

DIY(Draw Idea Yourself)

- Oculus VR을 연동한 과학상자 프로그램 - 프로젝트 완료 보고서

- 제정일자 : 2018년 11월 20일
- 문서버전 : Ver 2.0
- 팀명 : 제발만들어주세요

문 서 승 인 정 보

프로젝트 명	DIY(Draw Idea Yourself)
TASK 명	Oculus VR을 연동한 과학상자 프로그램
문 서 명	프로젝트 완료보고서
발행 년 월일	2018년 11월 21일

구 분	성 명	서 명	일 자
Team Member	유 경 동		2018.11.20
Team Member	한 다 인		2018.11.20
Team Member	백 주 성		2018.11.20
Team Leader	이 준 섭		2018.11.20
프로젝트관리자	문 상 환		2018.11.20

문서이력정보

[illegible]

목차

1. 프로젝트 개요
 - A. 프로젝트 명칭과 그 의미
 - B. 프로젝트 기획 의도
 - C. 프로젝트 방향
2. 프로젝트 진행 보고
 - A. 프로젝트 개발 분야
 - B. 프로젝트 진행 일정
 - C. 수행업무 및 담당자
 - D. 개발 도구
 - E. 프로젝트 목표
 - F. 단계별 아키텍처
3. 프로젝트 개발 일정
 - A. 팀 회의록
 - B. 팀 개발 일정
 - C. 개인 개발 일정
4. 프로젝트 개발 내용
 - A. 프로젝트 배경 지식/기술/알고리즘
 - B. 프로젝트 상세 개발 내용
5. 사용자 매뉴얼
6. 프로젝트 마무리
 - A. 기대효과
 - B. 난제 극복 사례
 - C. 문제점
 - D. 개선방안
 - E. 참고문헌 및 논문

F. 참고사이트

G. 팀원 별 소감

1. 프로젝트 개요

A. 프로젝트 명칭과 그 의미

DIY 는 Draw Idea Yourself 라는 의미로 사용자가 생각하는 대로 새로운 세상을 표현한다는 의미를 내포하고 있다. 이는 동일한 약어를 가지는 Do it Yourself와 유사하게 스스로 만들 수 있다는 의미를 최대한 살리고, 프로젝트의 뜻을 직관적으로 전달하고 싶어 그렇게 프로젝트의 이름을 설정하였다.

B. 프로젝트 기획 의도

프로젝트의 명칭 DIY(Draw Idea Yourself)의 의미대로 사용자가 생각한 대로 세상을 표현하는 프로그램 개발을 목표로 했기에 최대한 그 의도를 반영할 소재를 선택하기 위해 노력하였다. 명칭 안에 있는 '표현한다'는 의미를 최대한 살리고 누구나 쉽게 만들 수 있는 프로그램을 제작하고자 하였는데, 적합하게 떠오른 소재가 과학 상자와 Paint 기능의 프로그램이었다. 과학 상자는 초등학생 정도에서도 쉽게 연결하고 움직이게 할 수 있으며, Paint 프로그램 역시 그리기 방법만 안다면 본인이 원하는 그림을 자유롭게 그릴 수 있다. 그렇기에 사용의 제약이 크지 않은 두 소재를 결합하여 프로그램으로 만든다면 자신의 상상력을 더 확장할 수 있다는 판단이 들게 되었고, 이를 VR 장치와 결합하여 생동감 있는 프로그램을 제작하고자 하였다.

C. 프로젝트 방향

'만들다' 와 '그리다' 의 아이디어를 결합한 표현 툴의 구현을 목표로 진행하였으며, 아이디어의 벤치마킹 소재로 과학상자와 Oculus VR 에서 제공하는 Tilt Brush를 삼았다. 여기에 과학상자를 움직이고자 웹에서 제공하는 애플리케이션인 Scratch라는 교육용 프로그램의 형태를 벤치마킹하여 과학상자 및 Brush에 움직임을 제공하고자 하였다. 이 내용들을 결합할 개발 툴로써 Unity를 선택하였고, Unity에 해당 아이디어를 녹여내고자 하였다. 그렇게 하면서 동작은 최대한 쉽고, 직관적으로 함으로써 아이들이 쉽게 설계하고 디자인할 수 있도록 하였다. 프로그램으로 제작하기 때문에 과학 상자가 가지는 공간이나 소재의 제약성을 극복하고 Tilt Brush에서 주는 드로잉 기술을 결합하여 아이들이 최대한 본인들의 상상력을 키울 수 있는 능력을 기를 수 있는 것을 목표로 진행하였다.

2. 프로젝트 진행 보고

A. 프로젝트 개발 분야

- (가) 과학 상자의 소재로 사용될 기본 Component 제작
- (나) 기본 Component로 제작한 Kit 제작 기능
- (다) Tilt Brush 기능을 구현하여 Drawing Tool 제공
- (라) DIY 프로그램의 요소들을 움직이게 하는 스크립트 제작
- (마) Oculus VR 과 DIY 프로그램 연동

B. 프로젝트 진행 일정

1) 1순위



업무 분야	9월	10월																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
		일	월	화	수	목	금	토	일	월	화	수	목	금	토	일																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																							
1순위	Unity 설치	↑																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					

3) 3순위

업무 분야	구분	11월																			
		30 31		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
3순위	Tilt Brush																				
	항목 없음 항목 없음																				
	스크립트 컴포넌트 삭제																				
	스크립트 삭제																				
	제어함수 삭제																				
	저장하기//불러오기																				
	예시 오브젝트 만들기																				
과학상자	Object 회전 및 축소, 늘리기																				
	Object 색 변경																				

C. 수행업무 및 담당자

책임	직무	사진	이름	역할 설명
Leader	팀장		이준섭	일정 관리 기획서 관리 회의 주제 및 회의록 정리
과학상자	정		유경동	기본Component제작, Kit 제작, 선택 UI 제작
	부		이준섭	기본 스크립트 제작, 스크립트 프레임 제작, 스크립트 선택 Set 제작

Tilt Brush	정		한다인	Brush 기능 구현, Brush Line 그리기 구현
	부		백주성	Tilt Brush UI 제작
Oculus Rift	정		한다인	Oculus Rift에 사용할 Ray 구현, Oculus Rift 와 프로그램 Interaction
	부		이준섭	Oculus Rift와 UI 연동 Unity와 Oculus Rift 연동

D. 개발 도구

A. 기술

- Unity, C#

B. 개발 S/W

- Visual Studio 2017, Unity, Oculus

C. 장비

- Oculus Rift(Oculus Headset, Oculus Touch x 2, Oculus Sensor x 2), 그래픽 카드 GTX 960 미니

E. 프로젝트 목표

- 1단계 : 전체 프로젝트 달성을 위한 핵심적인 기능 구현

업무 분야	기능
Tilt Brush	① 선 그리기: 화면에 그림을 표현하는 것이 목적이다. A. Trail Renderer: 오브젝트의 움직임을 파악하여 그 자취를 따라가며 선을 Render 해준다. i. 색 변경: 선의 색을 변경한다. ii. 두께 조절: 선의 두께를 조절한다. iii. 공간감 표현: 화면상에서 공간의 깊이를 표현해준다. iv. 지우기: 전체 그림을 삭제해주거나 내가 원하는 부분을 지워준다. ② UI 만들기 A. Unity로 만들어진 팔레트 UI: i. 위의 선 그리기에 해당하는 항목들을 선택할 수 있다. ii. 팔레트 UI를 On/Off 할 수 있다. ③ Tilt Brush On/Off i. 그려진 그림에 대한 Render를 꺼주어 화면상에 그림이 보이지 않게 할 수 있다.
과학 상자	① Object 동작 : 완성된 Object 혹은 해당 Object 내에 있는 컴포넌트의 움직임을 표현하는 것이 목적이다. A. Object 회전 : 바퀴의 회전, 물체의 회전, 조인트 회전 등이 있다. i. 바퀴 회전 : 바퀴 회전 시 Object 움직임을 표현한다. ii. 조인트 회전 : 물체 Object 내 두 Object 접합부에 부착된 Joint에 의한 회전 움직임을 표현한다. iii. 물체 회전 : Base 물체 전체의 회전 움직임을 표현한다.

	<p>② .스크립트 : Object의 동작 제어를 위해 작성하는 스크립트를 의미한다.</p> <p>A. 제어 함수 : 스크립트 컴포넌트를 조합한 스크립트 셋, 스크립트 셋을 조합하여 만든 제어 함수를 통해 Object의 동작을 제어한다.</p> <ul style="list-style-type: none"> i. 스크립트 컴포넌트 : 스크립트 구성 요소를 의미한다. 주로 Transform 관련 변수나 단순한 움직임만을 의미한다. ii. 스크립트 : 움직임 동작을 의미한다. 자체로도 제어 함수가 될 수 있으며, 스크립트들의 조합으로 자연스러운 움직임을 제어할 수 있다. iii. 제어 함수 : 스크립트들의 조합으로 이뤄지며, 연속적인 동작을 만들어 Object마다 움직임을 정의할 수 있다. <p>B. 스크립트 UI : 스크립트, 제어 함수 등을 사용자 편의에 맞춰 조합할 수 있는 UI이다.</p> <ul style="list-style-type: none"> i. 스냅 : 스크립트들을 결합할 때 자석처럼 부드럽게 붙게 해준다.. ii. 묶기 : 스크립트 안의 스크립트 구조로 만들어준다. 즉 부모 - 자식 관계로 만들어 한 쪽의 스크립트 기능이 종속되게 한다. iii. 아웃라인 On/Off : 스냅이 될 대상이 표시되도록 바깥 선을 보여준다.
--	--

- 2단계 : Tilt Brush / 과학 상자에서 보다 현실에 가까운 느낌을 주기 위한 기능들을 추가해준다. 그 뿐 아니라 작업 수행에 있어서 유용하게 사용 가능한 기능들을 추가해준다

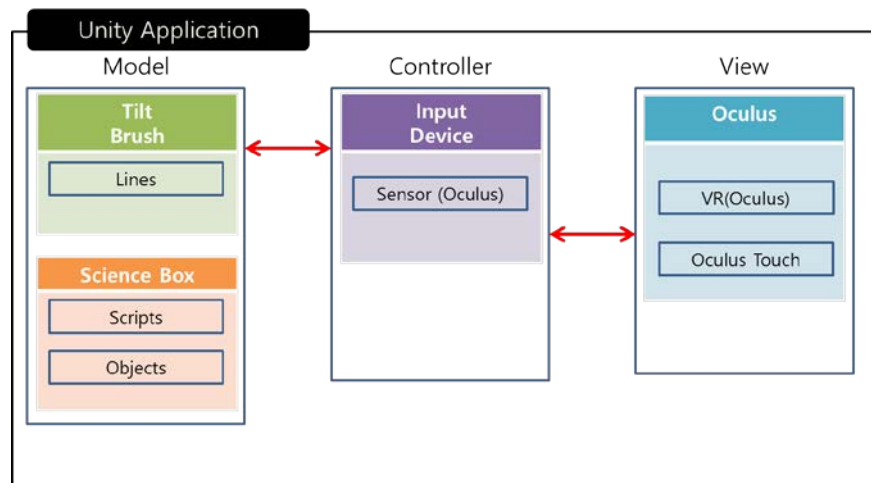
업무 분야	기능
1단계까지 개발된 기능은 기본적으로 포함된다.	
Tilt Brush	① 선 그리기 확장 : 1단계에서 단순하게 표현된 선을 세밀하게 묘사할 수 있으며, 이 정보를 저장하여 보관할 수 있다. A. 그림 저장하기/불러오기 : DB에 좌표 정보를 저장하여 그린 그림을 저장하거나 가져올 수 있다. B. 그림 실행 취소/되돌리기 : 그린 그림을 직전의 상태로 되돌릴 수 있다.
과학상자	① 스크립트 편집 : 작성한 스크립트 뿐 아니라 컴포넌트, 제어 함수를 편집할 수 있다. A. 컴포넌트 편집 : 스크립트에 등록한 컴포넌트를 편집 B. 스크립트 편집 : 작성된 스크립트의 내용을 편집 C. 제어함수 편집 : 등록된 제어함수의 구성을 편집 D. 스크립트 배치 : 3D 화면에서 스크립트 UI 간 간격이 일정하게 유지되고 방향이 일정하게 유지되도록 기능 구현 ② 세밀한 Object 배치 : 선택된 Object와 설치할 Object의 위치를 정확한 곳에 배치할 수 있게 한다. A. Object 투명도 : 선택된 Object가 다른 Object에 가려질 경우 가리는 대상을 투명하게 해준다. B. Object 외곽선 : Object 간 결합 시 결합될 대상 Object에 외곽선을 주어 표시해준다. ③ Object Set(Kit) 제작 : 예시로 사용하거나 혹은 자신이 만들고 싶은 Kit를 저장하거나 불러올 수 있다.

- 3단계 : 독립적으로 작업한 내용들을 서로 연결해주고, 구현은 완료되었으나 사용자 인터페이스가 최대한 직관적으로 보일 수 있도록 업무 분야들을 수정 / 보완하는 방향으로 진행한다

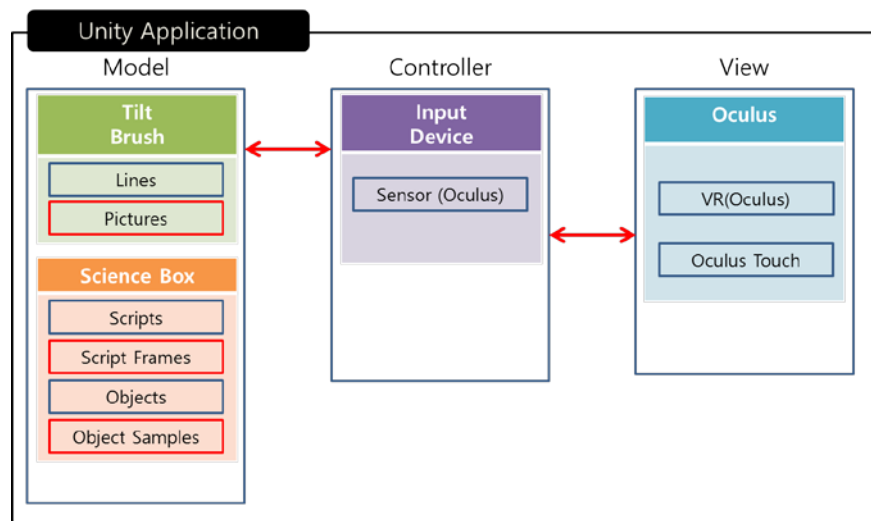
업무 분야	기능
2단계까지 개발된 기능은 기본적으로 포함된다.	
Tilt Brush	추가 개발 사항 없음
과학상자	① 스크립트 삭제 : 작성한 스크립트 뿐 아니라 컴포넌트, 제어 함수를 삭제할 수 있다. A. 컴포넌트 삭제 : 스크립트에 등록한 컴포넌트를 삭제 B. 스크립트 삭제 : 작성된 스크립트의 내용을 삭제 C. 제어함수 삭제 : 등록된 제어함수를 삭제

F. 단계별 아키텍처

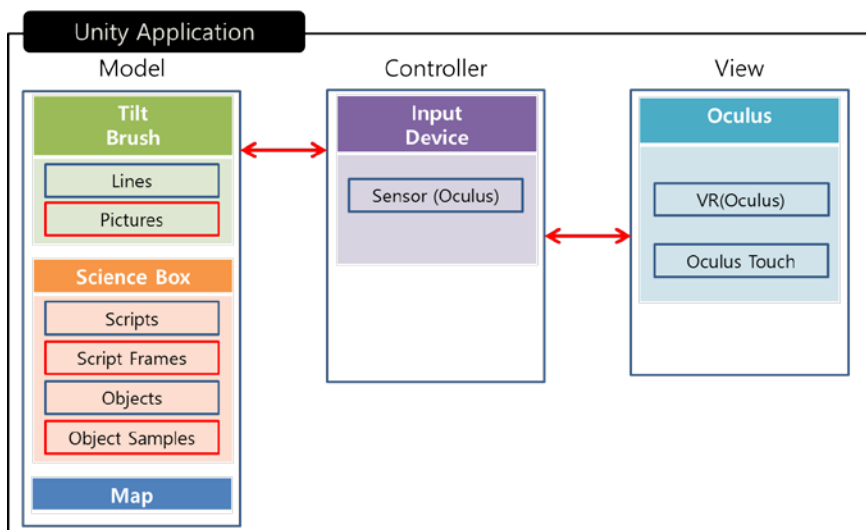
● 1차 아키텍처



● 2차 아키텍처



● 3차 아키텍처



3. 프로젝트 개발 일정

A. 팀 개발 일정

2018 년 09 월 27 일			
참석자	이준섭, 하홍복, 한다인, 유경동		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	Tilt Brush 팔레트 UI 기능 구현
			Unity 로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Tilt Brush Oculus 에 연동하여 선그리기 완료
			선 색, 두께 변경 구현 목표.
	과학 상자	작업 진행 상황	
		향후 계획	Joint 선택 회전 Test
			Object 투명도 스크립트 Test 완료
			WheelCollider 를 이용한 자동차 컨트롤 구현 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 구현

2018 년 09 월 28 일	
참석자	이준섭, 하홍복, 유경동, 한다인

안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Unity 로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
		향후 계획	Tilt Brush 팔레트 UI 기능 구현
			Tilt Brush Oculus 에 연동하여 선그리기 완료
	Tilt Brush Oculus 에 연동하여 선그리기 완료		
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
		향후 계획	스크래치 스크립트 동작 구현
2018 년 10 월 01 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.

