틈의 좌표 값을 리턴.

FINAL

AR 블록 쌓기 & C# Macro

[프로젝트 개요]

- **개발 기간 :** 2018-02-05 ~ 2018-04-04

- **투입 인원 :** 2명

- **프로젝트 소개**
  1. Android Platform AR(증강현실) 박스 쌓기 게임
     * AR SDK Computer Vision 중 Markerless Tracking 기능 적용
  2. C# & Winform 앱 플레이어 모바일 게임 자동 Macro
     * OpenCV dll을 이용하여 이미지 유사도 검출 이 후, 검출된 이미지 해당 좌표 값에 Win API 마우스 클릭 이벤트 발생

- **개발 목적**
  - 새로운 기술(AR)의 도입과, C#을 활용한 윈도우OS 환경 프로그램 제어 학습을 위해, Android Platform AR 박스 쌓기 게임과, C#을 이용한 앱 플레이어 모바일 게임 자동 Macro를 개발하게 되었습니다.
  - 2명의 인원으로, 동시에 같은 분야 개발을 진행하였고, 2명의 소스 코드 중 최적의 소스 코드를 게임에 적용하며 개발을 진행 하였습니다.

- **본인 역할**
  - 팀장
  - AR SDK 분석 및 개발 주도.
  - OpenCVSharp dll 분석 및 Macro 개발 주도.

[개발환경]

- AR 박스 쌓기

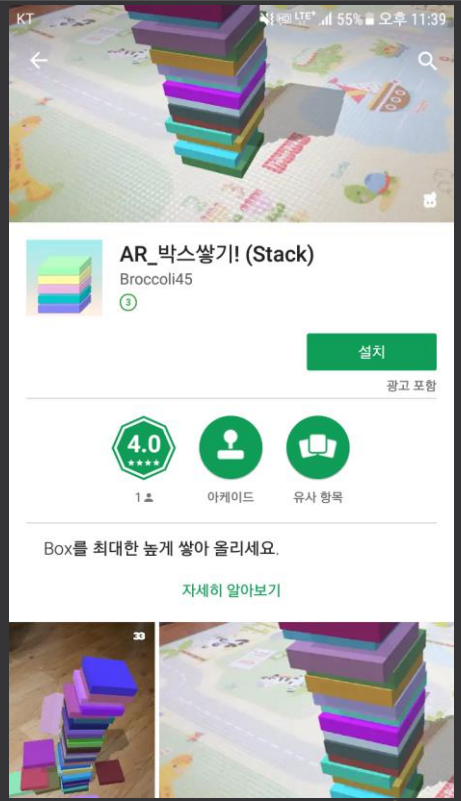
· OS : Windows 10
· 사용 언어 : C#
· 사용 Tool : Unity, Visual Studio 2017, Android Studio
· API, SDK, Lib : Google AR Core, jdk, Admob
· etc : Android 7.0 Version or Higher Device (Galaxy s7)

- C# Macro

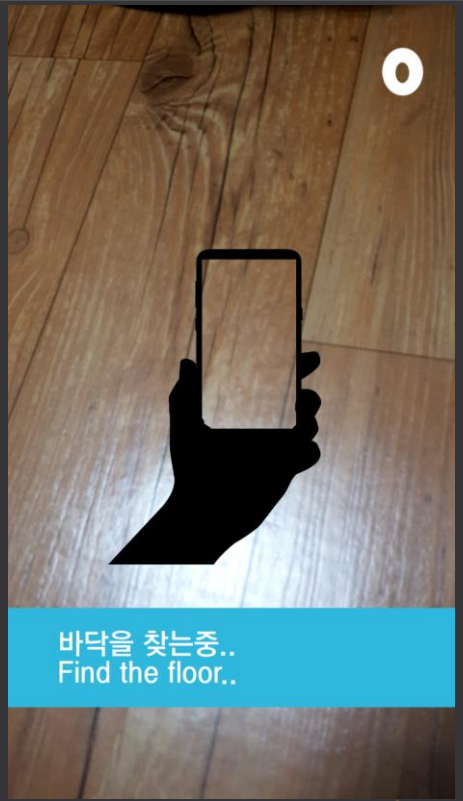
· OS : Windows 10
· 사용 언어 : C#
· 사용 Tool : Visual Studio 2017
· API, SDK, Lib : OpenCVSharp
· etc : Android App Player(MoMo AppPlayer)

