3D Bot Project

개발 기간	2016.01.04 ~ 2016.02.26
개발 스펙	Unity [C#] JsonFx Asset [iTween / SimpleDecalSystem / HighlightingSystem]
개발 내역	클라이언트 및 맵에디터 프로토타입 구현
요 약	홍익대학교 연구보조생 및 Makerslab 산학협력생을 하면서 진행했던 프로젝트입니다. 버전1 프로젝트에서는 프로토타입 및 버그 수정 작업을 하였고 버전2 개발에서는 프로그램 설계 작업에 참여 하였습니다.
성과 / 결과	이화여대 부속 초등학교에서 테스트 진행 웹페이지 출시(링크)
소스 코드	https://github.com/InwonHwang/3D_Bot_Project
게임 영상	https://www.youtube.com/watch?v=R8gkMF1Hnw8

커스터마이징

커스터마이징 기능으로는 로봇의 부품변경, 색상변경, 스케일변경, 스티커 부착이 있습니다. 파츠 변경 기능은 UI 오브젝트의 이름으로 같은 이름의 부품을 찾아서 변경하는 방식을 사용하였습니다.

```
//Parts
for (int i = 0; i < nameOfParts.Length; i++)
{
    var scrollable = GameObjectAgent.Instance.addComponent<GUIScrollable>(GUIAgent.Instance.GuiObjects["UI/2_C
    scrollable.init(nameOfParts[i].ToLower());
    scrollable.action = () =>
    {
        if (!highlightedObject) return;
        if (highlightedObject.name.Contains("Sticker")) return;
        if (highlightedObject.parent.parent.name.CompareTo(scrollable.MinButtonName) == 0) return;

        var newPart = scriptOfBot.activate(scrollable.MinButtonName);

        if (newPart == null) return;

        scriptOfBot.matchJoint();
        HighlightAgent.Instance.constantOnImmediate(newPart.name, Color.cyan);
        highlightedObject = newPart.transform.GetChild(1);
    };
}
```

<부품변경코드>

컬러 변경 또한 UI 오브젝트의 이름으로 부품의 머테리얼 컬러를 변경하는 방식으로 구현하였고 스케일 경우는 UI가 회전한 각도에 비례하게 스케일을 변경하는 방식으로 구현하였습니다.

```
// Scale
EventTriggerAgent.Instance.addEvent("UI", "2_Customizing/Frame/ScrollPanel/Scale/Center/Toggle", EventTriggerType.Drag, () =>
{
    if (!highlightedObject) return;

    float x;
    float y;

    RaycastHit hit;
    Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
    float ang = 0;
    if (Physics.Raycast(ray, out hit, 100f))
    {
        x = transform.InverseTransformPoint(hit.point).x;

        if (x < GUIAgent.Instance.GuiObjects["UI/2_Customizing/Frame/ScrollPanel/Scale/Left"].transform.localPosition.x)
            x = GUIAgent.Instance.GuiObjects["UI/2_Customizing/Frame/ScrollPanel/Scale/Left"].transform.localPosition.x;
        else if (x > GUIAgent.Instance.GuiObjects["UI/2_Customizing/Frame/ScrollPanel/Scale/Right"].transform.localPosition.x)
            x = GUIAgent.Instance.GuiObjects["UI/2_Customizing/Frame/ScrollPanel/Scale/Right"].transform.localPosition.x;

        y = Mathf.Sqrt(332 * 332 - x * x);

        ang = Mathf.Atan2(y, x) / Mathf.PI * 180 - 90;
        GUIAgent.Instance.GuiObjects["UI/2_Customizing/Frame/ScrollPanel/Scale/Center"].transform.rotation = Quaternion.Euler(0, 0, ang);
        highlightedObject.parent.localScale = new Vector3(13 - ang / 90, 13 - ang / 90 , 13 - ang / 90);

        scriptOfBot.matchJoint();
    }
```

<스케일변경코드>

스티커 부착기능은 마우스 드래그시 스티커 UI가 마우스 포인터를 따라가도록 하였고 드래그 종료 시 SimpleDecalSystem 에셋에 동적으로 메쉬를 만드는 기능을 사용해서 스티커를 부착하였습니다.

