Leap Motion 과 유니티 엔진을 이용한 게임 개발

요 약

스마트 기기의 발전에 따라 현실 세계와 가상 세계는 점점 경계가 사라지고 있다. 이런 현상은 컴퓨터가 사람의 눈과 같이 공간과 물체, 그리고 둘을 구분하는 컴퓨터 그래픽스/비전 기술을 기반으로 하며, 이 기술을 활용한 게임 소프트웨어가 점점 인기를 얻고 있다. 다양한 게임이 등장하고 인기를 얻고 있지만, 동시에 사용자들이 겪는 불편함 역시 다양해지고 있다. 이에 본 문서는 Leap Motion 기기를 활용하는 게임 콘텐츠를 개발을 통해 기존의 하드웨어 및 소프트웨어의 단점을 보완하는 게임 환경을 제시하여 사용자들의 편의를 증진하고자 한다.

1. 서론

1.1. 연구배경

2010 년 이후 지속된 스마트 기술 연구는 모바일, 컴퓨터 등의 스크린 내부에 제한되어 있던 가상 세계를 현실 세계와 상호작용할 수 있게 만드는 수준에 이르렀다. 이에 현실과 가상 세계의 보다 자연스러운 결합을 위한 컴퓨터 그래픽스, 컴퓨터 비전 기술이 앞으로의 스마트 기술을 더욱 더 발전시킬 수 있는 연구 분야로 주목받고 있다.

이런 컴퓨터 그래픽스/비전 기술을 탑재한 기기 및 소프트웨어가 계속해서 출시되고, 메타버스와 같은 신조어들이 미디어에 지속적으로 노출되는 과정 속에서 해당 기술의 인지도가 높아졌다. 특히 인지도의 상승에는 다양한 게임 기기와 소프트웨어들이 큰 역할을 했으며 계속해서 개발이 진행되고 있다. 다만 다양한 기기와 게임이 출시되는 것에 비해 사용자의 편의는 고려하는 것은 뒷전이 되고 있다.

특히 컴퓨터 비전 기술이 사용된 Kinect 카메라, VR 과 같은 기기들은 큰 활동성을 전제로 한다. 이제까지의 플레이어들이 사용한 플랫폼이 PC, 모바일, 콘솔이라는 점을 고려하면, 부적절한 기기라고 할 수 있다. 따라서 정적인 환경에서의 게임 플레이에 익숙한 유저들에게는 체력적인 요구조건을 완화하여 비전 기술 기반 게임을 제공할 필요가 있다.

사용자의 편의를 위해서는 불편함도 고려해야한다. 가상-현실 상호작용 기기가 출시되면서 보완점으로 지적되는 것은 3D 멀미다. 3D 멀미는 일부 가상 환경에 노출되었을 때 어지러움, 두통을 느끼고 심할 경우 구토 증세를 보이는 것을 말한다. 이것은 사람에 따라 증세의 정도가 다르며, 환경에 대한 적응력이 좋은 경우에도 장시간동안 환경에 노출될 경우 어지러움을 느낀다. 이 3D 멀미

증세는 가상 공간 또는 1 인칭 시점 환경에서 주로 발생하므로, 컴퓨터 비전 기술 기반의 기기를 사용하는데 있어서 큰 문제가 된다.

이런 문제점들을 최소화할 수 있는 도구로 Leap Motion 을 제시할 수 있다. Leap Motion 특유의 정밀한 손 트래킹 기술을 살리면, 활동성은 낮으면서 좁은 공간에서도 쉽고 가볍게 컴퓨터 비전 기기를 사용할 수 있다. 한편, 가상 인터페이스가 아닌 PC 와 모니터를 활용하기 때문에 3D 멀미를 막을 수 있다.

그러나 Leap Motion 은 단독으로 사용되기보단 주로 VR 의 보조기기로 사용된 탓에 앞서 저술한 Leap Motion 의 장점이 제대로 드러나지 못했다. 따라서 PC 에서 즐길 수 있는 Leap Motion 용 게임 소프트웨어 개발을 통해 디바이스의 장점을 살리고, 이를 통해 개선된 가상-현실 상호작용 게임 환경을 제공해본다.