		향후 계획	Oculus에 연동해서 선 지우기
			그려진 선 Rendering on/off
			선 색, 두께 변경 구현 목표.
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 완료
		향후 계획	Mirror 기능 구현
			Object Prefab 하기
			스크래치 Script 구조 및 기능 구현
	스크래치 Script와 UI 연결		
2018 년 10 월 02 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		

내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
		향후 계획	Oculus에 연동해서 선 지우기
			그려진 선 Rendering on/off
	선 색, 두께 변경 구현 목표.		
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결
		향후 계획	Mirror 기능 구현
			Object Prefab 하기
			스크래치 Script 구조 및 기능 구현

2018 년 10 월 04 일			
------------------	--	--	--

참석자	이준섭, 유경동, 백주성		
-----	---------------	--	--

안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
		향후 계획	Oculus에 연동해서 선 지우기
			그려진 선 Rendering on/off
			선 색, 두께 변경 구현 목표.
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결
		향후 계획	Mirror 기능 구현
			Object Prefeb 하기
			스크래치 Script 구조 및 기능 구현

2018 년 10 월 05 일

참석자	이준섭, 유경동		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
		향후 계획	Oculus에 연동해서 선 지우기
			그려진 선 Rendering on/off
			선 색, 두께 변경 구현 목표.
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 UI 완성

			스크래치 UI에 변수 추가 작업 중
		향후 계획	Mirror 기능 구현
			Object Prefab 하기
			과학 상자 세부 UI 툴 / 기능 완성
			스크래치 Script 구조 및 기능 구현
			스크래치 UI 변수 추가
2018 년 10 월 08 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
		향후 계획	Brush로 그린 그림 저장 / 불러오기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완

			료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
		향후 계획	Oculus Raycast 구현
			Mirror 기능 구현
			Object Prefeb 하기
			과학 상자 세부 기능 완성
			과학 상자 Block 쌓기 구현
			제어 스크립트 동적 생성
2018 년 10 월 10 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안건	작업 진행 상황 확인		

향후 계획			
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
		향후 계획	Brush로 그린 그림 저장 / 불러오기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
		향후 계획	Mirror 기능 구현

			Object Prefab 하기
			과학 상자 세부 기능 완성
			과학 상자 Block 쌓기 구현
			제어 스크립트 동적 생성
2018 년 10 월 11 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
		향후 계획	Brush로 그린 그림 저장 / 불러오기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료

			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성
			과학 상자 Block 쌓기 구현
		향후 계획	Object Prefab 하기
			Object 바퀴 달기
			과학 상자 세부 기능 완성
			과학 상자 - 스크래치 스크립트 통합
			Mirror 기능 구현
2018 년 10 월 12 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		

향후 계획			
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
			Brush로 그린 그림 저장 / 불러오기
		향후 계획	Brush 재질 변경하기
			WheelCollider를 이용한 자동차 컨트롤 구현 완료
	과학 상자	작업 진행 상황	Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성

			과학 상자 Block 쌓기 구현
			Object Prefab 하기
		향후 계획	Object 바퀴 달기
			과학 상자 세부 기능 완성
			과학 상자 - 스크래치 스크립트 통합
			Mirror 기능 구현
			Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
2018 년 10 월 15 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
			Brush로 그린 그림 저장 / 불러오기 (코드 구현)

	과학 상자	향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
		작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성
			과학 상자 Block 쌓기 구현
			Object 바퀴 달기
			Object Prefab 하기
		향후 계획	과학 상자 세부 기능 완성
			과학 상자 - 스크래치 스크립트 통합

			Mirror 기능 구현
2018 년 10 월 16 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
			Brush로 그린 그림 저장 / 불러오기 (코드 구현)
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료

			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성
			과학 상자 Block 쌓기 구현
			Object 바퀴 달기
			Object Prefab 하기
		향후 계획	과학 상자 세부 기능 완성
			과학 상자 - 스크래치 스크립트 통합
			Mirror 기능 구현
2018 년 10 월 17 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안건	1차 발표		

내용	1차 발표		
2018 년 10 월 18 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안건	향후 계획		
	Tilt Brush	향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Oculus Brush 선 에러 수정
			Brush 전체 지우기
			Brush 재질 변경하기
			Color 팔레트 추가
	과학 상자	향후 계획	Mirror 기능 구현
			Object UI 선택 창 기능 구현
			스크립트 윤곽선 구현
			스크립트 자석 구현
			과학상자 Oculus 연동
2018 년 10 월 19 일			

참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료
			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
			Brush로 그린 그림 저장 / 불러오기 (코드 구현)
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료

			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성
			과학 상자 Block 쌓기 구현
			Object 바퀴 달기
			Object Prefab 하기
		향후 계획	과학 상자 세부 기능 완성
			과학 상자 - 스크래치 스크립트 통합
			Mirror 기능 구현
2018 년 10 월 22 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Tilt Brush Oculus에 연동하여 선그리기, 색 변경, 두께 변경 완료

			Unity로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
			Oculus에 연동해서 선 지우기
			Brush로 그린 그림 저장 / 불러오기 (코드 구현)
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	WheelCollider를 이용한 자동차 컨트롤 구현 완료
			Joint 선택 회전 Test 완료
			Object 투명도 스크립트 Test 완료
			Object 윤곽선 스크립트 Test 완료
			스크래치 스크립트 동작 코드 구현 완료
			스크래치 Script와 UI 연결 완료
			과학 상자 기본 / 세부 UI 완성
			스크래치 UI에 변수 추가 완료
			Oculus Raycast 구현
			제어 스크립트 동적 생성
			과학 상자 Block 쌓기 구현

		향후 계획	Object 바퀴 달기
			Object Prefab 하기
			Mirror 기능 구현
2018 년 10 월 23 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현

			스크립트 프레임에 넣기 / 빼기
			스크립트 깊이 인식 배치
			과학 상자 - 스크래치 스크립트 통합
2018 년 10 월 24 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현
			스크립트 프레임에 넣기 / 빼기

			스크립트 깊이 인식 배치
			과학 상자 - 스크래치 스크립트 통합
2018 년 10 월 25 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 전체 지우기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현
			스크립트 프레임에 넣기 / 빼기

			스크립트 깊이 인식 배치
			과학 상자 - 스크래치 스크립트 통합
2018 년 10 월 26 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 전체 지우기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현
			스크립트 프레임에 넣기 / 빼기

			스크립트 깊이 인식 배치
			과학 상자 - 스크래치 스크립트 통합
2018 년 10 월 29 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
			Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
		향후 계획	Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush 선택 지우기
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현

		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			스크립트 깊이 인식 배치
			스크립트 프레임에 넣기 / 빼기
			이족보행 Animation 적용
			포크레인 컴포넌트 제작
2018 년 10 월 30 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
			Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
		향후 계획	Brush 선택 지우기

			Keyboard UI에 기존 Text값 초기화 해결
			Brush 재질 변경하기
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
			스크립트 프레임에 넣기 / 빼기
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			스크립트 깊이 인식 배치
			이족보행 Animation 적용
			포크레인 컴포넌트 제작
2018 년 10 월 31 일			
참석자	이준섭, 유경동, 한다인, 백주성		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동

			Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 선택 지우기
		향후 계획	Keyboard UI에 기존 Text값 초기화 해결
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
			스크립트 프레임에 넣기 / 빼기
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)			
스크립트 깊이 인식 배치			
이족보행 Animation(움직임) 적용			
제어구문 스크립트 깊이 설정			
	포크레인 컴포넌트 제작		
2018 년 11 월 01 일			

참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
			Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 선택 지우기
			Keyboard UI에 기존 Text값 초기화 해결
		향후 계획	Brush로 그린 그림 저장 / 불러오기 (Build 시 문제 해결)
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
			스크립트 프레임에 넣기 / 빼기
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 /

			자동차 Kit / 이족보행)
			스크립트 깊이 인식 배치
			이족보행 Animation(움직임) 적용
			제어구문 스크립트 깊이 설정
			포크레인 컴포넌트 제작
2018 년 11 월 02 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	File Browser와 VR Keyboard 연동
			Brush 전체 지우기
			File Load 시 redo / undo 기능 수정 보완
			Brush로 그린 그림 저장 / 불러오기 (Oculus에 적용)
			Brush 선택 지우기
			Keyboard UI에 기존 Text값 초기화 해결
			Brush로 그린 그림 저장 / 불러오기 (Build 시 문제 해결)

			팔레트UI 오른쪽 손에 붙이기
		향후 계획	Brush 재질 변경하기
			색 팔레트 추가
	과학 상자	작업 진행 상황	Object Prefab 하기
			Mirror 기능 구현
			스크립트 프레임에 넣기 / 빼기
		향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			스크립트 깊이 인식 배치
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			포크레인 컴포넌트 제작
			2018 년 11 월 05 일
참석자	이준섭, 유경동, 한다인		
안건	2 차 발표		

내용	2 차 발표		
2018 년 11 월 06 일			
참석자	이준섭, 유경동, 한다인		
안건	향후 계획		
내용	Tilt Brush	향후 계획	Map 만들기
			다중 라인 객체 저장
			라인 좌표 이동
	과학 상자	향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			포크레인 컴포넌트 제작
2018 년 11 월 07 일			
참석자	이준섭, 유경동, 한다인		

안건	향후 계획		
내용	Tilt Brush	향후 계획	Map 만들기
			라인 좌표 이동
	과학 상자	향후 계획	다중 스크립트 프레임 구성
			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			포크레인 컴포넌트 제작
2018 년 11 월 08 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	Map 만들기
			라인 좌표 이동
	과학 상자	작업 진행 상황	
향후 계획		다중 스크립트 프레임 구성	

			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			포크레인 컴포넌트 제작
			동력 Script 제작
2018 년 11 월 09 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	Unity 로 Tilt Brush 팔레트 기본 UI / 세부 UI 완료
			Tilt Brush 팔레트 UI ON/OFF 완료.
		향후 계획	Map 만들기
			라인 좌표 이동
	과학 상자	작업 진행 상황	동력 Script 제작
			포크레인 컴포넌트 제작
		향후 계획	다중 스크립트 프레임 구성

			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			Key Script 제작
			Frame 삭제 기능
2018 년 11 월 12 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	Map 만들기
			라인 좌표 이동
	과학 상자	작업 진행 상황	동력 Script 제작
			포크레인 컴포넌트 제작
			Frame 삭제 기능
		향후 계획	다중 스크립트 프레임 구성

			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			Key Script 제작
2018 년 11 월 13 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	Map 만들기
			라인 좌표 이동
	과학 상자	작업 진행 상황	동력 Script 제작
			포크레인 컴포넌트 제작
			Frame 삭제 기능
			UI에서 컴포넌트 선택 배치
		향후 계획	다중 스크립트 프레임 구성

			과학 상자 세부 기능 완성
			Oculus 컴포넌트 불러오기 기능 구현(사각뿔 / 자동차 Kit / 이족보행)
			이족보행 Animation(움직임) 적용 및 보완
			제어구문 스크립트 깊이 설정
			Key Script 제작
2018 년 11 월 14 일			
참석자	이준섭, 유경동, 한다인		
안건	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	
	과학 상자	작업 진행 상황	
		향후 계획	Joint 선택 회전 Test 스크래치 스크립트 동작 구현
2018 년 11 월 15 일			
참석자	이준섭, 유경동, 한다인,		
안건	작업 진행 상황 확인		

	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	전체 기능 통합
	과학 상자	작업 진행 상황	
		향후 계획	전체 기능 통합
2018 년 11 월 16 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	전체 기능 통합
	과학 상자	작업 진행 상황	
		향후 계획	전체 기능 통합
2018 년 11 월 19 일			
참석자	이준섭, 유경동, 한다인,		
안전	3차 발표		

내용	3 차 발표		
2018 년 11 월 20 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	세부 기능 보완
			프로그램 디버깅
	과학 상자	작업 진행 상황	
		향후 계획	세부 기능 보완
			프로그램 디버깅
2018 년 11 월 21 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	

		향후 계획	세부 기능 보완
			프로그램 디버깅
	과학 상자	작업 진행 상황	
		향후 계획	세부 기능 보완
프로그램 디버깅			
2018 년 11 월 22 일			
참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	세부 기능 보완
	프로그램 디버깅		
	과학 상자	작업 진행 상황	
		향후 계획	세부 기능 보완
	프로그램 디버깅		
2018 년 11 월 23 일			

참석자	이준섭, 유경동, 한다인		
안전	작업 진행 상황 확인		
	향후 계획		
내용	Tilt Brush	작업 진행 상황	
		향후 계획	세부 기능 보완
			프로그램 디버깅
	과학 상자	작업 진행 상황	
		향후 계획	세부 기능 보완
			프로그램 디버깅

B. 팀 개발 일정

① 1주차

회차	팀원명	금일 실제 진행한 일
날짜		09 월 27 일
1 회차	이준섭	스크래치 동작 및 제어 부분 UI 분석 스크래치 코드 구현을 위한 C# Template, Delegate, 가변인수 학습
	하홍복	Object 투명도 스크립트 작성 완료 및 단위 테스트 확인 Object 윤곽선 스크립트 작성 완료 및 단위 테스트 확인
	유경동	Joint Object 중심으로 회전하는 스크립트 작성 완료 선택한 Joint Object 회전 스크립트 작성 완료
	한다인	Tilt Brush 기능 관련 자료 탐색 완료 Oculus Rift Touch 관련 자료 탐색 완료 Unity 와 Oculus 연동 방법 탐색 완료
	백주성	팔레트 UI 기초 틀 완성
날짜		09 월 28 일
2 회차	이준섭	스크래치 동작(Action) 코드만 연결 Test 완료
	하홍복	Oculus Grabbing Test
	유경동	무한궤도 차량 동작 조사
	한다인	Oculus Rift Touch 로 Unity Line Renderer 적용
	백주성	팔레트 세부 UI 구상 및 제작

② 2주차

회차	팀원명	금일 실제 진행한 일
날짜		10 월 01 일
1 회차	이준섭	스크립트 UI 이동 / 복사 기능 구현 스크립트 제어 부분 Test
	하홍복	결석
	유경동	Mirror 관련 자료 탐색 Mirror 기본 Test
	한다인	선 지우기 자료 탐색 선 그리기 원리 분석
	백주성	팔레트 세부 UI 구현
날짜		10 월 02 일
2 회차	이준섭	스크립트 코드 Type 캐스팅 구현 스크립트 클래스 대표 Interface, Abstract Class 구현 Frame 스크립트 정의 Code 구현
	하홍복	결석
	유경동	Object Drag 코드 구현 Mirror 스크립트 기초 작업 수행
	한다인	선 지우기 자료 탐색 선 그리기 원리 분석
	백주성	팔레트 UI On / Off 구현 Oculus 에 팔레트 UI 적용
날짜		10 월 04 일
3 회차	이준섭	스크립트 사용자 정의 변수 만들기 진행 중
	하홍복	결석
	유경동	과학상자 UI 기본 틀 완성
	한다인	결석
	백주성	팔레트 UI 색상 변경 구현 중

날짜		10 월 05 일
4 회차	이준섭	스크립트 변수 입/출력 구현 스크립트 UI 프레임 배치 기본 Test
	하홍복	결석
	유경동	과학상자 세부 UI 및 기능(Script) 구현 진행 중
	한다인	Oculus Touch 로 그린 그림의 선 지우기 구현 Oculus Touch 로 그린 선 Redo / Undo 구현
	백주성	결석

③ 3주차

회차	팀원명	금일 실제 진행한 일
날짜		10 월 08 일
1 회차	이준섭	제어 스크립트 깊이 적용
	하홍복	결석
	유경동	Block 배치 지점 탐색 자료 분석 Cube 배치 자료 분석
	한다인	Oculus Touch 와 Unity Raycast 연동 및 제어 완료
	백주성	Unity Script 학습
날짜		10 월 10 일
2 회차	이준섭	동적 제어 스크립트 제작 구현 완료
	하홍복	결석
	유경동	Block 배치 지점 탐색 및 배치 구현
	한다인	저장하기 / 불러오기 탐색
	백주성	개인공부
날짜		10 월 11 일

3 회차	이준섭	스크립트 동작 정밀화 구현(90%)
	유경동	Wheel Collider 적용 분석(RigidBody 이용)
	한다인	저장하기 / 불러오기 탐색
	백주성	개인공부
날짜		10 월 12 일
4 회차	이준섭	스크립트 동작 정밀화
	유경동	Cube Prefab 화 하기 Cube 로 만든 Object 바퀴 결합
	한다인	Tilt Brush 그림 그린 것 저장하기/ 불러오기 (Unity 적용)
	백주성	결석

④ 4주차

회차	팀원명	금일 실제 진행한 일
날짜		10 월 15 일
1 회차	이준섭	오브젝트 크기 확장 오브젝트 바퀴 구성 Set 코드 구현 중
	유경동	오브젝트에 연결된 바퀴 회전 구현 중
	한다인	Line Mesh Objects 를 저장하고 불러오기(Oculus 연동) (90%)
	백주성	Unity 학습
날짜		10 월 16 일
2 회차	이준섭	과학상자 스크래치 통합

	유경동	과학상자 스크래치 통합
	한다인	PPT 발표 자료 준비(Tilt Brush)
	백주성	Unity 학습 발표 자료 준비(Tilt Brush UI)
날짜		10 월 17 일
3 회차	이준섭	1 차 발표
	유경동	1 차 발표
	한다인	1 차 발표
	백주성	1 차 발표
날짜		10 월 18 일
4 회차	이준섭	Mirror 기능 구현
	유경동	UI Cube 선택 / 배치 구현
	한다인	Procedural Mesh 원리 학습
	백주성	개인 공부
날짜		10 월 19 일
5 회차	이준섭	스크립트 자석 구현
	유경동	UI 를 통해서 Object 선택 기능 구현
	한다인	VR 키보드 구현
	백주성	개인 공부

⑤ 5주차

회차	팀원명	금일 실제 진행한 일
날짜		10 월 22 일
1 회차	이준섭	스크립트 회전 시 회전 각 맞추기
	유경동	과학상자 배치 UI Oculus 연동
	한다인	File Browser 코드 분석 및 구현
	백주성	개인 공부
날짜		10 월 23 일
2 회차	이준섭	스크립트 Gizmo 배치 구현
	유경동	Oculus Switch RayCast 로 Object 선택하기
	한다인	File Browser 와 VR Keyboard 연동 VR 내 Sava 기능 구현
	백주성	개인 공부
날짜		10 월 24 일
3 회차	이준섭	Frame GameObject 생성 구현 스크립트 프레임 이동 구현 스크립트 배치 깊이 맞추기
	유경동	Oculus Controller Raycast 로 Object 배치 구현
	한다인	Save / Load VR 과 연동 구현(Load 는 취약점 보완 필요)
	백주성	개인 공부
날짜		10 월 25 일
4 회차	이준섭	컴포넌트에 스크립트 적용 구현
	유경동	Oculus 와 연동하여 Prefab 생성하기 Oculus 에서 생성된 Object 를 작동시키기
	한다인	Save / Load 기능점검 및 보완 구현

		팔레트의 UI On/Off 구현
	백주성	개인 공부
날짜		10 월 26 일
5 회차	이준섭	스크립트 넣기 및 순서 변경기능 구현
	유경동	필요한 Component 구상 Kit 에 필요한 Component 제작
	한다인	라인 그리기 오류 고치기 파일 브라우저 On / Off 동작 문제 해결 브러쉬 Default 색상 선택
	백주성	개인 공부

⑥ 6주차

회차	팀원명	금일 실제 진행한 일
날짜		10 월 29 일
1 회차	이준섭	3D Builder 오브젝트 만들기 확인 스크립트 넣기 빼기 구현(스크립트 프레임 길이 제외)
	유경동	Object Square Pyramid 만들기 자동차 kit 를 오브젝트(Prefab) 제작
	한다인	Load Undo/Redo 예외 처리 전체 Line 삭제
	백주성	개인 공부
날짜		10 월 30 일
2 회차	이준섭	스크립트 넣기 빼기 구현 제어문 틀 만들기
	유경동	Kit 만들기(이족보행)
	한다인	라인 하나씩 지우기
	백주성	개인 공부

날짜		10 월 31 일
3 회차	이준섭	제어문 같은 깊이 배치 구현
	유경동	2 족 보행 kit 제작
	한다인	Parlette 오른손에 붙이고 따라다니게 하기 Keyboard UI 에 계속 text 가 남아있는 현상 해결
	백주성	개인 공부
날짜		11 월 01 일
4 회차	이준섭	제어문 다른 깊이 배치 구현
	유경동	2 족 보행 kit 제작 및 움직이게하기
	한다인	Save / Load 문제 해결
날짜		11 월 02 일
5 회차	이준섭	인접 스크립트 표시 기능 구현 프레임 적용 대상 표시 기능 구현 제어문 배치 프레임 기능 구현 만든 프레임 Renderer 활성화 / 비활성
	유경동	스크립트와 과학상자 연동
	한다인	컬러 팔레트 관련 자료 조사

⑦ 7주차

회차	팀원명	금일 실제 진행한 일
날짜		11 월 05 일
1 회차	이준섭	2 차 발표 준비
	유경동	2 차 발표 준비
	한다인	2 차 발표 준비
날짜		11 월 06 일
2 회차	이준섭	Rigidbody Script 제작
	유경동	과학상자 UI 세부 기능 구현

	한다인	저장된 다중 Line 불러오기
날짜		11 월 07 일
3 회차	이준섭	동력 Script 제작
	유경동	Rigidbody 적용한 물체 Raycast 인식
	한다인	휴일
날짜		11 월 08 일
4 회차	이준섭	동력 Script 제작 Wheel Collider 스크립트 연동
	유경동	포크레인 키트 삽 추가하기
	한다인	게임 오브젝트 Grab 해서 위치 옮기기 구현 중
날짜		11 월 09 일
5 회차	이준섭	Frame 삭제 기능 구현
	유경동	과학상자 UI Prefab 불러오기 문제 해결
	한다인	GameObject Grab 해서 위치 옮기기 구현 중

⑧ 8주차

회차	팀원명	금일 실제 진행한 일
날짜		11 월 12 일
1 회차	이준섭	Key Script 제작 중 Tensorflow 환경 설정 및 학습
	유경동	머신러닝의 개념과 용어 이해하기 TensorFlow 설치 및 기초 코드 작업해보기
	한다인	Tensorflow 설치 및 인터넷 강의 듣기
날짜		11 월 13 일
2 회차	이준섭	Key Script 제작 중 Tensorflow 학습