1. AR 박스 쌓기

[AR 박스 쌓기 – 설명]



다운로드



실행 후 바닥 인식

게임 Play

[AR 박스 쌓기 – Script_1]

ARController.cs

28_ARCoreTest_TestTest | GoogleARCore.HelloAR.ARController | Update()

```
119             TrackableHitFlags raycastFilter = TrackableHitFlags.PlaneWithinPolygon |
120                 TrackableHitFlags.FeaturePointWithSurfaceNormal;
121
122             if (startCheck == true)
123             {
124                 if (Frame.Raycast(touch.position.x, touch.position.y, raycastFilter, out hit))
125                 {
126                     VisualizePlanescnt = 1;
127                     floorObject = Instantiate(StartFloor, hit.Pose.position, hit.Pose.rotation);
128                     floorObject.transform.rotation = Quaternion.identity;
129                     showTouchFloorUI = false;
130
131                     boxrColor = rndColor.Next(0, 256);
132                     boxgColor = rndColor.Next(0, 256);
133                     boxbColor = rndColor.Next(0, 256);
134                     fixBox = GameObject.FindWithTag("FixedBox");
135                     fixBox.GetComponent<MeshRenderer>().material.color = new Color(boxrColor * 0.00392156f, b
136
137
138                     anchor = hit.Trackable.CreateAnchor(hit.Pose);
139                     if ((hit.Flags & TrackableHitFlags.PlaneWithinPolygon) != TrackableHitFlags.None)
140                     {
141                         Vector3 cameraPositionSameY = FirstPersonCamera.transform.position;
142                         cameraPositionSameY.y = hit.Pose.position.y;
143                     }
144
145                     floorObject.transform.parent = anchor.transform;
146                     startCheck = false;
147                     VisualizePlanes = false;
148
149                 }
150             }
151
152             TouchFloorUI.SetActive(showTouchFloorUI);
```

바닥 인식 후 터치, 초기 1회 floorObject 생성(바닥 위에 띄워질 첫 번째 박스)

R/G/B 랜덤 컬러 선언,

Anchor를 부모 오브젝트로 선언하여, AR환경에서 처음 생성된 첫 번째 박스의 위치가 고정되도록.

[AR 박스 쌓기 – Script_2]