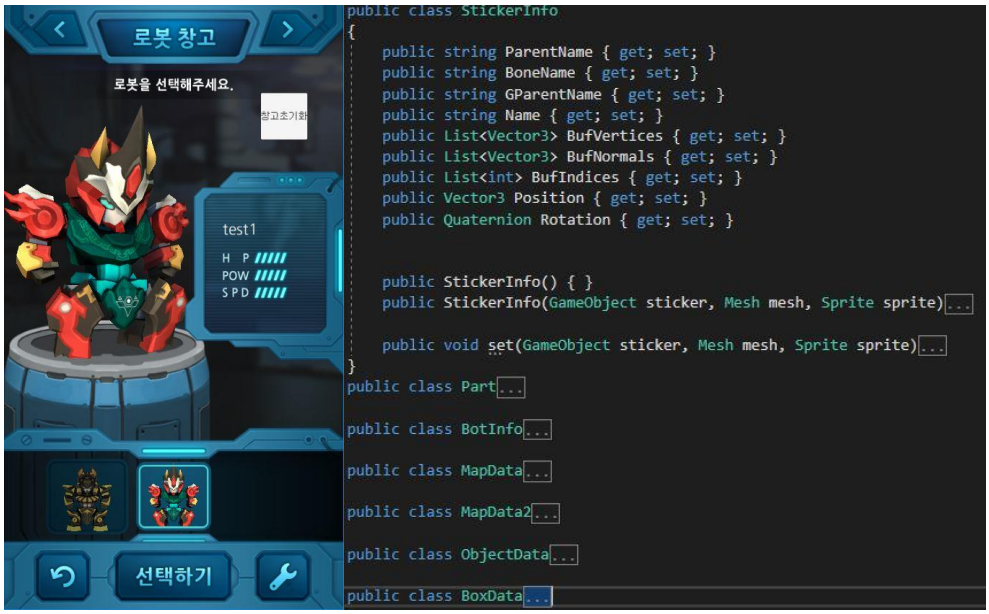
```
EventTriggerAgent.Instance.addEvent(image, EventTriggerType.Drag, () =>
{
    if (image.GetComponent<UnityEngine.UI.Image>().sprite.name.CompareTo(scrollableOfSticker.MinButtonName) != 0) return;

    RaycastHit rh;
    Ray ray = uiCamera.ScreenPointToRay(Input.mousePosition);

    if (Physics.Raycast(ray, out rh, Mathf.Infinity))
        dummySticker.transform.localPosition = transform.InverseTransformPoint(rh.point) + new Vector3(0, 600, 0);
});
```

```
EventTriggerAgent.Instance.addEvent(image, EventTriggerType.EndDrag, () =>
{
    if (image.GetComponent<UnityEngine.UI.Image>().sprite.name.CompareTo(scrollableOfSticker.MinButtonName) != 0) return;

    stickerParent.GetChild(0).transform.position = new Vector3(1000, 1000, 1000);

    if (!stickerParent || stickerParent.childCount == 1) return;

    var sticker = stickerParent.GetChild(1);

    Ray ray = stickerCamera.ScreenPointToRay(Input.mousePosition);
    RaycastHit rh;

    if (Physics.Raycast(ray, out rh, Mathf.Infinity) &&
        rh.transform.gameObject.layer == LayerMask.NameToLayer("Default") &&
        !rh.transform.name.Contains("Sticker"))
    {

        var decal = sticker.GetComponent<DecalSystem.Decal>();
        if (decal)
        {
            sticker.transform.position = rh.point;
            sticker.transform.SetParent(rh.transform);
            DecalSystem.DecalBuilder.BuildDecal(decal);
```

<스티커부착코드>

창고

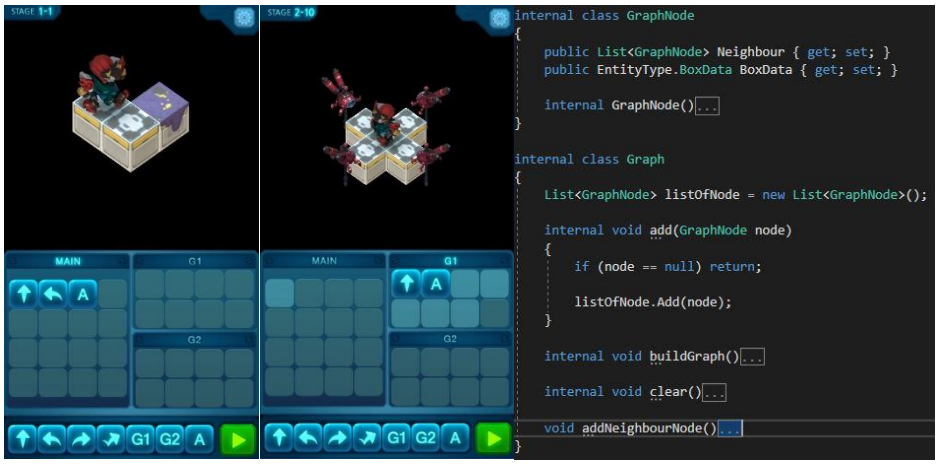
커스터마이징이 완료된 로봇은 자동으로 저장되고 창고내에서 저장된 로봇들을 선택하여 게임을 플레이 할 수 있습니다. 각 데이터는 JsonFx를 이용하여 Json 포맷으로 저장하였습니다.




```
public class StickerInfo
{
    public string ParentName { get; set; }
    public string BoneName { get; set; }
    public string GParentName { get; set; }
    public string Name { get; set; }
    public List<Vector3> BufVertices { get; set; }
    public List<Vector3> BufNormals { get; set; }
    public List<int> BufIndices { get; set; }
    public Vector3 Position { get; set; }
    public Quaternion Rotation { get; set; }


    public StickerInfo() { }
    public StickerInfo(GameObject sticker, Mesh mesh, Sprite sprite)...

    public void set(GameObject sticker, Mesh mesh, Sprite sprite)...
}
public class Part...

public class BotInfo...

public class MapData...

public class MapData2...

public class ObjectData...

public class BoxData...
```