	유경동	Linear Regression 의 개념 TensorFlow 로 Linear Regression 구현
	한다인	Deeplearning 인터넷 강의 듣기 Tilt Brush 관련 디버깅
날짜		11 월 14 일
3 회차	이준섭	Oculus 와 Script 연동
	유경동	Oculus 에서 연동 가능한 과학상자 UI 구현
	한다인	Deeplearning 인터넷 강의 듣기 Tilt Brush 관련 디버깅
날짜		11 월 15 일
4 회차	이준섭	Oculus 와 Script 연동
	유경동	Oculus 와 과학상자 UI 연동 과학상자 UI 관련하여 디버깅
	한다인	Tensorflow 환경 설정 Tilt Brush 관련 디버깅
날짜		11 월 16 일
5 회차	이준섭	Oculus 와 Script 연동
	유경동	프로그램 중간 통합 Oculus 입력 키 설정
	한다인	Deeplearning 인터넷 강의 듣기 Tilt Brush 관련 디버깅

⑨ 9주차

회차	팀원명	금일 실제 진행한 일
날짜		11 월 19 일
1 회차	이준섭	프로그램 기능 통합 3 차 발표 준비
	유경동	3 차 발표 준비
	한다인	프로젝트 통합 및 3 차 발표
날짜		11 월 20 일

2 회차	이준섭	AddRigidbody 스크립트 최상위 부모에 적용 ScriptSet UI Controller 와 연동
	유경동	Panel 작동 오류 수정 회전 기능 수정 / 보완 Component 색 변경 제작한 Kit Save / Load UI Rigidbody On / Off
	한다인	프로젝트 통합 및 문서 작업
날짜		11 월 21 일
3 회차	이준섭	ScriptUI 키 변경 ScriptUI 및 Frame Collider On / Off
	유경동	UI 입력 버튼 기능 통합
	한다인	프로젝트 통합 후 발생하는 오류 해결
날짜		11 월 22 일
4 회차	이준섭	Script Frame 최소화 옵션 Map 보완
	유경동	Component Save / Load 구현 Kit Save / Load 구현
	한다인	프로젝트 문서 작업
날짜		11 월 23 일
5 회차	이준섭	Map 보완 오류 검토 및 디버깅
	유경동	버그 검출 및 확인
	한다인	프로젝트 문서 작업

C. 개인 개발 일정

2018 년 09 월 27 일	
이준섭	<p>스크래치를 살펴보게 되면 기능 하나하나가 모여서 하나의 동작을 정의할 수 있는데 이 방식을 코드에 적용하기 위해서 1 차적으로 스크래치 분석이 필요하다.</p> <p>그 다음 이를 Code 화 할 수 있도록 C#에서 어느 기능을 써야 동일하게 적용할 수 있을 지 검색을 통해 구체적인 그림을 그려본다.</p> <p>이 후 Unity 에 C#으로 만든 Scratch Code 를 적용하여 원하는 대로 구동이 되는 지 확인해 본다..</p>
유경동	<ol style="list-style-type: none"> 1. 조인트 오브젝트를 만들어 본다. 2. Unity 에서 구현된 rotate c#스크립트를 찾아 본다. 3. 조인트에 적합한 C# 스크립트를 작성하여 적용한다.
한다인	<ol style="list-style-type: none"> 1. Unity 에 내장된 Trail Renderer 를 이용하여 마우스 이동에 따라 선이 그려지게 한다. 2. Trail Renderer 로 그려진 선의 색깔을 변경한다. 3. 선이 Oculus Rift Touch 로 그려질 수 있게 관련 스크립트를 찾는다.
백주성	<ol style="list-style-type: none"> 1.Tilt Brush 관련 팔레트 UI 를 찾아본다. 2.팔레트 UI 를 구상 해 본다. 3.팔레트 UI 를 만들어 본다.
2018 년 09 월 28 일	
이준섭	<p>스크래치 UI 중 가장 우선적으로 동작 부분 기능에 초점을 맞추었다.</p> <p>이를 C#으로 Code 화 하는 과정에서 Delegate 와 Template, 가변인수(Params) 문법을 토대로코드 구현 과정 진행중이다</p>
유경동	<ol style="list-style-type: none"> 1. 휠콜리더를 이용하여 자동차 같은데 쓰일 바퀴를 찾아본다. 2. 자동차 바퀴를 움직일 C#스크립트를 적용할 방법을 찾아보고 적용 해본다. 3.포크레인 바퀴에 무한궤도를 적용할 방법을 찾아본다.
한다인	<ol style="list-style-type: none"> 1. Unity 에 내장된 Trail Renderer 와 Oculus Rift 를 이용해서 선을 그릴 수 있다.

백주성	1.팔레트 UI 의 큰기능을 구현한다.
2018 년 10 월 01 일	
이준섭	1. 스크립트 UI 를 통한 스크립트 동적 배치 2. 스크립트 구조 설계
유경동	1. Mirror 기능과 관련된 자료를 찾아본다. 2. Mirror 기능의 Script 를 구현 해본다.
한다인	1. Unity 에 내장된 LineRenderer 와 MeshRenderer 를 이용해서 Oculus Rift 와 연동하여 그린 선을 지우고, 그려진 그림의 Render 를 On/Off 한다
백주성	1.팔레트 UI 의 큰기능을 구현한다.
2018 년 10 월 02 일	
이준섭	1. 스크립트 변수 값 동적으로 처리하는 부분을 구현할 예정이다.)
유경동	1. Mirror 스크립트 구현(Mirror 간격 조절) 2. Oculus Touch 에서 사용할 과학상자 기능 UI 구현
한다인	1. Unity 에 내장된 LineRenderer 와 MeshRenderer 를 이용해서 Oculus Rift 와 연동하여 그린 선을 지우고, 그려진 그림의 Render 를 On/Off 한다
백주성	1.팔레트 UI 의 큰기능을 구현한다.
2018 년 10 월 04 일	
이준섭	1. 스크립트 변수 값 동적으로 처리하는 부분 구현 2. 스크립트 UI 윤곽선 처리 적용)
유경동	1. Mirror 스크립트 구현(Mirror 간격 조절) 2. 스토어에 있는 UI Tool 분석하고, 과학상자에 사용 할 UI 기본적인 틀 구현하기
한다인	결석

백주성	1.팔레트 UI 의 세부 기능(색상 변경)을 구현한다.
	2018 년 10 월 05 일
이준섭	1. 스크립트 변수 값 동적으로 처리하는 부분 구현 2. 제어 스크립트 동적으로 등록하는 부분 구현 진행 중
유경동	1. 메인 UI 기본적인틀 완료. 2. 세부 UI 구현 및 Script 구현 하기.
한다인	Oculus Rift 와 연동하여 그린 선 Redo/Undo 한다
백주성	결석
	2018 년 10 월 08 일
이준섭	1. 동적 제어 스크립트 제작 구현 2. 스크립트 UI 윤곽선 구현 3. 과학상자 오브젝트 쌓기 자료 조사
유경동	1. BLOCK 을 쌓을 위치를 알아내는 스크립트 구현하기. 2. CUBE 를 원하는 위치에 쌓을 수 있도록 구현 하기.
한다인	1. Oculus Rift touch controller 로 RayCast 만들어서 UI 컨트롤하기
백주성	1.계획 구성
	2018 년 10 월 10 일
이준섭	1. 동적 제어 스크립트 제작 구현 2. 스크립트 UI 윤곽선 구현 3. 과학상자 오브젝트 쌓기 자료 조사
유경동	1. CUBE 를 원하는 위치에 쌓을 수 있도록 구현 하기.
한다인	Line Mesh Objects 를 저장하고 불러오기
백주성	핵심강좌 유니티 챕터 1~6 공부.
	2018 년 10 월 11 일
이준섭	1. 스크립트 동작 정밀화 2. 스크립트 UI 윤곽선 3. 과학상자 스크래치 통합

유경동	<ol style="list-style-type: none"> 1. 빈오브젝트를 리스트로 만들어서 Cube 를 생성하기. 2. 쌓은 큐브를 자동차 바퀴와 연결하여 동작시키기.
한다인	Line Mesh Objects 를 저장하고 불러오기
백주성	핵심강좌 유니티 챕터 7 ~ 12 공부.
2018 년 10 월 12 일	
이준섭	<ol style="list-style-type: none"> 1. 스크립트 동작 정밀화 완료 2. 스크립트 UI 윤곽선 3. 과학상자 스크래치 통합
유경동	<ol style="list-style-type: none"> 1. 만들어진 블록과 자동차 바퀴를 연결하여 움직일수 있게 구현하기. 2. 스크래치 스크립트와 만들어진 오브젝트간의 연결 3. Oculus 연동하기
한다인	1. Line Mesh Objects 를 저장하고 불러오기
백주성	핵심강좌 챕터 13~18 공부.
2018 년 10 월 15 일	
이준섭	<ol style="list-style-type: none"> 1. 오브젝트 크기 확장 2. 과학상자 스크래치 통합 3. 과학상자(+스크래치) Tilt Brush 통합
유경동	<ol style="list-style-type: none"> 1. 만들어진 블록과 자동차 바퀴를 연결하여 움직일수 있게 구현하기. 2. 스크래치 스크립트와 만들어진 오브젝트간의 연결하기 3. Oculus 연동하기
한다인	<ol style="list-style-type: none"> 1. Line Mesh Objects 를 저장하고 불러오기 2. 구현된 코드 Tilt brush 에 적용
백주성	1.핵심강좌 유니티 못다한 공부.
2018 년 10 월 16 일	
이준섭	<ol style="list-style-type: none"> 1. 바퀴 컴포넌트 좌우 대칭 배치 2. 과학상자 스크래치 통합 3. 과학상자(+스크래치) Tilt Brush 통합

	4. 스크래치 관련 발표 자료 준비
유경동	1. 만들어진 블락과 자동차 바퀴를 연결하여 움직일수 있게 구현하기. 2. 스크래치 스크립트와 만들어진 오브젝트간의 연결하기 3. Oculus 연동하기
한다인	1. 1 차 발표를 위한 ppt 를 준비한다. 2. 저장된 파일에 덮어쓰는 것이 가능하게 구현한다.
백주성	1.핵심강좌 유니티 챕터 1~6 공부.
2018 년 10 월 17 일	
이준섭	1 차 발표 준비
유경동	1 차 발표 준비
한다인	1 차 발표 준비
백주성	1.1 차 발표 준비 2.핵심강좌 챕터 7~12 공부.
2018 년 10 월 18 일	
이준섭	1. Mirror 기능 구현 2. 스크립트 윤곽선 구현
유경동	1. UI 를 이용하여 기존에 만들어진 Prefab 출력을 연동하기 2. Oculus 연동하기
한다인	1. 선 그리기 원리를 다시 파악하기 위해서 Procedural Mesh 원리를 공부한다 2. Save/Load 가 Build 하면 되지 않는 문제를 해결하여 다시 구현한다.
백주성	1.핵심강좌 챕터 13~18 공부.
2018 년 10 월 19 일	
이준섭	1. 스크립트 윤곽선 구현 2. 스크립트 자석 구현
유경동	1. UI 를 이용하여 기존에 만들어진 Prefab 출력을 연동하기 2. Oculus 연동하기
한다인	VR 키보드 구현
백주성	1.핵심강좌 유니티 챕터 19~24 공부.
2018 년 10 월 22 일	

2018 년

이준섭	1. 스크립트 회전 시 회전 각 맞추기 2. 스크립트 배치 깊이 맞추기
유경동	1. UI 를 이용하여 기존에 만들어진 Prefab 출력을 연동하기 2. Oculus 연동하기 Oculus Switch Script 분석하여 구현하기
한다인	1. 만들어져 있는 File Browser UI 를 이용해서 file 의 save/load 기능을 구현한다.
백주성	핵심강좌 유니티 챕터 4~5 공부
2018 년 10 월 23 일	
이준섭	1. 스크립트 배치 깊이 맞추기 2. 배치 프레임 이동 3. 배치 스크립트 넣기 / 빼기
유경동	1. Oclues Switch RayCast 를 받아서 원하는 지역에 Prefab 생성하기
한다인	1. 구현된 save/load 기능의 File Browser 와 vr keyboard 를 연동한다. 2. Save/Load 의 기능을 vr 상에서 구현한다
백주성	1.핵심강좌 유니티 챕터 6~7 공부.
2018 년 10 월 24 일	
이준섭	1. 스크립트 배치 깊이 맞추기 2. 배치 프레임 이동 3. 배치 스크립트 넣기 / 빼기
유경동	1. Oculus Switch RayCast 를 받아서 원하는 지역에 Prefab 생성하기
한다인	1. load 기능이 vr 과 연동할 때, 문제가 있었던 점을 보완한다 2. 파일이 제대로 save/load 되는지 확인한다
백주성	1. 핵심강좌 유니티 챕터 8~9 공부.
2018 년 10 월 25 일	
이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기

	2. 스크립트 넣기 / 빼기 3. 스크립트 넣기 / 빼기 시 깊이, 순서 변경 4. 컴포넌트에 스크립트 적용하기
유경동	1. Oculus Switch RayCast 를 받아서 원하는 지역에 Prefab 생성하기 2. Oculus Switch 로 생성된 Object 를 작동시키기
한다인	1. save/load 기능을 테스트해보고 효율적으로 고쳐야 할 점이 있으면 보완하고, 에러가 있으면 고쳐서 코드를 다진다. 2. 팔레트의 UI 를 On/Off 시킨다.
백주성	1.핵심강좌 유니티 챕터 10~11 공부.
	2018 년 10 월 26 일
이준섭	1. 스크립트 넣기 및 순서 변경기능 구현
유경동	1. robocraft 에서 다양한 샘플 모델들을 보고 필요한 오브젝트들을 구상해본다. 2. kit 를 만들 오브젝트들을 제작한다.
한다인	1. 라인을 새로 그릴 때, 바로 전에 그렸던 라인과 선이 연결되는 문제를 해결한다. 2. 파일을 저장하기 위해 브라우저를 띄웠을 때, 저장하지 않고 Cancel 버튼을 눌렀을 경우, 키보드가 사라지지 않고 계속 남아있는 현상을 해결한다. 3. 팔레트에서 아무런 색상을 선택하지 않은 상태에서, 라인을 그렸을 때 색깔이 항상 아무 것도 적용되지 않은 진분홍 색이었던 것을, Default Material 을 적용하여 분홍색이 나오지 않게 한다.
백주성	1.핵심강좌 유니티 챕터 12~13 공부.
	2018 년 10 월 29 일
이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기 2. 스크립트 넣기 / 빼기 3. 스크립트 넣기 / 빼기 시 깊이 맞추기
유경동	1. object Square Pyramid 만들기 2. 자동차 kit 를 만들 오브젝트들을 제작한다.
한다인	1. Load 하면 Undo/Redo 시 발생하는 예외를 처리한다. 2. 전체 라인을 한번에 지운다. 3. 라인을 선택해서 하나씩 지운다.
백주성	1.핵심강좌 유니티 챕터 14~15 공부.
	2018 년 10 월 30 일

이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기 2. 스크립트 넣기 / 빼기 시 깊이 맞추기 3. 제어문 배치
유경동	1. 2 족 보행 kit 를 만들 오브젝트들을 제작한다.
한다인	1. 라인을 선택해서 하나씩 지운다. 2. Keyboard UI 에 계속 text 가 남아있는 현상 해결
2018 년 10 월 31 일	
이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기 2. 스크립트 넣기 / 빼기 시 깊이 맞추기 3. 제어문 스크립트 배치 UI 구현
유경동	1. 2 족 보행 kit 를 만들 오브젝트들을 제작한다. 2. 포크레인과 같은 Joint 를 이용한 키트를 제작한다.
한다인	1.Parlette 오른손에 붙이고 따라다니게 하기 2. Keyboard UI 에 계속 text 가 남아있는 현상 해결
2018 년 11 월 01 일	
이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기 2. 스크립트 넣기 / 빼기 시 깊이 맞추기 3. 제어문 스크립트 배치 UI 구현
유경동	1. 2 족 보행 kit 를 만들 오브젝트들을 제작 및 움직임을 구현 2. 포크레인과 같은 Joint 를 이용한 키트를 제작한다.
한다인	1.Save/Load Build 시 문제나는 것 해결함.
2018 년 11 월 02 일	
이준섭	1. 스크립트 Gizmo 에 따라 깊이 맞추기 2. 스크립트 넣기 / 빼기 시 깊이 맞추기 3. 제어문 스크립트 배치 UI 구현
유경동	Script 에서 UI 에 합칠 때 마우스 키보드를 통해 잘 구현 될 수있도록 기능에 따른 예외처리 및 키보드버튼입력 기능을 최소화한다.
한다인	1. 컬러 팔레트 가져와서 적용하기 2. 텍스처 적용해보기
2018 년 11 월 05 일	

이준섭	2 차 발표 준비
유경동	2 차 발표 준비
한다인	2 차 발표 준비
2018 년 11 월 06 일	
이준섭	1. Rigidbody Script 제작
유경동	1. Object 가 출력기능이 없는 버튼들의 기능 추가한다. 2. Object 출력시 위치가 맞지 않는 것을 코드 수정한다.
한다인	1. 저장한 파일들 한 화면에 불러오기 2. 그려진 선 선택해서 이동 가능하게 하기위해 자료 조사
2018 년 11 월 07 일	
이준섭	1. Rigidbody Script 보완 2. 동력 Script 제작
유경동	기존에 Rigidbody 가 들어간 Object 가 Raycast 를 받지 못해서 Look at 을 통해 각 부위별 조종이 불가능한 것을 수정한다.
한다인	정기 휴일
2018 년 11 월 08 일	
이준섭	1. 동력 Script 제작 2. Wheel Collider 스크립트 연동
유경동	1. UI 에서 Object 를 출력할 때 각도 변경 및 색깔 변경 UI 넣기 2. UI 버튼 및 패널을 좀더 괜찮은 모양으로 변형시키기
한다인	1. 게임 오브젝트 Grab 해서 위치 옮기기
2018 년 11 월 09 일	
이준섭	1. Key Script 제작 2. Frame 삭제 기능 구현
유경동	1. UI 에서 Object 를 출력할 때 각도 변경 및 색깔 변경 UI 넣기 2. UI 버튼 및 패널을 좀더 괜찮은 모양으로 변형시키기 3. Dyno 가 움직일 때 애니메이션처럼 반복적으로 팔이 움직이게 한다.
한다인	1. 게임 오브젝트 Grab 해서 위치 옮기기

	2018 년 11 월 12 일
이준섭	1. Key Script 제작 2. Tensorflow 환경 설정 및 학습
유경동	1. 머신러닝의 개념과 용어 이해하기. 2. TensorFlow 설치 및 기초 코드 작업해보기.
한다인	1. Tensorflow 설치 및 인터넷 강의 듣기
	2018 년 11 월 13 일
이준섭	1. Key Script 제작 2. Oculus 와 Script 연동 3. Tensorflow 학습
유경동	1. Linear Regression 의 개념 2. TensorFlow 로 Linear Regression 구현.
한다인	1. Deeplearning 인터넷 강의 듣기
	2018 년 11 월 14 일
이준섭	1. Oculus 와 Script 연동
유경동	기존에 컴퓨터 마우스 버튼으로 입력받았던 버튼이 Oculus controller 에서 Ray 를 받지 못하여 작동하지 않기 때문에 미리 만들어 놓은 Oculus 용 UI 모양 변경 및 버튼 기능을 다시 입력
한다인	1. Deeplearning 인터넷 강의 듣기 2. Tilt Brush 관련 디버깅
	2018 년 11 월 15 일
이준섭	1. Oculus 와 Script 연동
유경동	PC Version UI 기능들에 프리뷰 및 버튼의 스크립트 기능을 추가하여 작업하고 기존의 패널들의 상호 작용이 Oculus 에 잘 적용 될 수 있게 작업.
한다인	1. Tensorflow 설치 및 인터넷 강의 듣기 2. Tilt Brush 관련 디버깅
	2018 년 11 월 16 일
이준섭	1. Oculus 와 Script 연동