1.2. 연구목표

연구 배경에서 제시된 바와 같이 주된 목표는 Leap Motion 을 사용하는 게임 소프트웨어를 개발하여 개선된 가상-현실 상호작용 게임 환경을 만드는 것이다.

이것을 구현하기 위해서 첫 번째로, 플레이어가 피로를 느끼지 않도록 게임을 개발해야 한다. Leap Motion 을 사용하는 것만으로도 활동성이 낮지만, 장시간 사용하면 팔에 피로감이 오게 된다. 따라서 짧은 플레이 타임을 가진 게임을 목표로 한다.

다음으로, 3D 멀미를 최대한 예방하는 방식으로 게임을 구현해야 한다. PC 게임이 3D 멀미를 전혀 일으키지 않는 것은 아니다. 따라서 3D 멀미의 주된 원인인 빠른 화면 전환 및 회전을 없애고, 고정된 시점에서의 게임 진행으로 사용자의 어지러움을 최소화한다.

마지막으로 Leap Motion 이 가진 정밀한 손 트래킹의 장점을 선명히 담아야 한다. 기존 Leap Motion 기반의 게임은 기기의 고유 특성을 살리기 보다, 가상의 키보드를 사용하는 용도로 Leap Motion 을 활용했다. 하지만 가상 키보드를 사용한다면 물리 키보드를 사용하는 것과 큰 차이가 없고, 결국 컴퓨터 비전 기술을 사용하는 의미가 사라진다. 따라서 별도의 컨트롤러를 사용하기 보다, 플레이어의 손 그 자체를 컨트롤러로 사용하여 Leap Motion 의 장점을 살리고 가상-현실 상호작용 기기의 특성을 최대화한다.

2. 관련연구

2.1. Leap Motion

Leap Motion 은 손동작을 추적, 인식하여 컴퓨터를 제어하기 위한 장치다. 기기 내부에 탑재된 2 대의 카메라와 적외선 LED 센서가 손동작을 인식하는 방식으로, 손목부터 손가락 끝까지 뼈와 관절을 모델링한다. 단순히 손의 위치만을 인식할 경우 손의 일부가 가려졌을 때 손동작을 제대로

인식하지 못하는데, Leap Motion 의 경우 움직임에 대한 가상 모델을 통해 손의 일부분이 보이지 않더라도 높은 인식률을 유지할 수 있다.

2 대의 카메라와 적외선 센서가 있지만 8cm*3cm*1.3cm(길이*너비*높이)의 작은 크기로 좁은 공간에서도 사용할 수 있다. 손동작의 인식은 카메라로부터 10cm~60cm 의 간격을 권장하며, 최대 80cm 까지 인식한다.

2.2. Unity Engine

Unity Engine(이하 유니티)은 대표적인 게임 개발 엔진이다. 2D, 3D 그리고 VR/AR 게임 어플리케이션을 개발할 수 있으며, PC 와 모바일, 콘솔 기기 등 다양한 플랫폼으로 배포할 수 있다는 장점이 있다. 최근에는 게임 개발에 제한되지 않고 애니메이션 제작, 건축 설계, 공장 생산 라인 등의 다양한 분야에서 필요한 그래픽을 모델링하는데 사용하고 있다.

2.3. 기존 모션 인식에 관련된 기기 연구

2.3.1. Azure Kinect DK

Azure Kinect DK(이하 키넥트 또는 키넥트 카메라)는 컴퓨터 비전 기술과 음성을 제공하는 AI 센서가 포함된 개발자용 키넥트 카메라다. 이전에 개발되었던 xbox 용 키넥트보다 깊이 센서, 비디오 카메라 기술이 발전되었고, 원거리 음성을 포착하는 어레이 마이크 기술이 탑재되었다. 깊이 센서를 통해 3D 공간을 구현할 수 있으며, 촬영만으로 인체의 골격을 인식하여 각각의 관절 계층 구조 데이터를 이용하는데 용이하다. windows 환경에서 주로 개발 가능하며, 기존 xbox 용 키넥트 카메라와는 달리 게임 콘솔에는 연결이 불가능하다.