퍼즐게임

커스터마이징으로 완료된 로봇 또는 창고에서 선택한 로봇으로 퍼즐게임을 플레이 할 수 있습니다. 게임 방식으로는 로봇으로 맵 내에 오염된 타일을 정화하거나 적 로봇을 쓰러뜨리면 클리어 됩니다. 퍼즐 로봇 주변물체를 판정하기 위해 그래프 자료구조를 사용했습니다. 아래 명령어를 클릭하면 활성화된 패널(Main, G1, G2) 에 명령어가 추가됩니다. 플레이 버튼을 누르면 명령어에 따라 로봇이 행동합니다.




```
internal class GraphNode
{
    public List<GraphNode> Neighbour { get; set; }
    public EntityType.BoxData BoxData { get; set; }

    internal GraphNode()...
}

internal class Graph
{
    List<GraphNode> listOfNode = new List<GraphNode>();

    internal void add(GraphNode node)
    {
        if (node == null) return;

        listOfNode.Add(node);
    }

    internal void buildGraph()...

    internal void clear()...

    void addNeighbourNode()...
}
```

맵 에디터로 저장된 Json 파일을 레벨클래스에서 읽어 들여서 맵을 구성하는 오브젝트들을 활성화 하고 그래프를 만들어냅니다.

```
void setMapData(string fileName)
{
    if (fileName == null) return;

    string data = StreamAgent.Instance.readFile("Level", fileName);
    mapData = Converter.Deserialize<EntityType.MapData2>(data);
    GUIAgent.Instance.GuiObjects["UI/3_Ingame/Bottom/Button/Play"].GetComponent<UnityEngine.UI.Button>().enabled = true;
}
```

```
void activateObjectsByMapData()
{
    clear();

    var parent = GameObjectAgent.Instance.findChild("Level", "Objects/Tile_Basic");
    var newParent = GameObjectAgent.Instance.findChild("Level", "ToBeActivated/Boxes");

    for (int i = 0; i < mapData.ListOfBoxData.Count; i++)
    {
        var box = findParent(mapData.ListOfBoxData[i].Name).transform.GetChild(0);

        if(box.name.Contains("Plague"))
            box.transform.GetChild(0).gameObject.SetActive(true);

        box.gameObject.SetActive(true);
        box.position = mapData.ListOfBoxData[i].Position;
        box.rotation = mapData.ListOfBoxData[i].Quaternion;
        Debug.Log("check obj name  :  " + box.name);
        if (box.name.Contains("Transport"))
        {
            GameObject warpEffect = Resources.Load<GameObject>("Effects/Prefabs/WarpEffect");
            warpEffect = Instantiate(warpEffect, new Vector3(box.transform.position.x,box.tran
            warpEffect.transform.SetParent(box.transform);
        }
        box.SetParent(newParent.transform);
    }
```

<레벨 읽기 및 활성화>

플레이를 누르면 Main패널에 추가된 명령어리스트를 가져와 순차적으로 실행합니다. G1, G2 명령 명령 수행될 때 G1, G2을 스택에 추가합니다. 명령어에 따른 기능은 Switch 문을 사용하고 해당 기능이 실행되는 시간을 반환하도록 했습니다.

```
switch (panelName)
{
    case "Go":
        ...

    case "Jump":
        ...
    case "Left":
        ...
        break;
    case "Right":
        for (int i = 0; i < selectedRobot.transform.GetChild(0).childCount; i++)
            AnimationAgent.Instance.setInteger(selectedRobot.transform.GetChild(0).GetChild(i).name, "State", 6);

        time = ResourcesManager.Instance.animationClips["robot_turn_R"].length - 0.1f;
        StartCoroutine(MoveAgent.Instance.turnRight(selectedRobot, time / speed));

        break;
    case "Action":
        ...
        break;
}

return time / speed;
```

명령어를 순차적으로 수행하는 코루틴의 일부 코드입니다.

```
Stack<BaseInfo> stackOfBase = new Stack<BaseInfo>();

var curBase = baseMain;
var curIndicator = indicator_main;
Color alphaColor = curIndicator.GetComponent<Image>().color; alphaColor.a = 0.5f;
Color originColor = curIndicator.GetComponent<Image>().color; originColor.a = 1.0f;

for (int i = 0; i < curBase.childCount; i++)
{
    var panel = curBase.GetChild(i);
    curIndicator.transform.position = panel.position;

    if (panel.name.CompareTo("G1") == 0 && baseG1.childCount != 0)...
    if (panel.name.CompareTo("G2") == 0 && baseG2.childCount != 0)...


    float time = moveByPanel(panel.name);
    curIndicator.transform.position = panel.position;

    if (level.Count == 0)...


    yield return new WaitForSeconds(time);
```