유경동	Oculus 에 사용되는 버튼을 협의하여 설정하고, 겹치는 스크립트가 있는지 확인하고 담당한 자료를 패키지화하여 Tilt Brush 쪽에 불러와 결합시킨다.
한다인	1. Deeplearning 인터넷 강의 듣기 2. Tilt Brush 관련 디버깅
	2018 년 11 월 19 일
이준섭	1. 프로그램 기능 통합 2. 3 차발표 준비
유경동	3 차발표 준비
한다인	프로젝트 통합 및 3 차 발표
	2018 년 11 월 20 일
이준섭	1. Script UI 키 연동 문제 해결 2. Script 입력 키 버튼 On / Off 방식 변경 3. Script Frame 과 대상 연결 시 UI 숨기기 구현
유경동	1. Panel 기능이 제대로 작동되지 않는 경우가 있어 스크립트 확인 후 수정 할 것. 2. 회전시키는 기능을 컨트롤러가 아닌 UI로 작업 할 수 있도록 하기. 3. Object 색깔을 변경할 수 있도록 하기. 4. Object 를 만든것을 Save, Load 를 할 수 있도록 하기 5. UI 로 Rigidbody 를 끄고 킬 수 있도록 하기.
한다인	프로젝트 통합 및 문서 작업
	2018 년 11 월 21 일
이준섭	1. ScriptUI 키 변경 2. ScriptUI 및 Frame Collider On / Off
유경동	기존에 과학사장 UI 를 선택할 때 오쿨러스의 One 버튼을 입력받아 출력했었지만 마지막 레이를 받은 지점이 버튼을 클릭하는 버그를 발견하여 기존의 버튼을 입력받는 방식에서 상위 UI 버튼을 만들어서 과학상자 UI 를 출력할 수 있게 수정.
한다인	1. 프로젝트 통합 후 발생하는 오류 해결
	2018 년 11 월 22 일
이준섭	1. Script Frame 최소화 옵션

	2. Map 보완
유경동	작업시 만들던 작업을 저장하고 불러올 수 있는 Saver 와 Load 및 잘못 만들었을 때 필요한 삭제 기능 구현.
한다인	1. 프로젝트 문서 작업
	2018 년 11 월 23 일
이준섭	1. Map 보완 2. 오류 검토 및 디버깅
유경동	Tilt Brush 와 과학상자를 합치면서 겹으로 보이는 버그는 수정했지만 추가적인 버그나 제대로 작동되지 않는 부분이 없는지 테스트.
한다인	1. 프로젝트 문서 작업

4. 프로젝트 개발 내용

A. 프로젝트 배경 지식/기술/알고리즘

I. Development Tool

1. Unity

저사양/소규모 게임의 개발에 적합한 게임 엔진으로 2005년 6월 8일에 처음 발표되었다. 초창기에는 Windows, Mac OS X 같은 PC 기반 플랫폼만을 지원했으나 기술이 발전함에 따라 Android, iOS와 같은 모바일 플랫폼 뿐 아니라 PS3, XBOX 360, Wii와 같은 콘솔 게임기에까지 지원하는 플랫폼이 확장되었다. 최근에는 VR 기기와의 연동도 가능해졌다. 초창기에는 저사양 / 소규모의 3D 혹은 2D 타겟의 웹 미디어 제작툴이었으나 점차 게임 엔진으로 변모하였다. 유니티의 장점은 첫째, GUI가 직관적이기에 툴 사용이 익숙하지 않은 사람도 편하게 사용할 수 있다 둘째, 플랫폼 빌드가 간단하게 이뤄진다. 셋째, 초보 개발자들이 개발하기 쉽도록 하고자 애셋스토어를 제공해주어 플리마켓처럼 유니티 사용자들이 다양한 콘텐츠를 다운받아 이용할 수 있다. 넷째, 라이선스 비용이 저렴하여 인디 개발자나 소규모 콘텐츠 제작자들이 이용개발하기 용이한 환경이다. 기본 스크립트 작성을 위해서 C#을 기본 프로그래밍 언어로 사용하며, 이를 유니티와 결합하고자 Mono 프레임워크 기반에서 코드가 동작하도록 구성되어있다

2. Visual Studio 2017

마이크로소프트에서 개발한 통합 개발 환경(IDE)로 윈도우에서 동작하는 거의 대부분의 프로그램들을 만들 수 있다. 개발 환경을 전체 설치할 경우 80GB에 육박하는 용량을 감당해야 하지만, 원하는 기능만 최소로 설치할 경우 많은 용량을 절약할 수 있다. C++ / C# 영역에서는 타 IDE에서 따라올 수 없는 개발 환경을 가지고 있으며, 그러한 특징으로 인해 해당 언어 개발 뿐 아니라 해당 언어 기반의 환경이라면 최적화가 잘 되어 있어 툴에 사전지식이 부족하더라도 쉽게 개발 할 수 있다.

II. Hardware

1. Oculus Rift

오쿨러스 리프트(Oculus Rift) 또는 리프트(Rift)는 오쿨러스 VR사에서 개발한 가상 현실(VR) 머리장착디스플레이(HMD, Head Mounted Display)이다. 2016년 3월 28일에 처음 출시되었으며, 창시자는 팔머 럭키이다. 오쿨러스 리프트는 기존의 HMD와 구현

방식이 다른데, 이는 기존의 가정용 HMD가 눈 앞에 가까이 디스플레이를 놓음으로써 시야에서 크게 보이게 하는 정비율 출력 기기인 반면, 오쿨러스 리프트는 렌즈를 이용해 시야각 전체를 커버하고 이로 인한 왜곡은 출력보정으로 변환하는 혁신적인 방식을 선택했기 때문이다. 이전의 HMD는, 초소형의 LCD/OLED 패널을 눈 앞에 배치하거나, 비슷한 크기의 상을 눈 앞에 뿌려주는 방식으로 구현되었다. 이 방식은 2개의 패널을 이용하기 때문에 가격이 올라가고, 패널이 매우 작기에 패널의 dpi가 매우 높아야 하며, 안구에 상을 왜곡없이 맺히도록 하기위해 매우 높은 가격의 정교한 렌즈가 필요하다. 이는 고화질의 영상기기로서 다양한 콘텐츠 출력기기와의 호환성을 고려한 것이다. 실제로 보면 모니터 화면이 어두운 터널 저편에 있는 듯한 느낌이다. 양안시를 통한 입체감이나 집중효과는 있으나 쌍안경으로 핸드폰을 보는 느낌에 가까워서 현실감은 떨어진다.

그러나, 오쿨러스 리프트는 상대적으로 대형의 6인치 패널 1개를 좌우로 나누어 각 안구용으로 사용하며 가격이 비싼 복잡하고 정교한 렌즈 대신, 컴퓨터 측에서 두 눈의 시야에 맞추어 어안 렌더링을 한후 합쳐서 패널로 출력한 뒤 HMD 본체에서 좌/우 각각의 화면을 따로따로 각각의 볼록렌즈로 좌/우 안구에 적절한 상이 맺히도록 한다. 1개의 대형, 저dpi 패널과 극히 단순한 볼록렌즈만을 이용하기 때문에 가격면이나 구현 면에서 유리하며, 넓은 시야각을 통해 현실감이 증대된다. 이는 크게 고화질이 필요치 않으면서 높은 현장감을 가지는 가상현실을 저가의 기기에서 구현하기 위한 것이다.

결과적으로 오쿨러스 리프트는 기존의 HMD보다 훨씬 넓은 시야각을 자랑하면서도 가격은 저렴하게 책정되었다.[6]기존의 HMD가 마치 영화관에서처럼 앞에 커다란 스크린이 있는 것 처럼 보이는 반면 오쿨러스 리프트는 정말로 게임 내 1인칭 시점으로 들어간 느낌을 주게 된다.

개발자 버전인 DK1, DK2는 1개의 디스플레이를 반으로 쪼개셨으나 소비자 버전인 CV1 부터는 2개의 디스플레이를 사용한다. 각 눈당 1080 x 1200 해상도를 가진다.

요구하는 시스템 사양은 다음과 같다.

	최소 사양	권장 사양
CPU	Intel 코어 i3-6100 AMD FX 4350 또는 그 이상	Intel 코어 i5-4590 AMD Ryzen 1500X 또는 그 이상
VGA	NVIDIA 지포스 GTX 960 또는 그 이상	NVIDIA 지포스 GTX 970 AMD 라데온 R9 290 또는 그 이상
OS	Windows 8 또는 이후 버전	Windows 7 SP1 64비트 또는 이후 버전
RAM	8 GB 또는 그 이상	
기타	HDMI 1.3, USB 3.0 ×1, USB 2.0 ×2	HDMI 1.3, USB 3.0 ×3, USB 2.0 ×1



Oculus rift 본체
(HMD, 입체 서라운드 헤드폰, 자이로 트래킹장치 내장)



트래킹 센서
(위치 이동 트래킹)



오culus 터치
(자이로 장치 내장, 위치이동 및 조작)

2. 그래픽 카드(GeForce 960 mini)

Oculus Rift와 연동하기 위해서 최소 사양으로 필요한 VGA 장치로 최소 사양으로 갖춰져야 Oculus 구동을 위한 애플리케이션 실행이 가능하다.



칩셋 사양은 다음과 같다.

칩셋 사양

칩셋 제조사	NVIDIA	GPU 제조 공정	28nm
칩셋 그룹	지포스 GTX 10xx	세부칩셋	지포스 GTX 960
코어 클럭	1190MHz (Boost 1253MHz)	쿠다 프로세서	1024개
인터페이스	PCI-Express 3.0 x16		

메모리 사양

메모리 종류	GDDR5(DDR5)	메모리 클럭	7010MHz
메모리 용량	2GB	메모리 버스	128-bit

제품 외형

카드 크기 (가로*세로)	170mm * 106mm	두께	40mm
카드 규격	일반	코어 냉각방식	방열판+클러

영상 출력 지원 사양

DVI 출력포트	1개	HDMI	1개
DisplayPort	3개	최대 모니터 지원	4대

부가 기능

Dual-Link DVI	○	4K 해상도 지원	○
HDMI 2.0 지원	○	D-SUB 지원	DVI to D-SUB젠더

전력 관련

최대 사용 전력	147W	보조 전원 연결	6핀 x 1개
권장 파워 용량	500W		

III. 개발 언어

1. C

Unity에서 GameObject들이 기능 수행을 위해서 필요한 스크립트 파일을 작성할 때 사용하는 언어이다. C++과 같이 객체 지향 프로그래밍 언어로 C++ 의 객체지향 요소를 더욱 살려 구성되어 있으며, C++만큼이나 자바와 큰 유사성을 띤다. 닷넷 프레임워크의 한 요소로 만들어져 있다. Unity에서는 외부 Tool과 결합하여 C# 언어를 작성하며 주로 Visual Studio와 결합이 된다. 이 때 C# 컴파일러로써 Mono를 사용하며 Mono 기반에서 Script를 작성해주게 된다. 이 때 작성하는 C# 스크립트 파일명은 해당 스크립트 내에 있는 클래스명과 동일하여야 하며, 클래스는 MonoBehaviour Class를 상속받아야 Unity GameObject에 적용 가능한 클래스로 그 기능을 수행할 수 있다.

B. 프로젝트 상세 개발 내용

I. 기능정의 리스트

- Tilt Brush

대분류	중간항목	상세항목	항목번호	설명	담당자	순위
Tilt Brush	설치	Unity 설치	TB-ST-01	Tilt Brush 기능을 만들 Unity Application 을 설치한다.	유경동	1
		Oculus Rift 설치	TB-ST-02	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 소프트웨어를 설치한다.	유경동	1
		그래픽 카드 설치	TB-ST-03	Oculus Rift 를 구동하기 위해 사용 PC 에 그래픽카드를 설치한다.	유경동	1
		Oculus Rift 장비 설치	TB-ST-04	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 장비를 PC 에 연결한다.	유경동	1
		Oculus Rift 센서 연동 확인	TB-ST-05	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 센서와 조이스틱이 인식이 정상적으로 작동하는지 확인한다.	유경동	1
	그리기	선 그리기	TB-DR-01	Trail Renderer 를 사용하여 선을 표현한다.	이준섭 한다인 백주성	1
		색 변경하기	TB-DR-02	Trail Renderer 로 그려진 색을 변경한다.	이준섭 한다인 백주성	1
		선 지우기	TB-DR-03	Trail Renderer 로 그린 선을 지운다.	이준섭 한다인 백주성	1
		두께 조절하기	TB-DR-04	Trail Renderer 로 그릴 선의 두께를 조절한다.	이준섭 한다인 백주성	1
		선의 깊이 조절하기	TB-DR-05	Trail Renderer 로 그릴 선의 깊이를 조절한다.	이준섭 한다인 백주성	1
		실행 취소 / 되돌리기	TB-DR-06	Trail Renderer 로 그린 것을 취소하거나 되돌릴 수 있다.	이준섭 한다인 백주성	2

	팔레트 UI 작성하기	TB-DR-07	팔레트 UI 를 만든다.	이준섭 한다인 백주성	1
	팔레트 UI On / Off	TB-DR-08	팔레트 UI 활성화 / 비활성화를 가능하게 한다.	이준섭 한다인 백주성	1
	그림 Rendering On / Off	TB-DR-09	그림 Rendering 활성화 / 비활성화를 가능하게 한다.	이준섭 한다인 백주성	1
	팔레트 UI 기능 구현	TB-DR-10	팔레트 UI 에서 옵션 선택 시 기능을 수행할 수 있도록 해준다.	한다인, 백주성	2
	그림 저장하기	TB-DR-11	Trail Renderer 로 그린 그림을 저장할 수 있다.	이준섭 한다인 백주성	2
	그림 불러오기	TB-DR-12	저장한 그림을 불러 올 수 있다.	이준섭 한다인 백주성	2
	VR Keyboard 구현	TB-DR-13	Oculus 에서 사용할 입력 Keyboard 를 출력하고 사용할 수 있다	이준섭 한다인	2
	File Browser 구현	TB-DR-14	만들어지는 Data 정보를 가져올 수 있도록 파일 저장 / 불러오기 UI 를 구현한다	한다인	2
	선택 선 지우기	TB-DR-15	Controller 가 선택하는 선을 지울 수 있다.	이준섭 한다인	2
제어	센서 인식 제어	TB-CO-01	센서 인식 제어를 가능하게 한다.	이준섭 한다인 백주성	1

● 과학 상자

대분류	중간항목	상세항목	항목번호	설명	담당자	순위
과학상자	설치	Unity 설치	SB-ST-01	과학상자 기능을 만들 Unity Application 을 설치한다.	유경동	1
		Oculus Rift 설치	SB-ST-02	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 소프트웨어를 설치한다.	유경동	1
		그래픽 카드 설치	SB-ST-03	Oculus Rift 를 구동하기 위해 사용 PC 에 그래픽카드를 설치한다.	유경동	1
		Oculus Rift 장비 설치	SB-ST-04	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 장비를 PC 에 연결한다.	유경동	1
		Oculus Rift 센서 연동 확인	SB-ST-05	Unity 로 만든 Application 을 실행하기 위해서 Oculus Rift 센서와 조이스틱이 인식이 정상적으로 작동하는지 확인한다.	유경동	1
	Object동작	바퀴회전	SB-AC-01	자동차 바퀴 오브젝트를 만들어 어느 오브젝트에 붙어도 바퀴가 회전하게 할 수 있으며 스크립트를 통해 조건 제어가 가능하게 한다.	유경동	1
		회전 조인트	SB-AC-02	두 오브젝트가 만나는 곳에 조인트 오브젝트를 넣어 조인트의 00방향으로 회전할 수 있으며 스크립트를 통해 조건 제어가 가능하게 한다.	이준섭 하홍복 유경동	1
		Base 회전객체	SB-AC-03	Base 객체 위에 있는 모든 오브젝트를 회전 시키며 스크립트를 통해 회전 속도 및 회전 방향 조건 제어가 가능하게 한다.	이준섭 하홍복 유경동	1
		Kit 및 Component 움직이기	SB-AC-04	Kit나 Component 들이 Script에 의해 움직일 수 있다	이준섭 유경동	2
	스크래쳐	스크립트 컴포넌트 정의	SB-SC-01	스크립트 제작을 위한 기본 스크립트 구성재료를 정의한다.	이준섭	1
		스크립트 컴포넌트 삭제	SB-SC-02	등록한 기본 스크립트 구성 재료를 삭제한다.	이준섭	3
		스크립트 컴포넌트 편집	SB-SC-03	등록한 기본 스크립트 구성 재료를 수정한다.	이준섭	2

		스크립트 제작	SB-SC-04	스크립트 컴포넌트를 조합하여 스크립트를 만든다.	이준섭	1
		스크립트 편집	SB-SC-05	만들어진 스크립트를 재배포하거나 내용을 수정한다.	이준섭	2
		스크립트 삭제	SB-SC-06	만들어진 스크립트를 삭제한다.	이준섭	3
		제어함수 제작	SB-SC-07	스크립트를 조합하여 Object의 동작을 제어할 제어함수를 만든다.	이준섭 하홍복	1
		제어함수 편집	SB-SC-08	제작한 제어함수를 편집한다.	이준섭 하홍복	2
		제어함수 삭제	SB-SC-09	제작한 제어함수를 삭제한다.	이준섭	3
		제어함수 등록	SB-SC-10	제작한 제어함수를 Object 움직임으로 정의한다.	이준섭 유경동	1
		스크립트 스냅	SB-SC-11	화면 상에 스크립트UI들을 붙게한다.	하홍복	1
		스크립트 묶기	SB-SC-12	화면 상에 스크립트 UI가 하나의 제어함수로 묶이고 순서를 결정하기 위해 부모 - 자식 관계로 만들어준다.	하홍복	1
		제어함수 길이 확장	SB-SC-13	스크립트UI가 붙어있을 때 UI의 최대 길이만큼 UI Set이 연장되도록 해준다.	하홍복	2
		스크립트 UI 아웃라인	SB-SC-14	스크립트UI끼리 접하게 할 경우, 접합이 가능한 UI에 접합가능 바깥 선을 표시해준다.	하홍복	1
		스크립트 자석 구현	SB-SC-15	스크립트끼리 가까이 근접하면 붙는 효과를 보여줄 수 있다	이준섭	2
		스크립트 깊이 설정	SB-SC-16	스크립트 제어 기능을 사용하기 위해 깊이를 설정해준다	이준섭	2
		스크립트 배치된 UI 위치 조정	SB-SC-17	스크립트가 붙을 때 마다 위치를 재조정하여 준다.	이준섭	2
	Object 만들기	간단한 Object 불러오기	SB-OB-01	Cube, cylinder, capsule, sphere, plane, Quad 등의 Object를 불러올 수 있다.	이준섭 하홍복 유경동	1
		Object 회전 및 축소, 늘리기	SB-OB-02	불러온 object를 늘리거나, 축소 회전 시킬 수 있다.	이준섭 유경동	3

		Mirror기능	SB-OB-03	Object를 생성하는 방향에 양쪽 대칭으로 블록이 생성 가능하게 한다.	이준섭 하홍복 유경동	1
		저장하기/ 불러오기	SB-OB-04	현재 object의 작업 환경을 그대로 저장하거나 불러올 수 있게 한다.	이준섭 하홍복 유경동	1
		예시 오브젝트 만들기	SB-OB-05	자동차, 포크레인 등 만들 수 있는 키트들의 Object를 제공한다.	이준섭 유경동	3
		Object UI 투명도	SB-OB-06	선택된 Object UI가 카메라 시점에서 가려질 때 그 Object가 보이도록 주변부 Object를 투명하게 해준다.	하홍복	1
		Object UI 아웃라인	SB-OB-07	Object를 새로 배치하거나 배치된 Object를 움직일 때 접합할 수 있는 Object 근처에 갔을 때 배치가 가능한 Object에 주변 선을 표시해준다	하홍복	2
		Object Prefab 제작	SB-OB-08	만든 Object를 Kit로 만들어 줄 수 있다	유경동	2
		Oculus Ray로 Object 배치	SB-OB-09	Ray가 쏘는 위치에 Object를 배치할 수 있다	유경동	2
		Oculus와 과학상자 UI 연동	SB-OB-10	과학상자 UI가 선택하는 기능을 Oculus 상에서 사용할 수 있다	유경동	2
		Object 색 변경	SB-OB-11	Object 색상을 변경할 수 있다	유경동	3
	제어	센서 인식 제어	SB-CO-01	센서 인식 제어를 가능하게 한다.	유경동	1