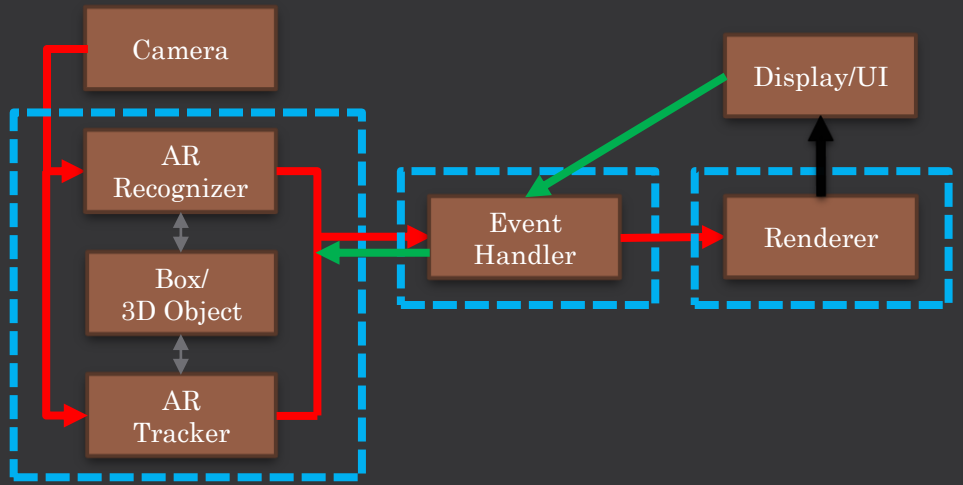
CubeCoroutine.cs | ARController.cs

_ARCoreTest_TestTest | MoveCubeCoroutine | GameStart()

```
77  void Start()
78  {
79      BoxPrefabs = (GameObject)Resources.Load("moveCube");
80  }
81
82  void Update()
83  {
84      if (this.GetComponent<GoogleARCore.HelloAR.ARController>().MovePhoneFindFloorCnt == 1)
85      {
86          StartCoroutine(PhoneZoomInOut());
87      }
88      if (this.GetComponent<GoogleARCore.HelloAR.ARController>().MovePhoneFindFloorCnt == 0 && Cnt == 0)
89      {
90          StopCoroutine(PhoneZoomInOut());
91          sw.Stop();
92          StartCoroutine(GameStart());
93          Cnt = 1;
94      }
95  }
96
97  IEnumerator GameStart()
98  {
99      while (true)
100     {
101         yield return null;
102         if (this.GetComponent<GoogleARCore.HelloAR.ARController>().startCheck == false)
103         {
104             if (firstCnt == true)
105             {
106                 startBeginningX = this.GetComponent<GoogleARCore.HelloAR.ARController>().floorObject.trar
107                 startBeginningY = this.GetComponent<GoogleARCore.HelloAR.ARController>().floorObject.trar
108                 startBeginningZ = this.GetComponent<GoogleARCore.HelloAR.ARController>().floorObject.trar
109                 beginningX = this.GetComponent<GoogleARCore.HelloAR.ARController>().floorObject.transform
110                 beginningZ = this.GetComponent<GoogleARCore.HelloAR.ARController>().floorObject.transform
```

리소스의 오브젝트를 prefab으로 생성 후 첫 번째 박스가 생성되면, 해당 오브젝트에 있는 MovePhoneFindFloorCnt index값을 참조하여, GameStart Coroutine 진행과 동시에 초기 생성박스 위치좌표를 받아오며, X/Z좌표 왕복 운동을 하는 박스를 생성.

[AR 박스 쌓기-아키텍처]



Camera : 영상입력

AR Rocognizer : Camera로부터 입력 받은 영상에서 특정 객체(바닥)을 인식
*개발에 사용된 Tracking 기술은 Markerless기반 이므로, 특이점의 밀집도 분석을 통해 바닥인지 아닌지를 판단.

AR Tracker : 카메라를 통해 입력 받는 영상 안에서 3D Object가 계속 움직이는 경우 해당 Object를 지속적으로 추적/인식하는 역할.

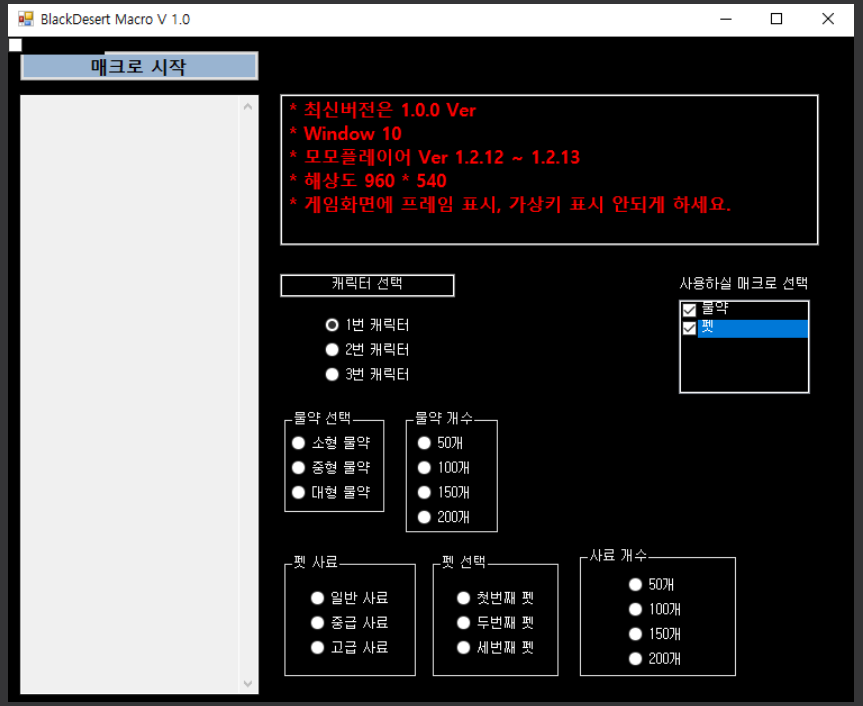
Display/UI : 최종적으로 합성된 영상을 사용자에게 보여주는 부분, 스마트폰의 디스플레이에 해당.