2.3.2. 닌텐도 스위치 액세서리 '링콘'

일반적으로 메인 컨트롤러 링콘, 링콘에 추가로 탑재하는 Joy-Con, 허벅지에 착용하여 사용하는 레그 스트랩의 3 가지 액세서리를 하나의 세트로 취급한다. 메인 컨트롤러인 링콘은 물리 센서를 통해 사용자가 주는 힘을 인식하여 소프트웨어에 전달하는 기능을 가지고 있다. 링콘에 추가로 장착하는 Joy-Con 은 닌텐도 스위치의 기본 컨트롤러로, 모션 IR 카메라의 적외선 장치를 통해 사용자의 건강 상태를 점검할 수 있는 장치이다. 허벅지에 착용하는 레그 스트랩의 경우 가속도 센서와 자이로 센서를 통해 하체의 움직임과 힘에 대한 인식을 담당한다.

2.3.3. PlayStation 카메라

Leap Motion, 키넥트 카메라와 같이 모션 인식 기능이 탑재된 카메라이다. 사용자의 움직임을 추적해서 PlayStation VR 및 PlayStation 과의 연동이 가능하며, 이를 통해 자신의 게임 플레이를 실시간으로 스트리밍할 수 있다. 또한 얼굴 인식 센서를 통해 특별한 조작 없이 PlayStation 의 전원을 켤 수 있다. 기기의 전원을 켠 후에도, PlayStation 카메라의 음성 인식 센서를 이용하면 게임을 켜거나 원하는 데이터를 검색할 수 있다.

2.4. 기존 모션 인식 기기의 한계 및 립모션과의 비교

2.4.1. Azure Kinect DK

컴퓨터 그래픽스/비전 기술을 직접적으로 사용하여 모션 인식을 한다는 점에서는 Leap Motion 과 같으며, 전신을 촬영 및 깊이 센서를 이용하여 많은 데이터를 이용할 수 있다는 점에서 Leap Motion 보다 더 우수한 부분도 있다. 그러나 Leap Motion 과 달리 엄지 손가락을 제외하고는 손가락 골격을 인식하지 못하며, 때로는 엄지 손가락과 새끼 손가락을 구분하지 못할 정도로 상세한 동작에 대한 인식률이 낮다.

2.4.2. 링콘

링콘의 경우 직접적으로 컴퓨터 그래픽스/비전 기술을 활용했다고 하기 어렵다. 3 가지의 컨트롤러를 사용하지만, 각각의 컨트롤러는 물리적인 압력, 위상 변화에 대해 반응하는 것일 뿐 Leap Motion 과 같이 공간과 물체에 대해 인식하고 반응하는 것이 아니다.

반면, Leap Motion 은 링콘과 달리 직접적으로 카메라로 손을 촬영하고 인식한다. 이 과정에서 적외선 LED 센서를 통해 손동작을 구체적으로 인식하기 때문에, Leap Motion 을 사용하는 매 순간이 컴퓨터 그래픽스/비전 기술을 직접적으로 사용하는 순간이다.

2.4.3. PlayStation 카메라

모션 인식이 가능하다는 점에서는 Leap Motion, 키넥트와 같지만, 골격정보는 얻지 못한다. 특히 깊이 센서를 통해 3D 공간 및 모델을 구현할 수 있는 키넥트, 가상 모델링을 통해 입체 모델을 구현할 수 있는 Leap Motion 에 비해 PlayStation 카메라는 3D 공간을 구현할 수 없다.

그 외에도 낮은 인식률이 문제점으로 지적되며 사용을 위해서는 PlayStation 기기가 필수라는 점에서 Leap Motion 에 비해 접근성이 부족하다.