II. 주요 소스코드

● Tilt Brush 선 그리기 코드

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MeshLineRenderer : MonoBehaviour {

    public Material lmat;
    private Mesh ml;
    private Vector3 s;
    private float lineSize = .1f;
    private bool firstQuad = true;

    void Start()
    {
        ml = GetComponent<MeshFilter>().mesh;
        GetComponent<MeshRenderer>().material = lmat;
    }
    //선의 굵기 정하기
    public void setWidth(float width)
    {
        lineSize = width;
    }
    //그림을 그리면서 Quad Mesh의 Vertex 위치가 이동하므로, 이동한 점에 대한 Vector 값을 전달
    public void AddPoint(Vector3 point)
    {
        if (s != Vector3.zero)
        {
            AddLine(ml, MakeQuad(s, point, lineSize, firstQuad));
        }
    }
}
```

```
        firstQuad = false;
    }

    s = point;
}
//하나의 Quad Mesh를 만들어주기 위한 메소드
Vector3[] MakeQuad(Vector3 s, Vector3 e, float w, bool all)
{
    w = w / 2;

    Vector3[] q;
    if (all)
    {
        q = new Vector3[4];
    }
    else
    {
        q = new Vector3[2];
    }

    Vector3 n = Vector3.Cross(s, e);
    Vector3 l = Vector3.Cross(n, e - s);
    l.Normalize();

    if (all)
    {
        q[0] = transform.InverseTransformPoint(s + l * w);
        q[1] = transform.InverseTransformPoint(s + l * -w);
        q[2] = transform.InverseTransformPoint(e + l * w);
        q[3] = transform.InverseTransformPoint(e + l * -w);
    }
    else
    {

```



```
        q[0] = transform.InverseTransformPoint(s + l * w);
        q[1] = transform.InverseTransformPoint(s + l * -w);
    }
    return q;
}

//계속해서 Quad를 늘려나가 하나의 선처럼 보이기 위한 메소드
void AddLine(Mesh m, Vector3[] quad)
{
    int vl = m.vertices.Length;

    Vector3[] vs = m.vertices;
    vs = resizeVertices(vs, 2 * quad.Length);

    for (int i = 0; i < 2 * quad.Length; i += 2)
    {
        vs[vl + i] = quad[i / 2];
        vs[vl + i + 1] = quad[i / 2];
    }

    Vector2[] uvs = m.uv;
    uvs = resizeUVs(uvs, 2 * quad.Length);

    if (quad.Length == 4)
    {
        uvs[vl] = Vector2.zero;
        uvs[vl + 1] = Vector2.zero;
        uvs[vl + 2] = Vector2.right;
        uvs[vl + 3] = Vector2.right;
        uvs[vl + 4] = Vector2.up;
        uvs[vl + 5] = Vector2.up;
        uvs[vl + 6] = Vector2.one;
        uvs[vl + 7] = Vector2.one;
    }
}
```

```
else
{
    if (vl % 8 == 0)
    {
        uvs[vl] = Vector2.zero;
        uvs[vl + 1] = Vector2.zero;
        uvs[vl + 2] = Vector2.right;
        uvs[vl + 3] = Vector2.right;

    }
    else
    {
        uvs[vl] = Vector2.up;
        uvs[vl + 1] = Vector2.up;
        uvs[vl + 2] = Vector2.one;
        uvs[vl + 3] = Vector2.one;
    }
}

int tl = m.triangles.Length;

int[] ts = m.triangles;
ts = resizeTriangles(ts, 12);

if (quad.Length == 2)
{
    vl -= 4;
}

// front-facing quad
ts[tl] = vl;
ts[tl + 1] = vl + 2;
ts[tl + 2] = vl + 4;
```

```
ts[tl + 3] = vl + 2;
ts[tl + 4] = vl + 6;
ts[tl + 5] = vl + 4;

// back-facing quad
ts[tl + 6] = vl + 5;
ts[tl + 7] = vl + 3;
ts[tl + 8] = vl + 1;

ts[tl + 9] = vl + 5;
ts[tl + 10] = vl + 7;
ts[tl + 11] = vl + 3;

m.vertices = vs;
m.uv = uvs;
m.triangles = ts;
m.RecalculateBounds();
m.RecalculateNormals();
}

Vector3[] resizeVertices(Vector3[] ovs, int ns)
{
    Vector3[] nvs = new Vector3[ovs.Length + ns];
    for (int i = 0; i < ovs.Length; i++)
    {
        nvs[i] = ovs[i];
    }

    return nvs;
}

Vector2[] resizeUVs(Vector2[] uvs, int ns)
```

```

    {
        Vector2[] nvs = new Vector2[Uvs.Length + ns];
        for (int i = 0; i < Uvs.Length; i++)
        {
            nvs[i] = Uvs[i];
        }

        return nvs;
    }

    int[] resizeTriangles(int[] ovs, int ns)
    {
        int[] nvs = new int[Ovs.Length + ns];
        for (int i = 0; i < Ovs.Length; i++)
        {
            nvs[i] = Ovs[i];
        }

        return nvs;
    }
}

```

● Tilt Brush와 Oculus 연동 코드

//Oculus Touch 컨트롤러의 Trigger 부분을 누를시에 계속해서 선을 그리는 Method를 호출한다

```

        if(OVRInput.GetDown(OVRInput.Button.PrimaryIndexTrigger,
OVRInput.Controller.RTouch))
        {
            GameObject go = new GameObject();
            go.transform.SetParent(GroupOfLines.transform);
            go.tag = "Lines";

```

```
go.AddComponent<MeshFilter>();
go.AddComponent<MeshRenderer>();
currLine = go.AddComponent<MeshLineRenderer>();

//Change color
Change_Color();
//Change width
Change_WidthValue();

numClicks = 0;
Linelist.Add(go);
newline_idx++;
}
else if (OVRInput.Get(OVRInput.Axis1D.SecondaryIndexTrigger,
OVRInput.Controller.Touch) > 0.1f)
{
    //Change width
    Change_WidthValue();
    //Change color
    Change_Color();
    currLine.AddPoint(transform.position);
    numClicks++;
}
else if (OVRInput.GetUp(OVRInput.Button.PrimaryIndexTrigger,
OVRInput.Controller.RTouch))
{
    currLine.AddPoint(Vector3.zero);
}
```

● 스크립트 배치 코드

```

public class DragNewScript : MonoBehaviour
{

    public void StartMove()
    {
        StartCoroutine("MoveNewScript");
    }

    IEnumerator MoveNewScript()
    {
        OVRPointerVisualizer          ovrPointer          =
GameObject.FindGameObjectWithTag("LineRender").GetComponent
<OVRPointerVisualizer> ();
        Transform gazeTr = ovrPointer.gazePointer;
        Camera          camera          =
GameObject.Find("CenterEyeAnchor").GetComponent<Camera> ();
        Debug.Log("MoveNewScript");
        FrameTarget          frameTarget          =
GameObject.Find("FrameManager").GetComponent<FrameTarget> ();
        Vector3 initPosition = transform.position;
        Quaternion initRotation = transform.rotation;
        Vector3          scrSpace          =
camera.WorldToScreenPoint(transform.position);
        Vector3 offset = transform.position
-          camera.ScreenToWorldPoint(new
Vector3(Input.mousePosition.x, Input.mousePosition.y, scrSpace.z));

        GameObject newObject = Instantiate(this.gameObject) as
GameObject;
    }
}

```

```

        Destroy(newObject.GetComponent<DragNewScript>());
        newObject.AddComponent<ScriptScaleInfo>();
        newObject.name = this.gameObject.name;
        newObject.transform.position = this.transform.position;
        newObject.transform.rotation = this.transform.rotation;

        // 배치할 스크립트에 붙을 대상 Setting
        frameTarget.SetTargetScript(newObject);

        ArrangeScript arrangeScript =
        GameObject.Find("ArrangeManager").GetComponent<ArrangeScript>();
        while (OVRInput.Get(OVRInput.Button.PrimaryIndexTrigger,
        OVRInput.Controller.LTouch))
        {
            gazeTr = ovrPointer.gazePointer;
            Vector3 curScreenSpace = new
            Vector3(Input.mousePosition.x, Input.mousePosition.y, scrSpace.z);
            Vector3 curPosition =
            camera.ScreenToWorldPoint(curScreenSpace) + offset;
            newObject.transform.position = gazeTr.position;
            Debug.Log(gazeTr.position);
            yield return null;
        }
        frameTarget.FreeTargetScript();

        // ArrangeAdd함수 호출
        CallArrangeAdd(arrangeScript, newObject);
    }

```

```
private void CallArrangeAdd(ArrangeScript arrangeScript,
    GameObject newObject)
{
    // 스크립트 형태에 따라 다른 추가방식
    IStateScriptSet stateScript =
this.GetComponent<IStateScriptSet>();
    IMotionScriptSet motionScript =
this.GetComponent<IMotionScriptSet>();
    IEventScriptSet eventScript =
this.GetComponent<IEventScriptSet>();

    if (stateScript != null)
        arrangeScript.MakeStateFrame(newObject);
    if (motionScript != null)
        arrangeScript.ArrangeAfterAdd(newObject);
    if (eventScript != null)
        arrangeScript.EventScriptAdd(newObject);
}
}
```


● 스크립트 UI 상속 구조

```
namespace ScratchScripts
{
    public delegate bool Script();
    public interface IMotionScriptSet
    {
        bool ScriptFunction();
        void SetTarget(GameObject target);
        void SetDepth(int depth);
        int GetDepth();
        bool IsInFrame();
    }

    public interface IStateScriptSet
    {
        void SetTarget(GameObject target);
        void CancelTarget();
    }

    public interface IEventScriptSet
    {
        void SetTarget(GameObject target);
        bool IsKeyInput();
    }

    public interface IValueScriptSet
    {
        void SetValue(string key, string value);
    }
}
```

```
namespace ActionScripts
{

    public abstract class ActionSet : MonoBehaviour, IMotionScriptSet ,
    IValueScriptSet
    {
        protected GameObject targetObject;
        protected Hashtable valueTable;
        protected bool isOP;
        protected int depth;
        public Hashtable ValueTable
        {
            get { return this.valueTable; }
            set { this.valueTable = value; }
        }
        public bool ScriptFunction()
        {
            return Action();
        }
        abstract protected bool Action();

        public void SetValue(string key, string value)
        {
            valueTable[key] = (float)Convert.ToDouble(value);
        }
        public void SetTarget(GameObject target)
        {
            targetObject = target;
        }
    }
}
```

```
    }  
    public void SetDepth(int depth)  
    {  
        this.depth = depth;  
    }  
    public int GetDepth()  
    {  
        return this.depth;  
    }  
    public bool IsInFrame()  
    {  
        return false;  
    }  
}  
  
}  
  
namespace ControlScripts  
{  
    public abstract class ControlSet : MonoBehaviour, IMotionScriptSet,  
    IValueScriptSet  
    {  
  
        protected GameObject targetObject;  
        protected Hashtable valueTable;  
        protected int depth;  
  
        public Hashtable ValueTable  
        {  

```

```
        get { return this.valueTable; }
        set { this.valueTable = value; }
    }

    public bool ScriptFunction()
    {
        return Control();
    }
    abstract protected bool Control();

    public void SetValue(string key, string value)
    {
        valueTable[key] = Convert.ToInt32(value);
    }

    virtual public void SetTarget(GameObject target)
    {
        targetObject = target;
    }
    virtual public void SetDepth(int depth)
    {
        this.depth = depth;
    }
    virtual public int GetDepth()
    {
        return this.depth;
    }
    virtual public bool IsInFrame()
    {
        return false;
    }
```

```
    }  
}  
  
public abstract class ParentControlSet : ControlSet  
{  
    protected bool isKeyDown;  
    protected PlayScript playScript;  
    public bool IsKeyDown  
    {  
        get { return this.isKeyDown; }  
        set { this.isKeyDown = value; }  
    }  
    public PlayScript PlayScript  
    {  
        get { return this.playScript; }  
        set { this.playScript = value; }  
    }  
    protected void ParentAwake()  
    {  
  
    }  
    protected void ParentStart()  
    {  
        Debug.Log(this.name + " : ParentStart()");  
    }  
  
    public void SetScriptTarget(GameObject target)  
    {  
        playScript.ScriptFrame.ScriptTarget = target;  
    }  
}
```

```
        override public bool IsInFrame()
        {
            return true;
        }

        override public void SetDepth(int depth)
        {
            this.depth = depth;
            ((ScriptFrame)playScript.ScriptFrame).Depth = depth + 1;
        }

        override public void SetTarget(GameObject target)
        {
            targetObject = target;
            if (playScript.ScriptFrame != null)
                playScript.ScriptFrame.ChangeTarget(target);
        }

        public void SetScriptFrame(GameObject frame)
        {
            this.playScript = frame.GetComponent<SubScript>();
            if (frame.GetComponent<ScriptFrame>() == null)
                frame.AddComponent<ScriptFrame>();
            playScript.ScriptFrame = frame.GetComponent<ScriptFrame>();
        }
    }
}
```

namespace PowerSet

```
{  
    public abstract class PowerScript : MonoBehaviour, IStateScriptSet  
    {  
        protected GameObject targetObject;  
        abstract public void SetTarget(GameObject target);  
        abstract public void CancelTarget();  
    }  
}  
  
namespace EventScripts  
{  
    public abstract class EventSet : MonoBehaviour, IEventScriptSet  
    {  
        protected GameObject targetObject;  
        abstract public void SetTarget(GameObject target);  
        abstract public bool IsKeyInput();  
    }  
}
```

● 물체 움직이는 기능을 담당하는 코드

```
public class MoveObject : ActionSet
{
    private float totalMove;
    private float endMove;
    void Awake()
    {

        valueTable = new Hashtable();
        valueTable["X"] = (float).0;
        valueTable["Y"] = (float).0;
        valueTable["Z"] = (float).0;
        isOP = false;
        totalMove = 0;
        endMove = 0;
        depth = 0;
    }

    override protected bool Action()
    {

        Vector3 startPosition = targetObject.transform.position;
        Debug.Log(isOP);
        if (!isOP)
        {
            Vector3 targetPosition = new Vector3(startPosition.x +
(float)valueTable["X"],
startPosition.y +
(float)valueTable["Y"],
startPosition.z +
```



```
(float)valueTable["Z"]);
    endMove = Vector3.Distance(targetPosition, startPosition);
    Debug.Log("endMove : " + endMove);
    isOP = !isOP;
}

    targetObject.transform.Translate((float)1.0 * (float)valueTable["X"] *
Time.deltaTime,
                                     (float)1.0 * (float)valueTable["Y"] *
Time.deltaTime,
                                     (float)1.0 * (float)valueTable["Z"] *
Time.deltaTime);
    Vector3 MovePosition = targetObject.transform.position;
    totalMove += Vector3.Distance(MovePosition, startPosition);

    startPosition = targetObject.transform.position;

    // 길이만큼 이동 시 반복 종료
    if (totalMove >= endMove)
    {
        //Debug.Log("Total Move : End Move " + totalMove + " / " +
endMove);
        isOP = !isOP;
        totalMove = 0;
        endMove = 0;
        return true;
    }
    return false;
}
```

```
}
```

- 스크립트 입력 키 지정 스크립트

```
public class KeyInputScript : EventSet, IValueScriptSet{

    private string keyName;
    public string KeyName
    {
        get { return this.keyName; }
    }
    void Awake()
    {
        keyName = "return";
    }

    public void SetValue(string key, string value)
    {
        if(key.Equals("KeyName"))
            keyName = value.ToLower();
    }
    override public void SetTarget(GameObject target)
    {
        targetObject = target;
    }
    override public bool IsKeyInput()
    {
        return true;
    }
}
```

```
}
```

- 물체에 Rigidbody 추가하는 코드

```
public class AddRigidBody : PowerScript {

    private float mass;

    void Awake()
    {
        mass = 1500f;
    }
    override public void SetTarget(GameObject target)
    {

        if(targetObject != null && !targetObject.Equals(target))
            DestroyComponent();
        targetObject = target;
        CreateComponent();
    }
    override public void CancelTarget()
    {
        DestroyComponent();
        targetObject = null;
    }
    private void CreateComponent()
    {
        // 최상위 부모에 Rigidbody 없으면 추가
        if (this.targetObject != null &&
```

```
this.targetObject.transform.root.GetComponent<Rigidbody>() == null)
{

this.targetObject.transform.root.gameObject.AddComponent<Rigidbody>().mass
= this.mass;

this.targetObject.transform.root.GetComponent<Rigidbody>().constraints =
RigidbodyConstraints.FreezeAll;

}
}
private void DestroyComponent()
{
    // 최상위 부모에 Rigidbody 있으면 삭제
    if (this.targetObject != null &&
this.targetObject.transform.root.GetComponent<Rigidbody>() != null)

Destroy(this.targetObject.transform.root.GetComponent<Rigidbody>());
}
}
```

● 엔진 추가 코드

```
public class AddEngineMotor : PowerScript, IValueScriptSet{

    private Hashtable valueTable;
    public Hashtable ValueTable
    {
        get { return this.valueTable; }
        set { this.valueTable = value; }
    }

    void Awake()
    {
        valueTable = new Hashtable();
        valueTable["MotorTorque"] = 0f;
    }

    override public void SetTarget(GameObject target)
    {
        if (targetObject != null && !targetObject.Equals(target))
            DestroyComponent();
        targetObject = target;
        CreateComponent();
    }
    override public void CancelTarget()
    {
        DestroyComponent();
        targetObject = null;
    }
}
```

```
public void SetValue(string key, string value)
{
    valueTable[key] = (float)Convert.ToDouble(value);
}
private void CreateComponent()
{
    if (this.targetObject != null &&
this.targetObject.GetComponent<SingleWheelController>() != null)
    {

        Debug.Log(this.targetObject.GetComponent<SingleWheelController>()
.maxMotorTorque);
        if (valueTable["MotorTorque"] != null)
        {

            this.targetObject.GetComponent<SingleWheelController>().maxMotorTorque
= (float)valueTable["MotorTorque"];

        }

    }
}
private void DestroyComponent()
{
    if (this.targetObject != null &&
this.targetObject.GetComponent<AddEngineMotor>() != null)
    {
        Destroy(targetObject.GetComponent<AddEngineMotor>());
    }
}
```

```
this.targetObject.GetComponent<SingleWheelController>().maxMotorTorque
= 0;

    }

}

}
```

- 바퀴 동력 추가(Rigidbody와 엔진 필요)

```
public class AddWheelPower : PowerScript
{
    private SingleAxleInfo singleAxel;
    private WheelCollider wheel;
    private bool motor;
    private bool steering;

    void Awake()
    {
        singleAxel = new SingleAxleInfo();
        motor = true;
        steering = true;

        singleAxel.motor = this.motor;
        singleAxel.steering = this.steering;
    }
    override public void SetTarget(GameObject target)
    {
        Debug.Log("AddWheelPower SetTarget");
        if (targetObject != null && !targetObject.Equals(target))
            DestroyComponent();
        targetObject = target;
    }
}
```

```

        CreateComponent();
    }
    override public void CancelTarget()
    {
        DestroyComponent();
        targetObject = null;

    }
    private void SetWheel()
    {
        Debug.Log("SetWheel");
        if (targetObject.GetComponentInParent<Rigidbody>() != null)
        {
            wheel = targetObject.GetComponentInChildren<WheelCollider>();
            if (targetObject.GetComponentInParent<SingleWheelController>()
== null)
            {

targetObject.GetComponentInParent<Rigidbody>().gameObject.AddComponent
<SingleWheelController>();
                targetObject.GetComponentInParent<SingleWheelController>()
.singleAxleInfos = new List<SingleAxleInfo>();
                targetObject.GetComponentInParent<SingleWheelController>()
.maxMotorTorque = 150;
                targetObject.GetComponentInParent<SingleWheelController>()
.maxSteeringAngle = 15;

            }
            singleAxel.wheel = wheel;
            List<SingleAxleInfo>                                axelList                                =

```



```

targetObject.GetComponentInParent<SingleWheelController>().singleAxleInfos;
    if(!axelList.Contains(singleAxel))
        axelList.Add(singleAxel);
    }
}
private void CancelWheel()
{
    if (targetObject.GetComponentInParent<Rigidbody>() != null)
    {
        if
(targetObject.GetComponentInParent<SingleWheelController>() != null)
        {
            List<SingleAxleInfo> axelList =
targetObject.GetComponentInParent<SingleWheelController>().singleAxleInfos;
            if(axelList.Contains(singleAxel))

targetObject.GetComponentInParent<SingleWheelController>().singleAxleInfos
.Remove(singleAxel);
            singleAxel.wheel = null;
            if
(targetObject.GetComponentInParent<SingleWheelController>()
.singleAxleInfos.Count == 0)
            {