Renderer : Event Handler에서 등록 및 결정된 정보 및 합성 정보를 이용하여 실질적인 영상 합성을 진행하며, 동시에 Display로 최종 합성 결과(게임 영상)를 전달.

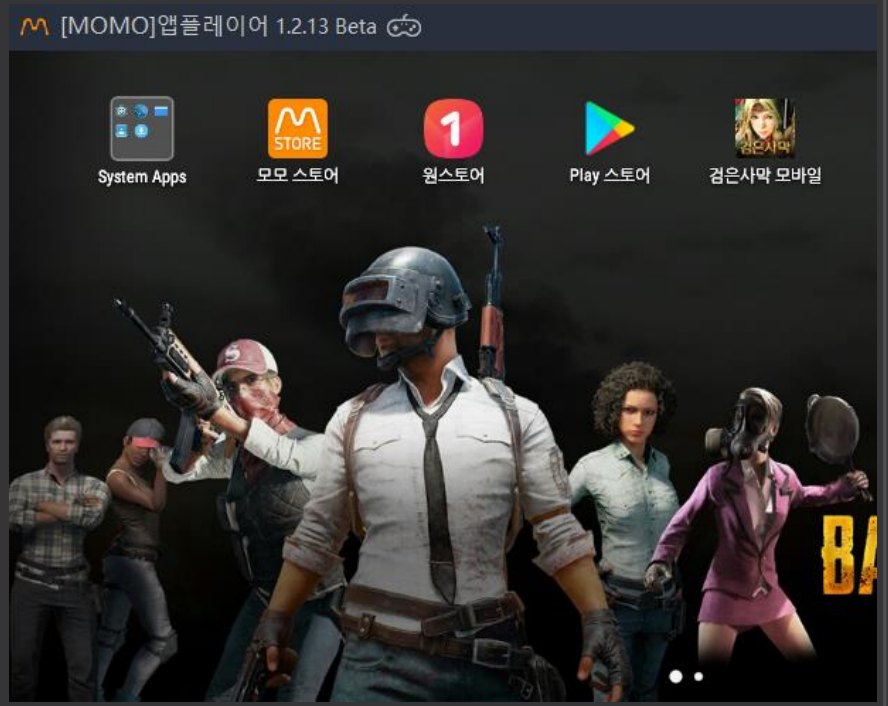
Event Handler : 카메라 영상과 3D Object의 합성 정보, 디바이스 입력 값을 생성하고 전달.

2. C# Macro

[C# Macro – 설명_1]

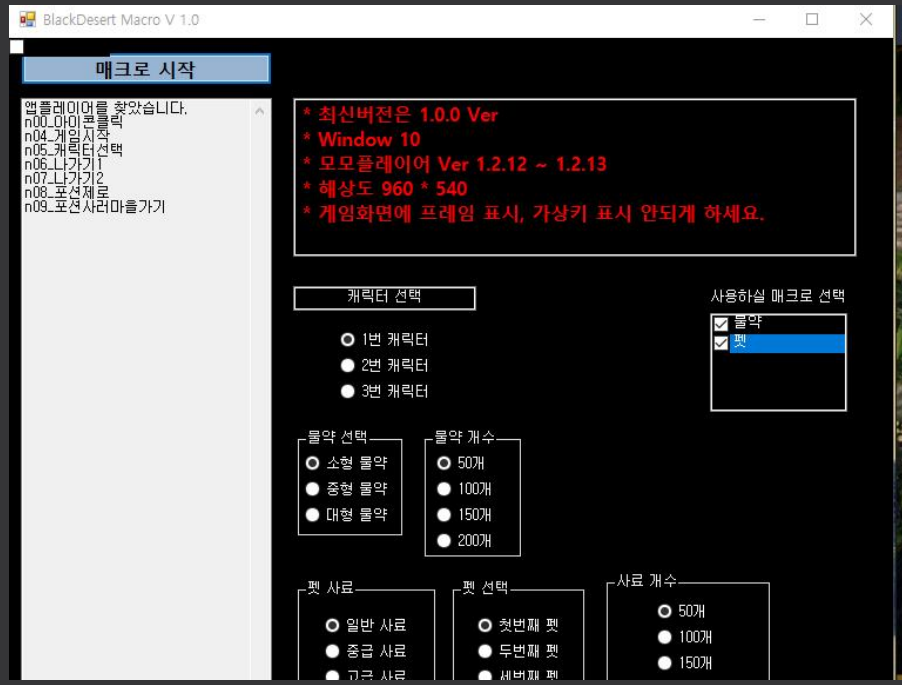


매크로 초기화면



앱 플레이어 초기화면

[C# Macro – 설명_2]