Destroy(targetObject.GetComponentInParent<Rigidbody>())
.gameObject.GetComponent<SingleWheelController>());
            }

        }
    }
}

```

```

    }
}
private void CreateComponent()
{
    if(this.targetObject != null)
        SetWheel();
}
private void DestroyComponent()
{
    if (this.targetObject != null)
        CancelWheel();
}
}

```

● 배치 스크립트 정렬 코드

```

public void ArrangeAfterAdd(GameObject newObject)
{
    bool isNewFrame = true;
    string searchName = null;
    int targetDepth = 0;
    float scriptHeight;

    Debug.Log("Arrange()");

    FrameTarget frameTarget =
    GameObject.Find("FrameManager").GetComponent<FrameTarget>();

    // 제어문 아래쪽 박스 만들기
    if (newObject.GetComponent<IMotionScriptSet>().IsInFrame())

```

```

    {
        GameObject belowBox
    = GameObject.CreatePrimitive(PrimitiveType.Cube);
        belowBox.name = "Below";
        belowBox.transform.localScale = newObject.transform.lossyScale;
        belowBox.transform.localPosition
    = newObject.transform.localPosition +
            new Vector3(0f, -newObject.transform.lossyScale.y, 0f);

        belowBox.GetComponent<MeshRenderer>().material.color =
ScriptColorList[newObject.GetComponent<IMotionScriptSet>().GetDepth()];

        belowBox.transform.parent = newObject.transform;
        newObject.AddComponent<SubFrame>();

        GameObject subFrame =

Instantiate(GameObject.Find("FrameManager").GetComponent<NewFrame>()
.OriginFrame);
        subFrame.name = "SubFrame" + FrameTarget.SubFrameNum++;
        scriptHeight =
newObject.GetComponent<SubFrame>().FrameSize();

        subFrame.transform.localPosition =
newObject.GetComponent<SubFrame>().CenterPos() + new Vector3(0, 0,
0.005f);
        subFrame.transform.localScale = newObject.transform.lossyScale
+ new Vector3(0.3f, scriptHeight, 0f);
        subFrame.transform.parent = newObject.transform;
    }

```

```

        subFrame.AddComponent<FrameGizmo>();
        // 프레임에 scriptFrame 추가
        subFrame.AddComponent<SubScript>();
        // FrameRenderer 추가
        subFrame.AddComponent<FrameRenderer>();

        newObject.GetComponent<ParentControlSet>().
SetScriptFrame(subFrame);
        subFrame.AddComponent<ScriptTarget>();

        frameTarget.AddFrame(subFrame.name, subFrame);

        newObject.GetComponent<ScriptScaleInfo>().ScriptScale =
newObject.transform.localScale + new Vector3(0f, scriptHeight, 0f);
    }
    else    newObject.GetComponent<ScriptScaleInfo>().ScriptScale =
newObject.transform.localScale;

    foreach (GameObject targetFrame in frameTarget.MadeFrames.Values)
    {
        string targetName = targetFrame.name;
        string parentName = targetFrame.name;

        EmptyFrameState    frameState =
targetFrame.GetComponent<EmptyFrameState>();
        if (frameState != null && !frameState.IsScriptEmpty)
        {
            searchName = targetName;

```

```

        newObject.transform.position =
targetFrame.transform.position;
        //newObject.transform.Translate(new Vector3(0f, 0f, -0.01f),
Space.Self);
        newObject.transform.rotation =
targetFrame.transform.rotation;

        Debug.Log("Arrange Delete Scripts...");
        frameState.DestroyEmptyStateInfos();
        Destroy(frameState);

        isNewFrame = false;
        break;
    }
    foreach (LookScript lookScript in
targetFrame.GetComponentsInChildren<LookScript>())
    {
        if (lookScript.IsSelect)
        {
            if
(targetScript.GetSelectGizmo().GetComponent<ScriptGizmo>().GizmoNo
== (int)GizmoName.IN)
            {
                targetName =
lookScript.GetComponentInChildren<SubScript>().name;
            }
            else
            {
                targetName = lookScript.transform.parent.name;
            }
        }
    }

```

```

        Debug.Log(targetName + " : " + lookScript.transform.
parent.name);

        GameObject currFrame =
frameTarget.GetFrame(targetName);
        frameScript = currFrame.GetComponent<PlayScript>();
        searchName = targetName;

        // 자식ScriptFrame 유무 확인(유무에 따라 TargetDepth 설정
        IMotionScriptSet connectScript =
lookScript.GetComponent<IMotionScriptSet>();
        if (connectScript.IsInFrame() &&
lookScript.GetSelectGizmo().GetComponent<ScriptGizmo>().GizmoNo
== (int)GizmoName.IN)
            targetDepth = connectScript.GetDepth() + 1;
        else targetDepth = connectScript.GetDepth();

newObject.GetComponent<IMotionScriptSet>().SetDepth(targetDepth);

        Transform scriptTr =
lookScript.GetSelectGizmo().transform;
        newObject.transform.position = scriptTr.position;
        newObject.transform.rotation = scriptTr.rotation;

        scriptTr.GetComponent<ScriptGizmo>().ArrangeScript =
newObject.transform;
        isNewFrame = false;
        //break;
    }

```

```
    }
    if (!isNewFrame)
    {

        GameObject currFrame = frameTarget.GetFrame(targetName);

        List<Transform> tmpScriptTrs = new List<Transform>();
        // 부모 임시 해제
        while (currFrame.transform.childCount != 0)
        {
            tmpScriptTrs.Add(currFrame.transform.GetChild(0));
            currFrame.transform.GetChild(0).parent
= currFrame.transform.parent;
        }

        // 지정된 프레임 크기 늘리기 ( 크기 / 위치 조정하는 값 수정
        필요 )

        if (newObject.GetComponent<SubFrame>() != null)
        {
            scriptHeight
= newObject.GetComponent<SubFrame>().FrameSize() +
newObject.transform.lossyScale.y;
        }
        else
        {
            scriptHeight = newObject.transform.lossyScale.y;
        }
        currFrame.transform.localScale
+= new Vector3(0f, scriptHeight, 0f);
    }
```

```
//currFrame.transform.localPosition
+= new Vector3(0f, -scriptHeight / 2, 0f);
currFrame.transform.Translate(new Vector3(0f, -scriptHeight /
2, 0f), Space.Self);

// 부모 다시 재설정
for (int i = 0; i < tmpScriptTrs.Count; ++i)
{
    tmpScriptTrs[i].parent = currFrame.transform;
}

break;
}
}
if (isNewFrame)
{
    string targetName = "CreateFrame" + FrameTarget.FrameNum++;
    GameObject player
= GameObject.FindGameObjectWithTag("MadeScripts");

// 새 프레임 생성
GameObject newFrame =
    Instantiate(GameObject.Find("FrameManager").
GetComponent<NewFrame>().OriginFrame);
newFrame.name = targetName;
newFrame.tag = "Frame";
```



```

        if (newObject.GetComponent<SubFrame>() != null)
        {
            scriptHeight =
newObject.GetComponent<SubFrame>().FrameSize();

            //newFrame.transform.localPosition =
newObject.GetComponent<SubFrame>().CenterPos() + new Vector3(0, 0,
0.01f);

            newFrame.transform.position =
newObject.GetComponent<SubFrame>().CenterPos();
            //newFrame.transform.Translate(new Vector3(0, 0, 0.01f),
Space.Self);

            newFrame.transform.localScale =
newObject.transform.lossyScale
+ new Vector3(0.3f, 0.3f, 0) + new Vector3(0f,
scriptHeight, 0f);
        }
        else
        {
            newFrame.transform.position = newObject.transform.position;

            newFrame.transform.localScale =
newObject.transform.localScale +
new Vector3(0.3f, 0.3f, 0);
        }
        newFrame.transform.rotation = newObject.transform.rotation;

        newFrame.AddComponent<DragFrame>();
        // FrameGizmo 추가

```

```

newFrame.AddComponent<FrameGizmo>();
// FrameRenderer 추가
newFrame.AddComponent<FrameRenderer>();

frameTarget.AddFrame(targetName, newFrame);
frameScript = newFrame.AddComponent<MainScript>();
newFrame.AddComponent<ScriptTarget>();
newFrame.transform.SetParent(player.transform);
searchName = targetName;

// frame 깊이 설정
((ScriptFrame)frameScript.ScriptFrame).Depth = 0;
// Script Depth 설정
newObject.GetComponent<IMotionScriptSet>().SetDepth(0);
// target Depth 설정
targetDepth = 0;

// frame 색 설정
newFrame.GetComponent<MeshRenderer>().material.color =
frameColorList[((ScriptFrame)frameScript.ScriptFrame).Depth];

}

//newObject.transform.parent =
frameTarget.GetFrame(searchName).transform;

newObject.transform.SetParent(frameTarget.GetFrame(searchName).transform);
//
newObject.transform.Translate(new Vector3(0, 0, -0.01f), Space.Self);

```

```
        if (newObject.GetComponent<SubFrame>() != null)
        {

            newObject.transform.GetChild(0).Translate(new Vector3(0, 0, -0.01f),
            Space.Self);

            newObject.transform.GetChild(1).Translate(new Vector3(0, 0, -0.01f),
            Space.Self);
        }

        newObject.GetComponent<MeshRenderer>().material.color =
        ScriptColorList[newObject.GetComponent<IMotionScriptSet>().GetDepth()];

        ((ScriptFrame)frameScript.ScriptFrame)
        .AddScript(newObject, targetDepth, searchName);

        this.FramePositionSetting(frameScript.ScriptFrame.ScriptPositions,
        frameTarget.GetFrame(searchName));
        newObject.AddComponent<DragArrangedScript>();
    }
```

● 스크립트 동작시키는 코드

```
public class MainScript : PlayScript {

    private ScriptEventManager eventManager;
    private KeyInputScript keyInputScript;
    private string keyName;
    void Awake()
    {
        if (target == null) target =
GameObject.FindGameObjectWithTag("Player");
        scriptFrame = this.gameObject.AddComponent<ScriptFrame>();
        ((ScriptFrame)scriptFrame).Depth = 0;
        scriptFrame.ScriptTarget = target;
        eventManager =
GameObject.Find("ScriptEventManager").GetComponent<ScriptEventManager>();
    }

    public bool IsInDefinedKey()
    {
        if (keyInputScript != null) return true;
        else return false;
    }

    public void SetFrameKeyInput(IEventScriptSet eventScript)
    {
        KeyInputScript target = eventScript as KeyInputScript;
        if(target != null)
        {
            keyInputScript = target;
        }
    }
}
```

```
}

public void CancelFrameKeyInput()
{
    keyInputScript = null;
}

override public void ChangeTarget(GameObject target)
{
    scriptFrame.ChangeTarget(target);
}

public override void StartRoutine()
{
    if (keyInputScript == null) keyName = "return";
    else keyName = keyInputScript.KeyName;
    if (scriptFrame.ScriptTarget != null &&
eventManager.ReturnButton(keyName) && !scriptFrame.IsPlaying)
    {
        Debug.Log("Play!!!!");
        eventManager.ButtonStop(keyName);
        ((ScriptFrame)scriptFrame).Play();
    }
}

// Update is called once per frame
void Update()
{
    StartRoutine();
}
```

```
}
```

```
}
```

● Block 생성 코드

```
//블락 구조체 만들기
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class csCRBlock : MonoBehaviour {

    [SerializeField]
    private BlockType[] allblockTypes;

    [HideInInspector]
    public Dictionary<int, Block> allBlocks = new Dictionary<int, Block>();

    private void Awake()
    {
        for (int i=0; i < allblockTypes.Length;i++)
        {
            BlockType newBlockType = allblockTypes[i];
            Block newBlock = new Block(i, newBlockType.blockName,
newBlockType.blockMat);
            allBlocks[i] = newBlock;
            Debug.Log("Block added to dictionary" + allBlocks[i].blockName);
        }
    }
}
```

```
public class Block
{
    public int blockID;
    public string blockName;
    public Material blockMaterial;

    public Block(int id, string name, Material mat)
    {
        blockID = id;
        blockName = name;
        blockMaterial = mat;
    }
}

[Serializable]
public struct BlockType
{
    public string blockName;
    public Material blockMat;
}

//Block 생성 위치 표시 및 생성
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class BuildingSystem : MonoBehaviour {

    private bool buildModeOn = false;
    private bool canBuild = false;
    private int curr = -1;
    private csCRBlock bSys;
    private List<GameObject> list;
    [SerializeField]
```

```

private LayerMask buildableSurfacesLayer;
private Vector3 buildPos;
private GameObject currentTemplateBlock;
[SerializeField]
private GameObject blockTemplatePrefab;
[SerializeField]
private GameObject blockPrefab;
[SerializeField]
private Material templateMaterial;
private int blockSelectCounter = 0;
private void Start()
{
    bSys = GetComponent<csCRBlock>();
}
public void OBJ_CR_Cube()
{
    Mainpanel.SetActive(false);
    buildModeOn = !buildModeOn;
    tryModeOn = false;
    LTirebuildModeOn = false;//tire- L
    CarbuildModeOn = false;//tire r
    PyramidbuildModeOn = false;
    CapsulebuildModeOn = false;
    tryModeOn = false;
    CarbuildModeOn = false;//tire r
    NLTirebuildModeOn = false;
    NRTirebuildModeOn = false;
    YjointbuildModeOn = false;
    XjointbuildModeOn = false;
    baseroModeOn = false;
    sqpyModeOn = false;
    OBJPanel.SetActive(false);
    YJointbuildModeOn = false;

```



```
XJointbuildModeOn = false;

if (buildModeOn)
{
    Cursor.lockState = CursorLockMode.Locked;
}
else
{
    Cursor.lockState = CursorLockMode.None;
}
}

private void Update()
{
    activeController=OVRInputHelpers.GetControllerForButton(OVRInput.
    Button.PrimaryIndexTrigger, activeController);
    Ray pointer = OVRInputHelpers.GetSelectionRay(activeController, trackingSpace);
    if (buildModeOn)
    {
        RaycastHit buildPosHit;

        if (Physics.Raycast(pointer,
            out buildPosHit, 10, buildableSurfacesLayer))
        {
            Vector3 point = buildPosHit.point;
            buildPos      =      new      Vector3(Mathf.Round(point.x),
            Mathf.Round(point.y)+0.9f, Mathf.Round(point.z));
            canBuild = true;
        }
        else
        {
            Destroy(currentTemplateBlock.gameObject);
            canBuild = false;
        }
    }
}
```

```

    }

    if (!buildModeOn && currentTemplateBlock != null)
    {
        Destroy(currentTemplateBlock.gameObject);
        canBuild = false;
    }
    if (canBuild && currentTemplateBlock == null)
    {
        currentTemplateBlock = Instantiate(blockTemplatePrefab, buildPos,
        Quaternion.identity);
        currentTemplateBlock.GetComponent<MeshRenderer>().material =
        templateMaterial;
    }
    if (canBuild && currentTemplateBlock != null)
    {
        currentTemplateBlock.transform.position = buildPos;

        if (OVRInput.GetDown(OVRInput.Button.PrimaryIndexTrigger,
        OVRInput.Controller.LTouch))
        {
            PlaceBlock();
        }
    }
}

//Block 생성
private void PlaceBlock()
{
    GameObject newBlock = Instantiate(blockPrefab, buildPos, Quaternion.identity);
    GameObject parent = list[curr];
    newBlock.transform.parent = parent.transform;
    Block tempBlock = bSys.allBlocks[blockSelectCounter];
    newBlock.name = tempBlock.blockName + "-Block";
    newBlock.GetComponent<MeshRenderer>().material = tempBlock.blockMaterial;
}

```

```
}

```

● Add Component 기능

//Grouping 버튼을 눌렀을 때 Create GameObject를 만든다.

```
public void Main_CR_Gameobject()
{
    GameObject emptyObject = new GameObject("empty" + (++curr));
    list.Add(emptyObject);
}
```

//Add Rigidbody를 눌렀을 때 마지막 Grouping 된 게임오브젝트를 찾아 Rigidbody기능을 추가

```
public void Main_Rigidbody()
{
    Rigidbody gameobjectRigidbody =
    GameObject.Find("empty" + (curr)).AddComponent<Rigidbody>();
    GameObject.Find("empty" + (curr)).transform.position = new Vector3(0, 2,
0);
    gameobjectRigidbody.mass = 15000;
}
```

● UI Panel On / Off 기능

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.SceneManagement;
using ControllerSelection;
public class JebalSwitchPanel : MonoBehaviour {
//Panel들을 불러오는
    public GameObject MainPanel;
    public GameObject SCBOXPanel;
    public GameObject ObjectPanel;
    public GameObject KitPanel;
    public GameObject AddCompenentPanel;
    public GameObject Kit_CarPanel;
    public GameObject Kit_DynoPanel;
    public GameObject kit_PorkrainPanel;
    public bool MainSwitchOn = false;

[System.Serializable]
    public class HoverCallback : UnityEvent<Transform> { }
[System.Serializable]
    public class SelectionCallback : UnityEvent<Transform> { }

[Header("(Optional) Tracking space")]
[Tooltip("Tracking space of the OVRCameraRig.\nIf tracking space is not set, the scene will be searched.\nThis search is expensive.")]
    public Transform trackingSpace = null;

[Header("Selection")]
[Tooltip("Primary selection button")]
    public OVRInput.Button primaryButton = OVRInput.Button.PrimaryIndexTrigger;

```

```
[Tooltip("Secondary selection button")]
public OVRInput.Button secondaryButton = OVRInput.Button.PrimaryTouchpad;
[Tooltip("Layers to exclude from raycast")]
public LayerMask excludeLayers;
[Tooltip("Maximum raycast distance")]
public float raycastDistance = 500;

[Header("Hover Callbacks")]
public OVRRawRaycaster.HoverCallback onHoverEnter;
public OVRRawRaycaster.HoverCallback onHoverExit;
public OVRRawRaycaster.HoverCallback onHover;

[Header("Selection Callbacks")]
public OVRRawRaycaster.SelectionCallback onPrimarySelect;
public OVRRawRaycaster.SelectionCallback onSecondarySelect;

protected Transform lastHit = null;
protected Transform triggerDown = null;
protected Transform padDown = null;

[HideInInspector]
public OVRInput.Controller activeController = OVRInput.Controller.None;

void Awake()
{
    if (trackingSpace == null)
    {
        Debug.LogWarning("OVRRawRaycaster did not have a tracking space set. Looking for one");
        trackingSpace = OVRInputHelpers.FindTrackingSpace();
    }
}

public void Sw_Add_Panel()
```

```
{
    AddCompenentPanel.SetActive(true);
    KitPanel.SetActive(false);
    Kit_DynoPanel.SetActive(false);
    kit_PorkrainPanel.SetActive(false);
    Kit_CarPanel.SetActive(false);
    ObjectPanel.SetActive(false);
    Debug.Log("AddPanel");
}
public void Sw_ObjectPanel()
{
    ObjectPanel.SetActive(true);
    AddCompenentPanel.SetActive(false);
    KitPanel.SetActive(false);
    Kit_DynoPanel.SetActive(false);
    kit_PorkrainPanel.SetActive(false);
    Kit_CarPanel.SetActive(false);
}
public void Sw_KitPanel()
{
    KitPanel.SetActive(true);
    AddCompenentPanel.SetActive(false);
    ObjectPanel.SetActive(false);
    Kit_DynoPanel.SetActive(false);
    kit_PorkrainPanel.SetActive(false);
    Kit_CarPanel.SetActive(false);
}
public void Sw_Kit_DynoPanel()
{
    Kit_DynoPanel.SetActive(true);
    AddCompenentPanel.SetActive(false);
    ObjectPanel.SetActive(false);
    kit_PorkrainPanel.SetActive(false);
}
```

```
        Kit_CarPanel.SetActive(false);
    }
    public void Sw_kit_CarKitPanel()
    {
        Kit_CarPanel.SetActive(true);
        AddCompenentPanel.SetActive(false);
        ObjectPanel.SetActive(false);
        Kit_DynoPanel.SetActive(false);
        kit_PorkrainPanel.SetActive(false);

    }
    public void Sw_Kit_PorkrainPanel()
    {
        kit_PorkrainPanel.SetActive(true);
        AddCompenentPanel.SetActive(false);
        ObjectPanel.SetActive(false);
        Kit_DynoPanel.SetActive(false);
        Kit_CarPanel.SetActive(false);
    }

    public void OnOffUI()
    {
        AddCompenentPanel.SetActive(false);
        ObjectPanel.SetActive(false);
        KitPanel.SetActive(false);
        kit_PorkrainPanel.SetActive(false);
        Kit_CarPanel.SetActive(false);
        Kit_DynoPanel.SetActive(false);
        MainSwitchOn = !MainSwitchOn;

        if (MainSwitchOn)
        {
            MainPanel.SetActive(true);
        }
    }
}
```

```
    }  
    else if (!MainSwitchOn)  
    {  
        MainPanel.SetActive(false);  
    }  
  
    }  
}
```


5. 사용자 매뉴얼

A. 개발환경설치

I. 유니티

3D 환경을 구축하기 위해 컴퓨터에 유니티 게임 엔진을 설치한다

II. Visual Studio

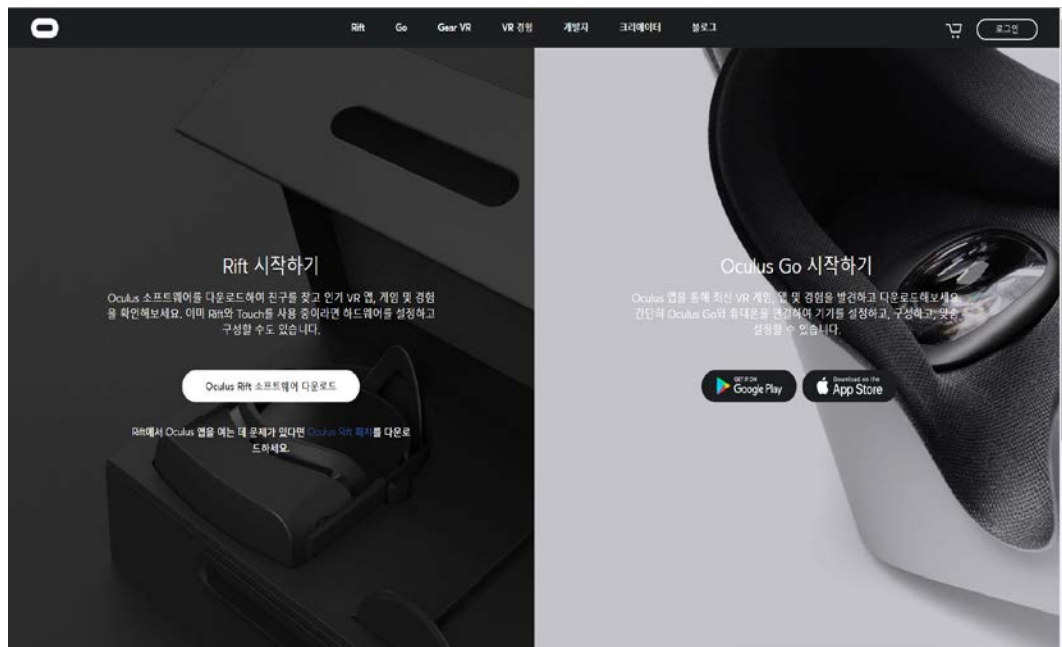
C# 언어로 구현된 코드를 편집하기 위해 Visual Studio를 설치한다

III. Oculus Rift

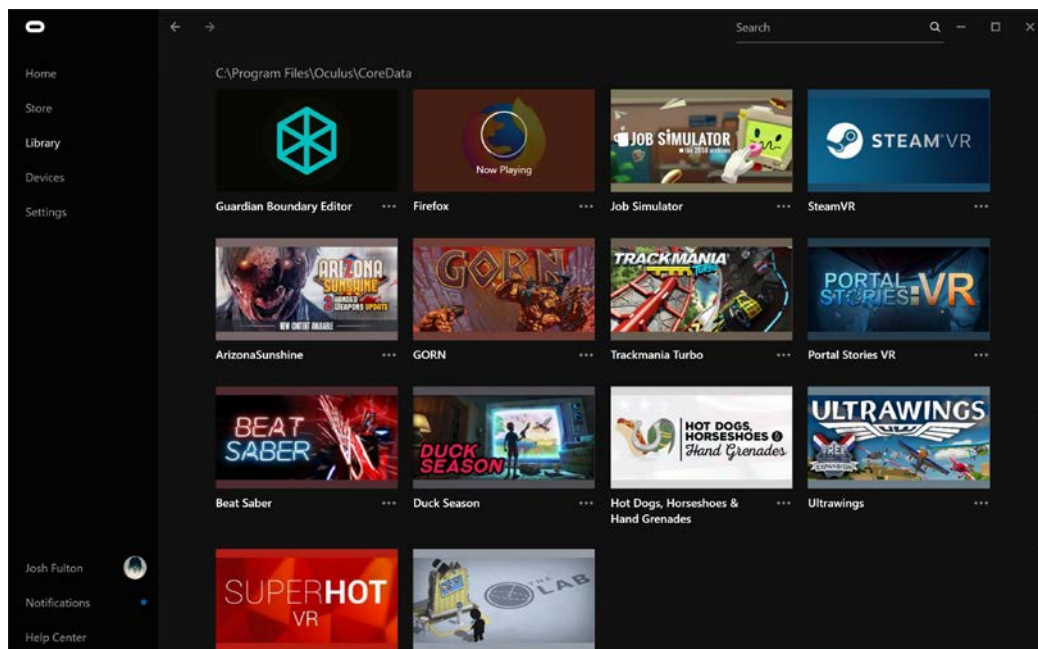
Oculus 프로그램을 설치하고, Oculus 장비들을 컴퓨터에 연동시킨다.

B. Oculus 설치하기

I. Oculus Rift 홈페이지에 접속해 Oculus Rift 소프트웨어를 다운로드 한다.

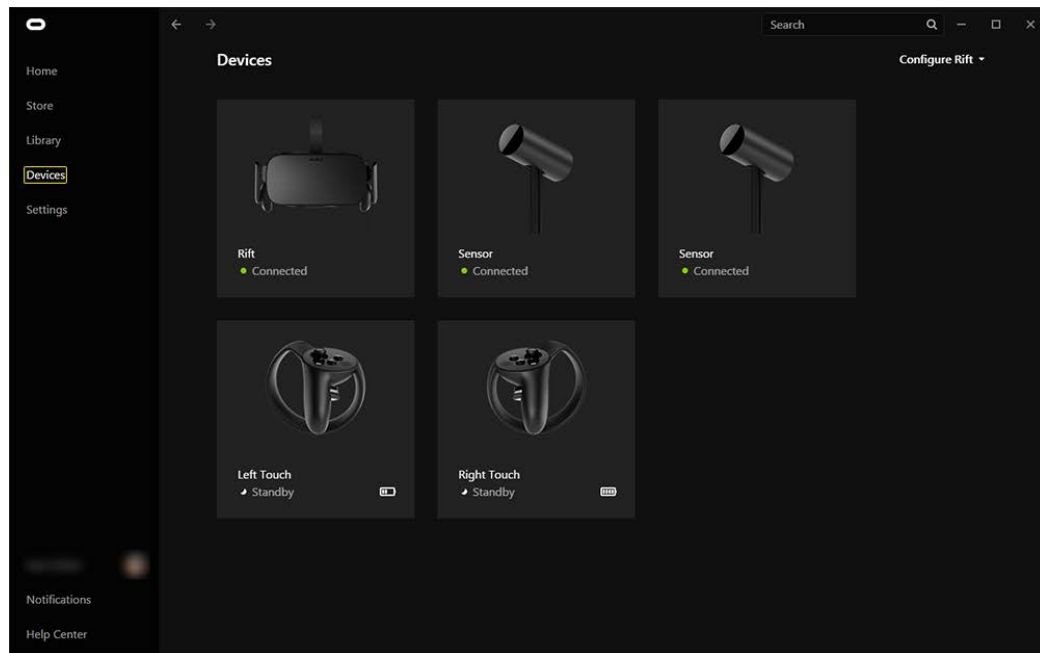


II. Oculus Program 을 실행시킨다.



- Oculus Rift를 사용하기 위해서는 로그인을 해야한다. 로그인 ID가 없을 경우 회원가입을 해야 한다.

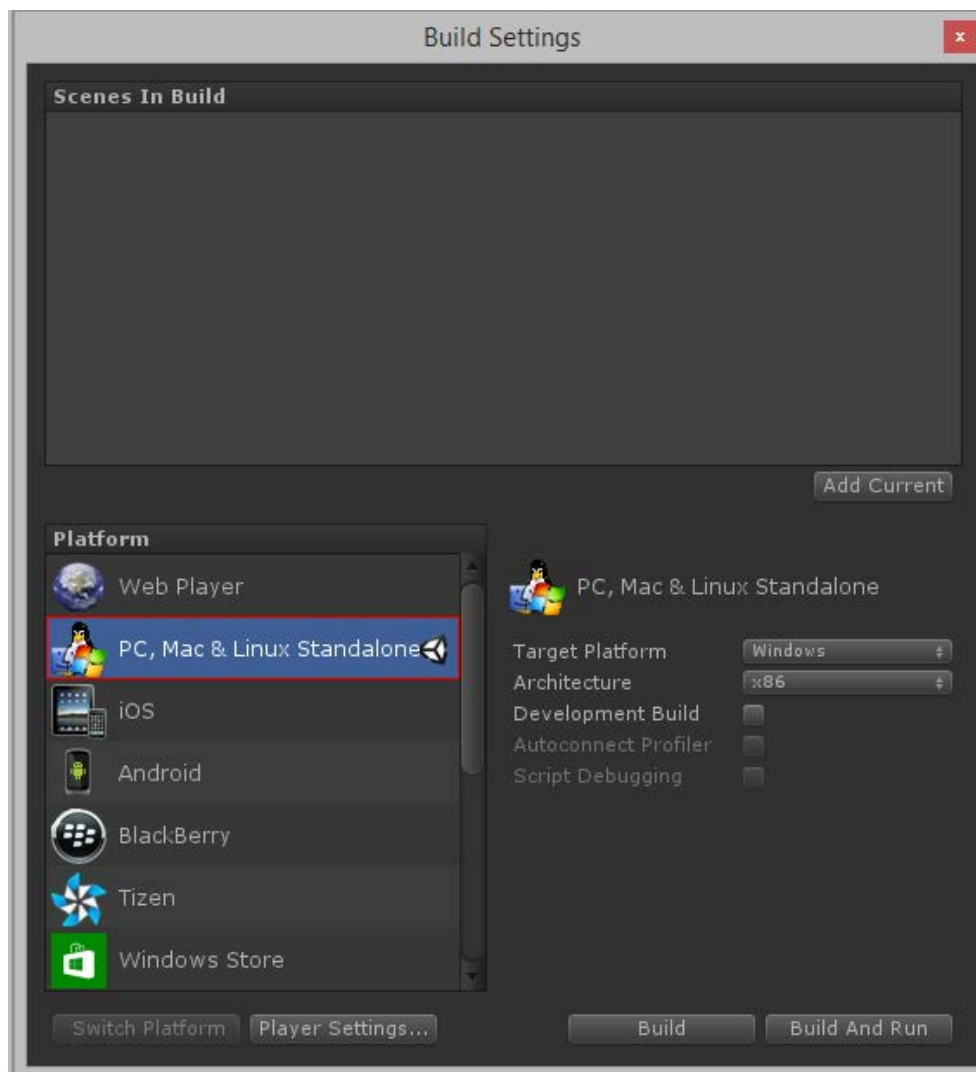
III. Oculus Rift 장비들이 제대로 연결이 되어있나 확인한다.

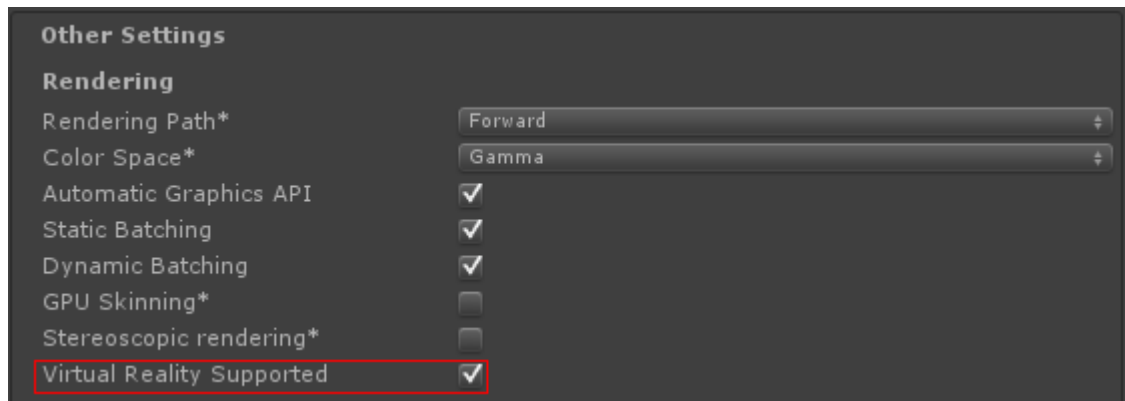


- Oculus Program에는 Devices 연결 확인을 위한 페이지가 있으므로, 클릭하여 자신의 장비가 잘 연결되어 있나 확인한다.

C. 유니티에 Oculus Rift 개발 환경 구축하기

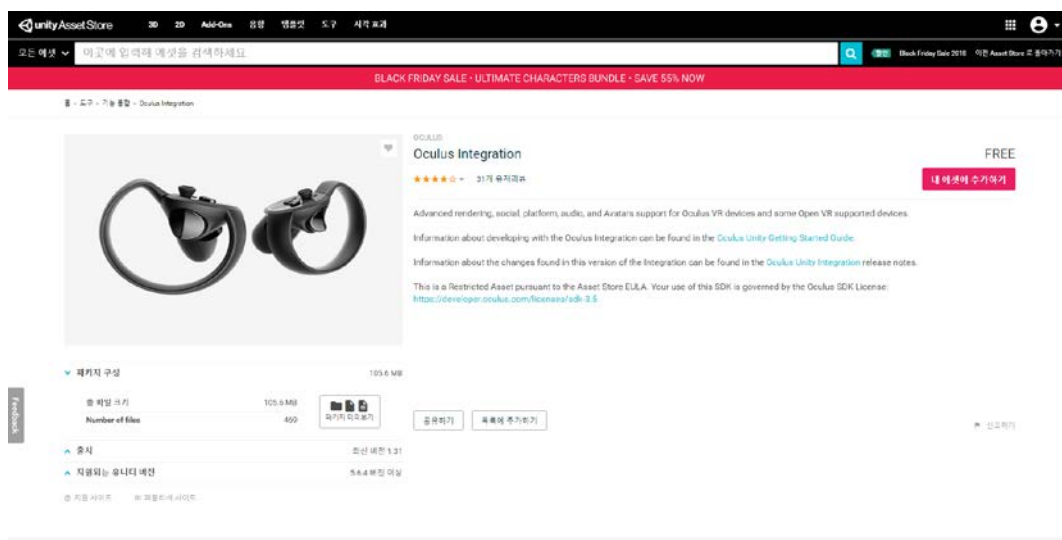
- I. Unity에서 VR 환경을 구축하기 위해서 Player Settings의 값들을 변경해준다.





- Oculus Rift가 설치되어 있는 상태이기 때문에, Unity에서 VR SDK에 Oculus를 자동으로 보여준다.

II. Unity Assets Store에 접속해 Oculus Integration 패키지를 다운로드한다.



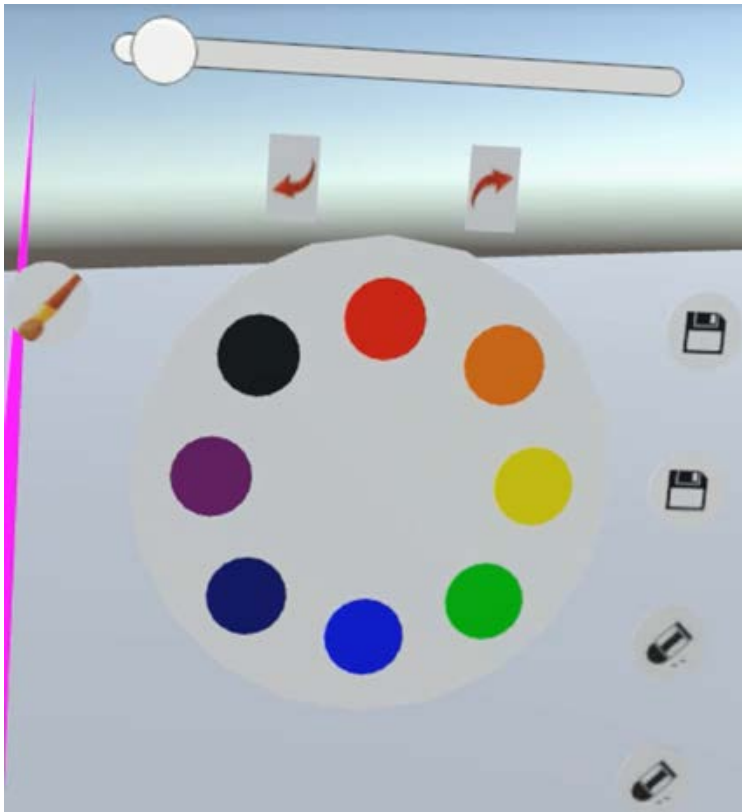
- Oculus Integration 에는 VR camera를 컨트롤 할 수 있는 인터페이스, 컨트롤러 Prefab들, 컨트롤러에 대한 각종 API, 게임 오브젝트 Grabbing 스크립트 등 유니티에서 기본적으로 사용할 수 있는 기능들이 포함되어 있다.
- 우리는 프로젝트를 위하여, Oculus Integration 패키지에서 제공하는 Camera와 컨트롤러의 Grabbing script 등을 사용하였다.