1. 매크로 초기화면 설정 값을 기준으로

1. 게임 자동실행
2. 캐릭터 선택
3. 자동 반복 퀘스트 진행

1. 물약 및 펫 사료 0개 검출 시 상점이동
2. 구매 후 사냥 터로 복귀

[C# Macro – Sources_1]

```
using OpenCvSharp;

namespace 02_Test
{
    public partial class Form1 : Form
    {
        //앱플레이어 찾기, User32 FindWindow API 사용을 위해 DllImport 선언
        [System.Runtime.InteropServices.DllImport("User32", EntryPoint = "FindWindow")]
        private static extern IntPtr FindWindow(string lpClassName, string lpWindowName);

        //앱플레이어 캡처, User32 PrintWindow API 사용을 위해 DllImport 선언
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        internal static extern bool PrintWindow(IntPtr hWnd, IntPtr hdcBlt, int nFlags);

        //마우스 클릭 이벤트를 발생시키기 위해 SendMessage, FindWindowEx API 사용.
        [System.Runtime.InteropServices.DllImport("user32.dll")]
        public static extern int SendMessage(IntPtr hwnd, int wMsg, int wParam, IntPtr lParam);
        [System.Runtime.InteropServices.DllImport("User32.dll")]
        public static extern IntPtr FindWindowEx(IntPtr Parent, IntPtr Child, string lpszClass, string lpszW

        public const int WM_LBUTTONDOWN = 0x201;
        public const int WM_LBUTTONUP = 0x202;

        private bool one = true;
        private bool mainGamestart = false;
        private bool GameStartSettingFinish = true;
        private bool startButton = true;
        private bool Repetition = false;   //포션 구매후 자동사냥

        int i = 0;

        double maxval;

        String AppPlayerName = "[MOMO]앱플레이어-2";

        Bitmap[] CheckImage = new Bitmap[33];
```

OpenCvSharp 및 핵심 API 사용 선언부.

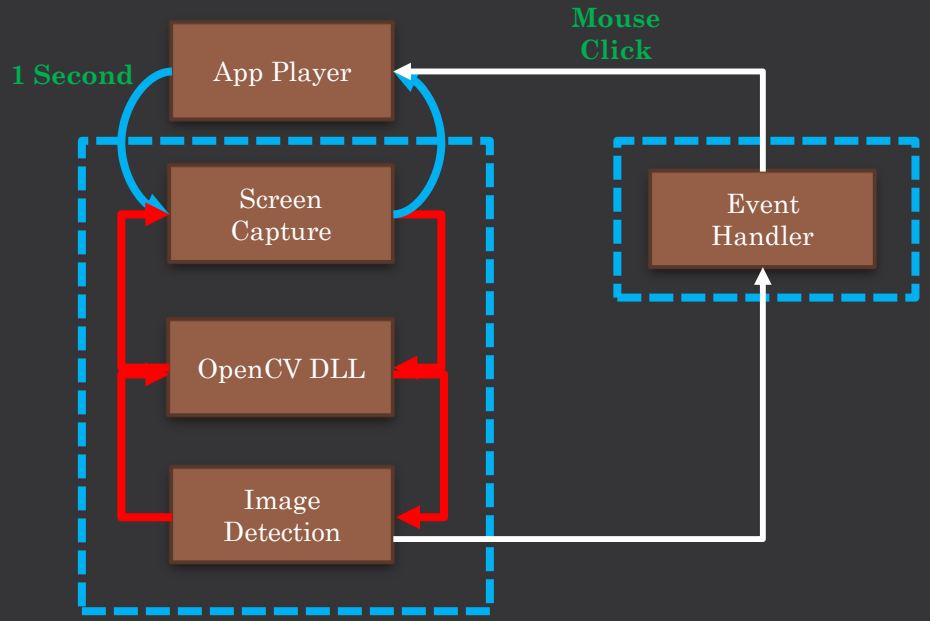
이 후, Bitmap 에 검출 해야할 이미지를 순차적으로 삽입.