III. Unity Integration에 있는 기본 기능을 테스트 해본다.



D. 프로그램 작동 매뉴얼

I. Tilt Brush Main UI



Brush 기능의 On / Off



선의 색상을 변경한다



선의 두께를 조절한다



가장 마지막으로 그려진 선을 비활성화한다



가장 마지막으로 비활성화된 선을 다시 활성화한다

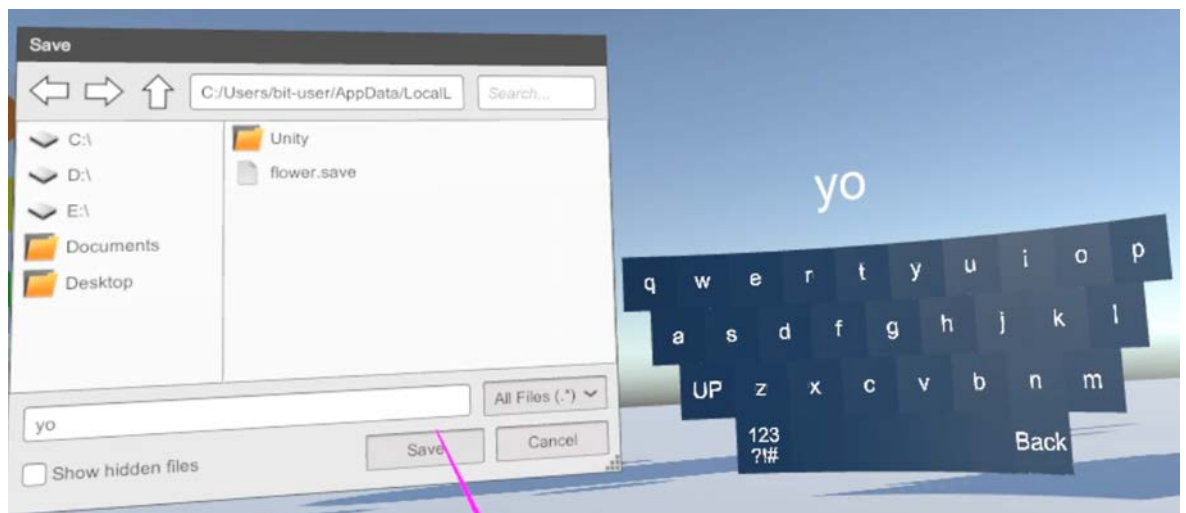


그려진 그림 전체를 지우거나(위), 지우고 싶은 선을 개별적으로 지운다(아래)

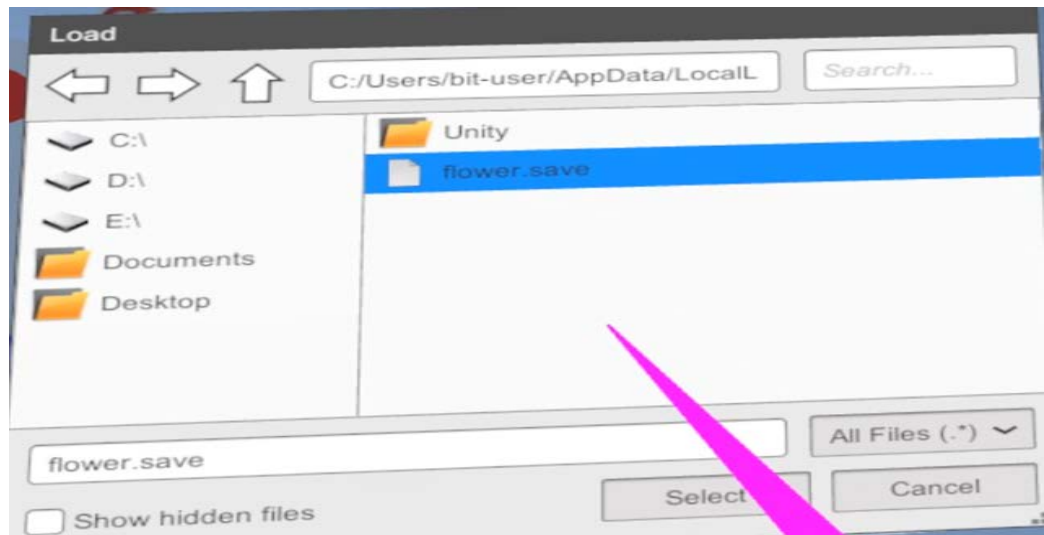


파일을 저장하고(위) 불러온다(아래)

II. Tilt Brush Save / Load UI



- 그림을 저장하기 위한 File Browser가 띄워진다
- 파일의 이름은 Keyboard UI를 통해서 정한다

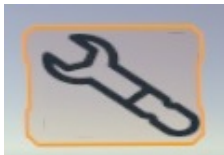


저장한 그림을 불러오기 위한 UI

III. Tilt Brush 로 그려진 그림의 예시



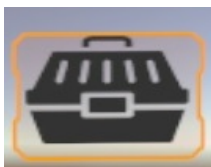
IV. 과학상자 Main UI



ADD Option: Gruping과 Physical Effect의 기능을 실행한다.



Object: pyramid, cube등의 Obect를 출력한다.



Kit:Object로 미리만들어 놓은 모형을 출력한다.



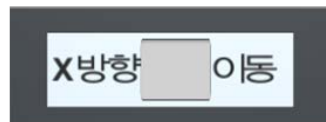
Script: Object에 Script를 넣어 움직임을 구현한다.

V.Script UI 예시

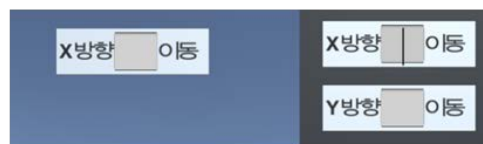


- 넣고 싶은 기능의 스크립트를 선택할 수 있다
- 선택한 대상을 Controller를 통해 Drag & Drop 해준다
- 선택 예시

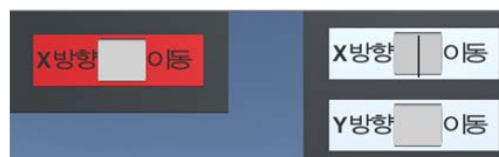
① 스크립트 선택



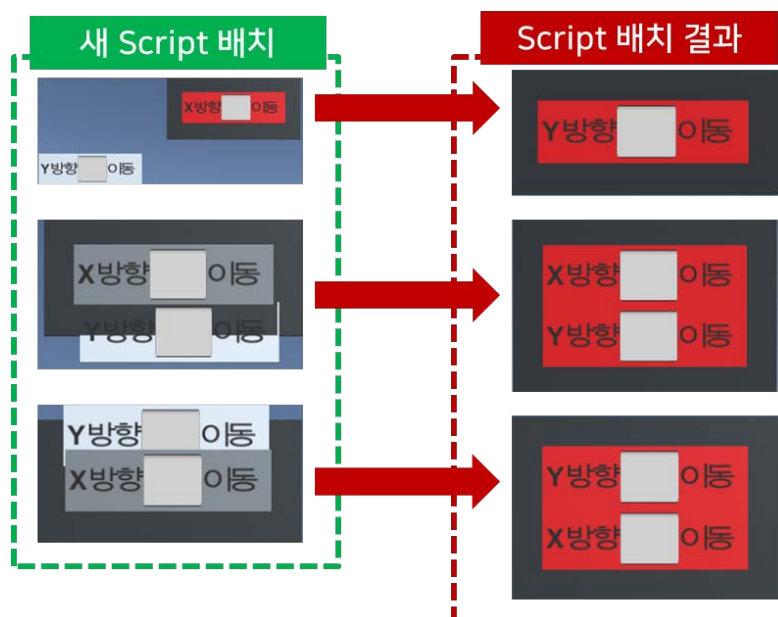
② 선택한 스크립트 화면으로 이동



③ 선택 해제



VI. Script UI 추가



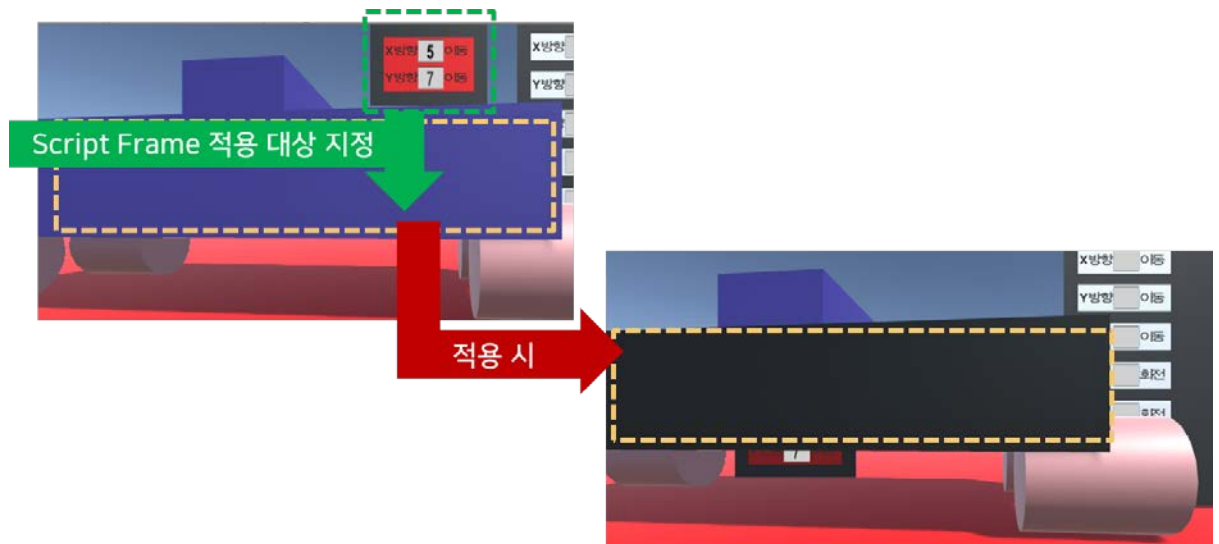
- 배치된 Script 위 / 아래에 추가로 배치할 수 있다
- 동일하게 기능을 사용하고 싶은 Script를 Drag & Drop 해준다
- 동일한 Frame에 있다면 기능이 위 - 아래 순으로 차례대로 수행된다

VII. Script 값 입력



- Script가 값 입력할 수 있는 형태라면 값을 입력할 수 있다
- Holder를 Controller가 쓰는 Ray를 통해서 선택할 수 있다.

VIII. Script 적용 대상 선택



- 만든 Frame을 적용할 대상을 지정할 수 있다
- Frame을 Controller의 Ray를 통해 선택하고 그 대상을 Drag 하여 적용 대상에 가까이 위치시킨다.
- 선택 대상이 적용 가능한 대상이라면 구분짓기 위해 Drag 상태에 있는 동안에는 기존 색과 다르게 표시되도록 한다.

6. 프로젝트 마무리

A. 기대효과

- i. 과학상자를 실물로 접하게 되면 분해 조립 과정에 있어 서로를 엮어주고 풀어주는 작업이 필요하다. DIY의 경우에는 분해 조립 과정이 비교적 단순한데, 이는 Block 형태로 쌓고 빼는 작업으로 구성되기 때문에 쉽게 구성할 수 있다.
- ii. 원하는 Contents를 만들기 위한 재료의 제약이 없다. 여러 Contents를 동시에 만들기 위해서는 재료가 그만큼 많이 필요한데, DIY는 PC 환경에서 제공되는 만큼 용량이나 컴퓨터 사양에 있어서 문제가 되지 않는다면 무한하게 만들고 싶은 Object를 만들 수 있다. 또한 동시에 그 Object들을 움직여 볼 수도 있다.
- iii. 아이들에게 친숙한 형태의 제작 Contents를 제공한다. Block은 단순한 모양을 가지고 있어 다양하게 만들고 쌓고 싶어하는 아이들에게 어렵지 않은 Contents이다. 또한 Brush의 경우에는 동화적인 느낌을 가지고 있기 때문에 그림 그리기를 좋아하는 아이들에게 있어서 PC에서 그림을 자유롭게 그릴 수 있다는 점에서 좋은 Contents라 할 수 있을 것이다. 때문에 창의력을 키울 수 있는 교육 Contents로 활용 할 수 있을 것이다.

B. 난제 극복 사례

- i. 기존 Unity에서 제공하는 GameObject의 모양이 단순 Cube, Cylinder, Sphere 등의 형태만 존재하여 삼각뿔이나 삼각 기둥과 같은 모양을 사용할 수 없다 이를 개선하고자 Asset Store에서 제공하는 Pro Builder를 통해 원하는 모양을 제작하여 Component로 제공하였다.
- ii. Script의 특성상 코드의 모양은 동일해도 기능이 달라 함수의 형태가 미묘하게 달라질 수 있어 코드를 분할할 경우 중복되는 코드가 많아지고 코드 간 모호성이 있어 디버깅 시 시간이 지체될 수 있다. 때문에 최대한 겹치는 형태나 기능이 있다면 상속이나 인터페이스를 사용하여 중복되는 코드를 막고, 해당 기능을 호출할 때에도 동적으로 실제 호출하고자 하는 코드가 실행될 수 있도록 구성하였다.
- iii. 처음 Brush 생성을 위해서 사용한 Trail Renderer는 Line Object를 생성하는 것이 아니라 Line을 계속적으로 그리는 형태인데 이는 Line을 그릴 때마다 Line이 구별되지 않는 특징을 지닌다. 때문에 Line을 따로 관리하기에는 문제가 있었기 때문에 이를 Line Renderer로 변경하였다. Line Renderer 역시 선을 그려주는 기능을 하지만, Line Renderer의 경우에는

Line 별로 GameObject 가 생성되기 때문에 Line 별 관리가 가능하고 Save / Load 시에도 구별된 Line으로 정보를 이용할 수 있다.

C. 문제점

- i. Oculus Rift 센서의 인식 가능 거리가 제한이 있기 때문에, 그 범위에서 벗어나면 프로그램이 잘 동작하지 않는 한계점이 있다.
- ii. 2족 보행 Kit를 제작할 수 있지만 움직임의 형태가 Wheel에 의한 움직임 / 단순 키 입력에 의한 단순 움직임만 구현이 되어 있어 2족 보행의 느낌처럼 부드럽게 움직이지 않는다는 한계점이 있다.
- iii. Oculus Controller의 버튼 수가 많지 않아 연동하는 시점에서 키보드처럼 다양한 키를 입력하기 어렵다.
- iv. 제공되는 Script 만을 가지고 Frame을 구성하여 적용할 수 있는데, 사용자가 정의하는 Script를 제작할 수 없다.
- v. 제공되는 Block만 사용할 수 있어 원하는 기본 Block을 만들어 낼 수 없다.

D. 개선방안

- i. 현재 Tilt Brush 에서 선택할 수 있는 색상이 제한이 있으므로, 컬러팔레트를 주어 광범위한 색상의 선택이 가능하게 만들 수 있다.
- ii. 선의 재질(천, 금속, 야광 등)을 선택할 수 있게하여, 그림에 다양성을 줄 수 있다.
- iii. 2족 보행 Kit의 부드러운 움직임을 위해서 적절한 Animation을 제작 / 응용하여 사용할 수 있다.
- iv. Oculus Controller가 쉽게 이용할 수 있는 자체 Controller UI를 최적화하여 제작, 사용한다
- v. 사용자 정의 Script를 제작하여, 특정 움직임을 만들어 낼 수 있다. 예를 들면 회전하면서 동시에 이동하는 등의 복합적인 Script 등이 그 예시이다.
- vi. Script UI의 경우 정해진 가로 길이만 제공되지만, Script 가 길어지는 경우에 한해서 가로 길이가 역시 길어져야 한다. 때문에 가로 길이가 동적으로 증가할 수 있도록 기능이 개선되어야 한다.
- vii. 전체적인 UI의 통일성이 없기 때문에 통일된 느낌을 줄 수 있도록 UI를 통일시켜주는 작업이 필요하다.

- viii. Block 추가 / 편집 기능을 두어 자유롭게 기본 Component를 제작할 수 있도록 한다.

E. 참고문헌 및 논문

F. 참고사이트

- i. Oculus
 - Oculus Rift Raycast 기능
<https://developer.oculus.com/blog/easy-controller-selection/>
 - Oculus Rift Touch Controller 사용법
<https://developer.oculus.com/documentation/unity/latest/concepts/unity-ovrinput/>
- ii. Tilt Brush 개발
 - Procedural Mesh 생성 기능
<http://jhrun.tistory.com/131>
 - Unity Basic Theory of Vector and Mesh 참고
<https://www.youtube.com/watch?v=7DK8aA2qee8/>
<https://www.youtube.com/watch?v=ucuOVL7c5Hw/>
<https://www.youtube.com/watch?v=yGfS-U740x4/>
<https://www.youtube.com/watch?v=v9E47DkckBE/>
https://www.youtube.com/watch?v=IJ9Tla_Q4gk/
- iii. 과학상자 개발
 - Block 만들기
<https://www.youtube.com/watch?v=0WZUgUtBxcg&feature=youtu.be>
- iv. 스크립트 개발
 - 스크래치 기능 UI 관련
<https://scratch.mit.edu/>
- v. 배경 지식
 - 유니티 배경 지식

[https://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%ED%8B%B0_\(%EA%B2%8C%EC%9E%84_%EC%97%94%EC%A7%84\)](https://ko.wikipedia.org/wiki/%EC%9C%A0%EB%8B%88%ED%8B%B0_(%EA%B2%8C%EC%9E%84_%EC%97%94%EC%A7%84))

- 비주얼 스튜디오 배경 지식

https://ko.wikipedia.org/wiki/%EB%A7%88%EC%9D%B4%ED%81%AC%EB%A1%9C%EC%86%8C%ED%94%84%ED%8A%B8_%EB%B9%84%EC%A3%BC%EC%96%BC_%EC%8A%A4%ED%8A%9C%EB%94%94%EC%98%A4#%EB%B9%84%EC%A3%BC%EC%96%BC_%EC%8A%A4%ED%8A%9C%EB%94%94%EC%98%A4_2017


- Oculus Rift 배경 지식

https://ko.wikipedia.org/wiki/%EC%98%A4%ED%81%98%EB%9F%AC%EC%8A%A4_%EB%A6%AC%ED%94%84%ED%8A%B8

<https://namu.wiki/w/%EC%98%A4%ED%81%98%EB%9F%AC%EC%8A%A4%20%EB%A6%AC%ED%94%84%ED%8A%B8>

- GTX 960 mini 사양

G. 팀원 별 소감

팀원	소감
<p>이준섭</p> 	<p>이전에 장기 프로젝트를 경험 해 본 적은 있지만, 팀장으로 역할을 수행한 것은 처음이라 부담감이 많이 있었습니다. 게다가 Unity 자체를 이번에 처음 배운 것이었는데, Unity에서 사용하는 툴의 기능들이 익숙치않아 적응하는데 시간이 필요했었고, 스크립트에 사용하는 c# 역시도 자체 프레임워크에 맞춰서 내용을 작성해야했기 때문에 곁핼기로만 알고 작업을 해야하다보니 이 또한 적응하는 시간이 필요했습니다. 프로젝트를 진행하면서 툴에 익숙해지고 스크립트 작성 방식에 익숙해지면서 어느정도 궤도에는 올라섰지만, 프로젝트의 자유도 설정에서 애를 먹으면서 원하는 기능을 100퍼센트 구현하지 못한 점은 아쉬웠고, 팀장으로 역할이 처음이어서 팀장으로 책임을 결단을 지어야 하는 부분에서 제대로 결정은 내리지 못한 적이 있어 진행도에서 약간의 지연이 있지 않았나 싶습니다. 하지만 제가 말았던 스크립트 기능 구현을 하면서 필연적으로 수행했던 상속 / 인터페이스 작성 연습은 객체 지향에 약간은 미숙했던 제게 있어서 한</p>

	<p>단계 설계 능력을 키울 수 있는 기회가 되었습니다. 또한 그동안 약했던 결단력을 가지고 일을 해결해가려는 노력들은 차후 현업에서 어려움이 봉착했을 때 큰 도움이 될 것이라 생각합니다.</p>
<p>유경동</p> 	<p>. VR기기를 사회에서 체험하지는 못했지만 내가 계획했던 아이디어를 팀원들이 찬성해주어서 시작할 수 있었던 프로젝트였다. 처음에는 내가 생각 했던 프로그램을 구상 및 계획을 하고 UI와 Object 프리팹을 만드는 것 까지는 순조롭게 진행되었지만 Object를 이용하여 자동차를 만들어 작동 시키거나 세부적인 기능을 추가할 때인 C#을 이용한 스크립트를 작성할 때부터 내가 할당받은 작업의 진행이 느려져서 빠르게 진행 되었던 것에 비해 많이 다른 팀원에게 비해 쳐지게된거 같았다. 이번 메인 프로젝트로 통해 미니 프로젝트에서 겪지 못했던 긴 시간동안 시간관리와 하루에 끝내야 할 목표를 끝내고 남는 시간에 차시 목표를 좀 더 미리 해보아야한다는걸 느꼈다.</p>
<p>한다인</p> 	<p>유튜브에서만 보던 VR 어플리케이션을 직접 구현한다는 것은 심리적으로 매우 부담되는 일이었습니다. 또한, 객체 지향 언어를 완벽하게 숙지하지는 않았기에, C# 기반의 프로젝트를 진행한다는 것은 일종의 모험이었던 것 같습니다. 하지만, 프로젝트의 한 기능을 도맡아 개발하면서, 내가 잘 모른다고 생각했던 지식이 사실은 적용을 해보지 않았을 뿐이었다는 것을 알게 되었습니다. 이번 메인 프로젝트는 알고있던 지식은 단단히 다지고, 모르고 있던 지식은 새로 습득해서 익혀나가는 일의 연속이었습니다. 또한, 팀원들과 프로젝트의 방향성, 계획, 일정 등을 상의해서 개발을 진행해 나가고, 프로젝트에 대한 자세한 설명이 기재된 문서를 작성하고, 구현된 프로그램을 지속적으로 발표하는 일 등을 통해서, 실제 현업에서 개발자들이 하고 있는 일을 간접적으로 체험한 것 같습니다.</p>