[C# Macro – Sources_2]

```
//searchIMG에 스크린 이미지와 검은사막 아이콘 이미지가 들어감
public void SearchIMG(Bitmap screen_img, Bitmap search_img)
{
    // 스크린 이미지 선언
    using (Mat ScreenMat = OpenCvSharp.Extensions.BitmapConverter.ToMat(screen_img))
    // 찾을 이미지 선언
    using (Mat FindMat = OpenCvSharp.Extensions.BitmapConverter.ToMat(search_img))

    //스크린 이미지에서 FindMat이미지를 찾아라
    using (Mat res = ScreenMat.MatchTemplate(FindMat, TemplateMatchModes.CCoeffNormed))
    {
        // 찾은 이미지의 유사도를 담을 더블형 최대 최소값을 선언.
        double minval;
        maxval = 0;
        // 찾은 이미지의 위치를 담을 포인트형을 선언.
        OpenCvSharp.Point minloc, maxloc;
        // 찾은 이미지의 유사도 및 위치 값을 받는다.
        Cv2.MinMaxLoc(res, out minval, out maxval, out minloc, out maxloc);
        Debug.WriteLine("이미지의 유사도" + i + " 번째: " + maxval);

        //검은 사막 아이콘을 찾았을경우 클릭 이벤트 발생, 게임 켜짐
        if (maxval >= 0.8)...
    }
}

//x,y 값을 전달해주면 클릭이벤트를 발생합니다.
public void InClick(int x, int y)
{
    //클릭이벤트를 발생시킬 플레이어를 찾습니다.
    IntPtr findwindow = FindWindow(null, AppPlayerName);
    if (findwindow != IntPtr.Zero)
    {
        //플레이어를 찾았을 경우 클릭이벤트를 발생시킬 핸들을 가져옵니다.
        IntPtr hwnd_child = FindWindowEx(findwindow, IntPtr.Zero, "RenderWindow", "TheRender");
        IntPtr lparam = new IntPtr(x | (y << 16));

        //플레이어 핸들에 클릭 이벤트를 전달합니다.
        SendMessage(hwnd_child, WM_LBUTTONDOWN, 1, lparam);
        SendMessage(hwnd_child, WM_LBUTTONUP, 0, lparam);
    }
}
```

찾을 이미지가 현재 스크린 이미지 내에서 유사도 80%이상이면 InClick 호출과 동시에 x, y인자 값 전달. 해당 위치 클릭이벤트 발생.

[C# Macro – 아키텍처]



App Player : 안드로이드 앱플레이어

Screen Capture : System Drawing Api를 이용하여 앱플레이어 전체 화면을 1초에 한번씩 캡처.

Image Detection : Event Handler를 통해 전달 받은 검출하고자 하는 이미지를 Screen 내에서 검출.

Event Handler : Image Detection을 통해 검출된 이미지의 좌표 값을 기준으로, 마우스 클릭 이벤트를 발생.

Unity–
Crazy Rhino

[프로젝트 개요]

- **투입 인원 :** 2명

- **프로젝트 소개**
  1. 월드맵을 돌아다니며, 각종 건물 및 야생동물을 파괴시키는 게임
     * Steam Goat Simulation
  2. Terrain 지형 구현
  3. 플레이어 Rhino와 건물 충돌 시 파괴 구현
  4. 도망 다니는 야생동물 AI 구현
  5. Unity(C#)

- **본인 역할**
  - 팀원
  - 기획
  - 지형 구현, 건물 파괴, 야생동물 AI 구현

[Crazy Rhino 설명_1]

Steam, Goat Simulation
참고하여 개발 진행.

[Crazy Rhino 설명_2]

Terrain Paint Texture 및 Paint Tree를 이용하여, 맵 디자인

[Crazy Rhino 설명_3]

플레이어 Rhino와 건물 충돌 시 파괴 구현

[Crazy Rhino 설명_4]

```
5   public class csBarrelsDestroy0 : MonoBehaviour
6   {
7       GameObject building;
8       GameObject[] BuildingTop;
9
10      float xpos;
11
12      Rigidbody rigid;
13      Rigidbody toprigid;
14
15      void Start()
16      {
17          rigid = GetComponent<Rigidbody>();
18          building = GameObject.FindWithTag("Barrels");
19          xpos = building.GetComponent<Transform>().position.x;
20      }
21      private void OnCollisionEnter(Collision collision)
22      {
23          BuildingTop = GameObject.FindGameObjectsWithTag("BarrelsTop");
24
25          foreach (GameObject t in BuildingTop)
26          {
27              toprigid = t.GetComponent<Rigidbody>();
28              if (collision.gameObject.tag == "Player")
29              {
30                  rigid.constraints = RigidbodyConstraints.None;
31                  if (xpos != building.transform.position.x)
32                  {
33                      toprigid.constraints = RigidbodyConstraints.None;
34                  }
35              }
36          }
37      }
```

- 각 건물의 벽, 바닥, 천장은 rigidbody → constaints → position/rotation 모두 FreezeALL 상태이다. 플레이어 Rhino와 Collision이 발생하면, RigidbodyConstraints.None으로 값을 변환. 건물이 고정 되어있다가, 부셔진다.

[Crazy Rhino 설명_5]


```
private void Update()
{

    float dist = Vector3.Distance(playerTransform.position, _transform.position);



    if (dist <= 40f)
    {

        pos = transform.position;

        direction = (pos - playerTransform.position).normalized;
        Quaternion rot = Quaternion.LookRotation(direction);
        Vector3 vAngle = rot.eulerAngles;
        transform.eulerAngles = new Vector3(0f, vAngle.y, 0f);


        anim.SetInteger("moveFlag", 2);
        transform.position = pos + direction * Time.deltaTime * 20f;



    }
    else
    {
        if (WalkIdle == 0)
        {
            if (cnt == 0)
            {
                Invoke("Idle2", randwait);
            }
            else
                Invoke("Idle2", 3f);

        }
        else if (WalkIdle == 1)
        {

            Invoke("Walk2", 3f);

        }
    }

        prepos = playerTransform.position;
}
```

- 플레이어 Rhino와 스크립트가 삽입된, 해당 오브젝트의 거리를 Vector3.Distance 를 이용하여 float 값으로 리턴 받은 후, 해당 거리가 40보다 작아지면, 플레이어 Rhino가 달려오는 방향으로 해당 동물이 달려서 도망간다.

MINI

1. C# Winform 슈팅 게임
2. OpenCV 차량 번호판 인식

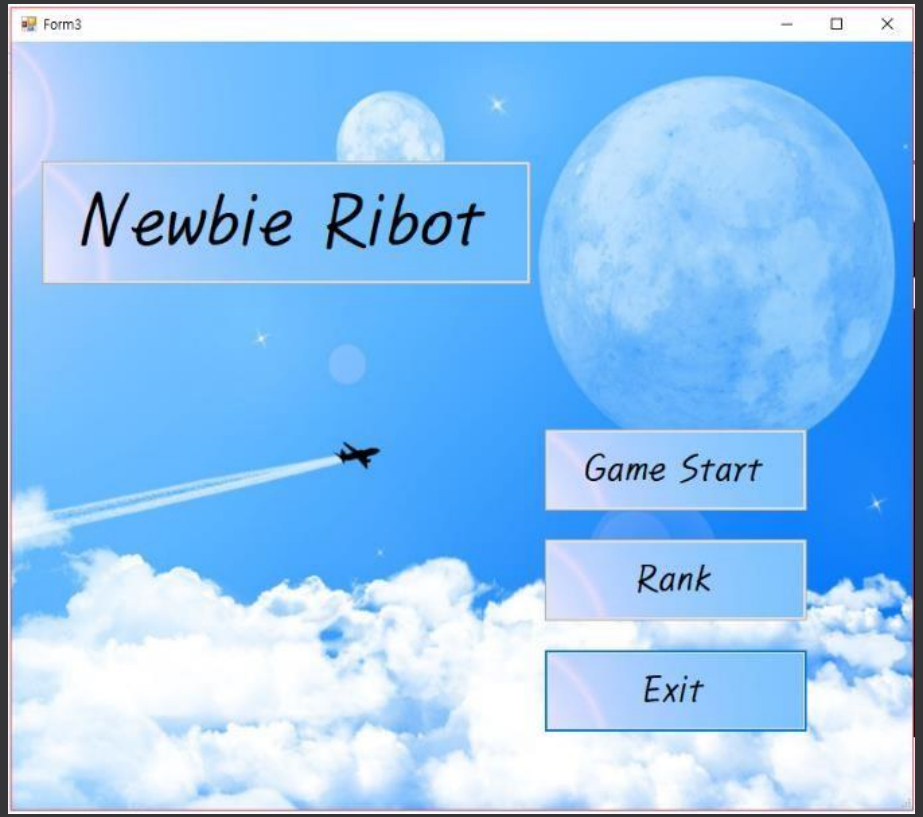
[C# Winform 슈팅 게임 개요]

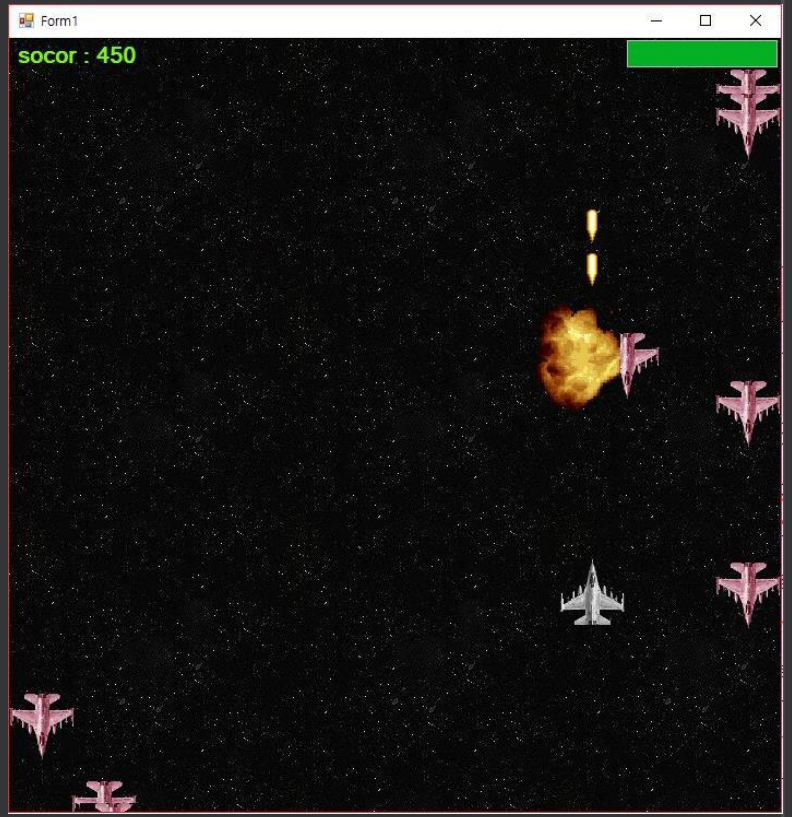
· 투입 인원 : 1명

· C# Winform 슈팅 게임 소개

- 키보드 방향 값 으로 움직이는 비행기 슈팅게임
- Picturebox 로 비행기 및 배경 제작.
- Picturebox 2개를 이용하여, 백그라운드 스크롤 효과,
- 적 비행기 랜덤 한 위치 생성,
- 미사일 picturebox와 적 비행기 picturebox x, y좌표가 일치되면, 적 비행기 폭파 picturebox(Gif) 출력
- progressBar 이용하여 체력 바 생성
- Visual Studio 2017 (C#)

[C# Winform 슈팅 게임 이미지_1]





[OpenCV 번호판 인식 개요]

· 투입 인원 : 1명

· **번호판 인식 소개**

- 차량 사진 內 번호판 검출 이후 콘솔에 숫자 출력.
- Canny 엣지검출 → GaussianBlur 노이즈 제거 이후, 검출 영역에 레이블링 작업.
- 실제 자동차 번호판의 가로/세로 비율과
  번호판 숫자의 가로세로 비율은 고정비율이므로,
  레이블링 작업 이후 가로/세로 비율 값을 기준으로, 번호판 검출 → 숫자 검출
  준비 놓은 0 – 9 까지의 2채널 bmp 파일과 비교하여, 일치하는 값을 콘솔에 출력
- Visual Studio 2017 (C++, OpenCV)

[OpenCV 번호판 인식 이미지_1]