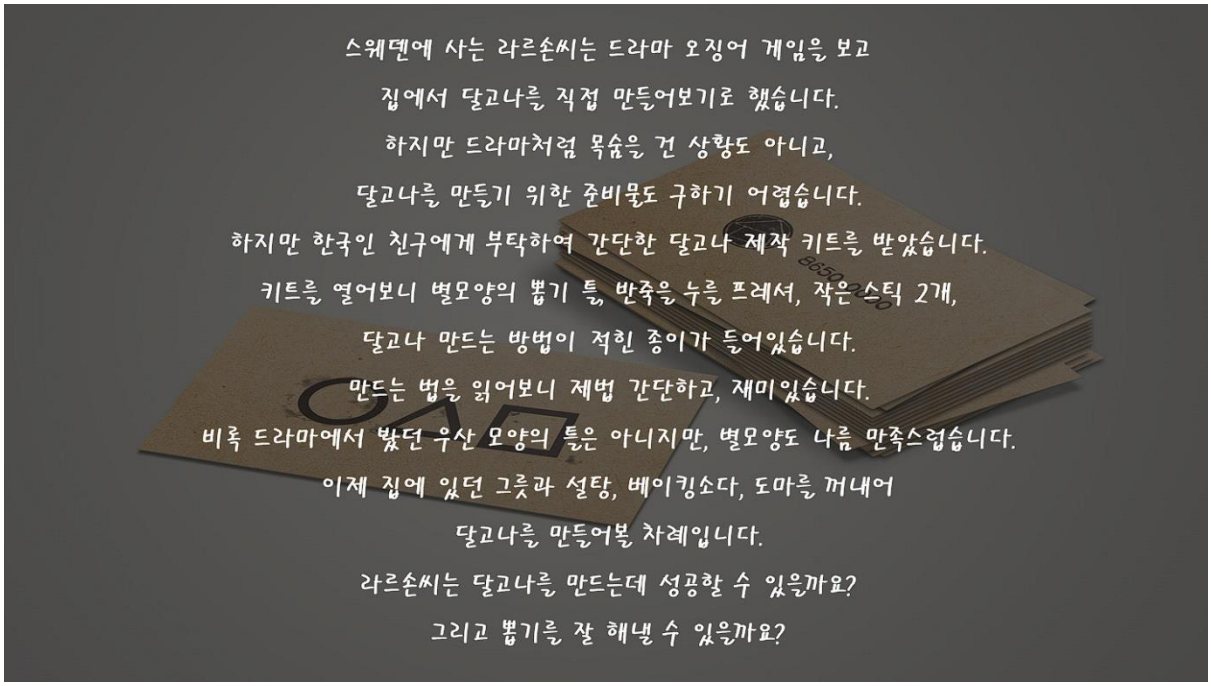
3. 프로젝트 내용

3.1. Game Concept

Leap Motion 기기의 특성과 연구 목표를 고려했을 때, 게임은 오직 플레이어의 손만 컨트롤러로 사용해야 한다. 동시에 짧은 플레이 타임과 정적인 게임을 목표로 하기 때문에, 재미를 얻는 것은 쉽지 않다. 따라서 목표에 부합하는 게임을 개발하면서 유저에게 재미를 주려면, 그만큼 흥미를 끌 컨셉트가 필요하다. 이에 인기를 끌었으며, 계속해서 주목을 받고 있는 드라마 '오징어 게임'에 등장하는 미니 게임을 소재로 삼는다. 해당 드라마에서 연출된 게임 중 전신을 사용하지 않고 손만 사용하는 게임은 2 가지, 달고나 만들기와 구슬치기가 있다.

각각의 게임이 가진 성격을 분석하면, 달고나는 누군가와 경쟁하는 방식이 아니라 개인별로 달고나의 모양을 훼손하지 않고 분리하는 게임이며, 구슬치기는 여러 방법이 있지만 결국 누군가와 경쟁을 전제로 하는 대전게임이다. Leap Motion 이 가진 손 트래킹 기술이 정밀한 것은 사실이지만, 완벽한 인식을 보장할 수는 없다. 따라서 순간적인 판단이나 컨트롤이 중요시되는 대전게임 방식은 인식 오류에 따른 높은 난이도가 요구될 수 있으므로 유저의 편의를 고려하는 이번 연구에 어울리지 않는다. 이에 개발할 게임의 컨셉트는 '달고나 만들기' 게임으로 한다.

3.2. Game Story



[사진 1] 게임 스토리

스토리를 중심으로 하는 게임이 아니므로, 게임 스토리는 간단하게 구성한다. 드라마 속에서 등장한 달고나를 본 어떤 외국인이, 직접 집에서 달고나를 만들어보려고 하는 상황이다. 플레이어는 이 외국인의 입장에서 달고나를 성공적으로 제작하고 모양을 뽑아내면 된다.

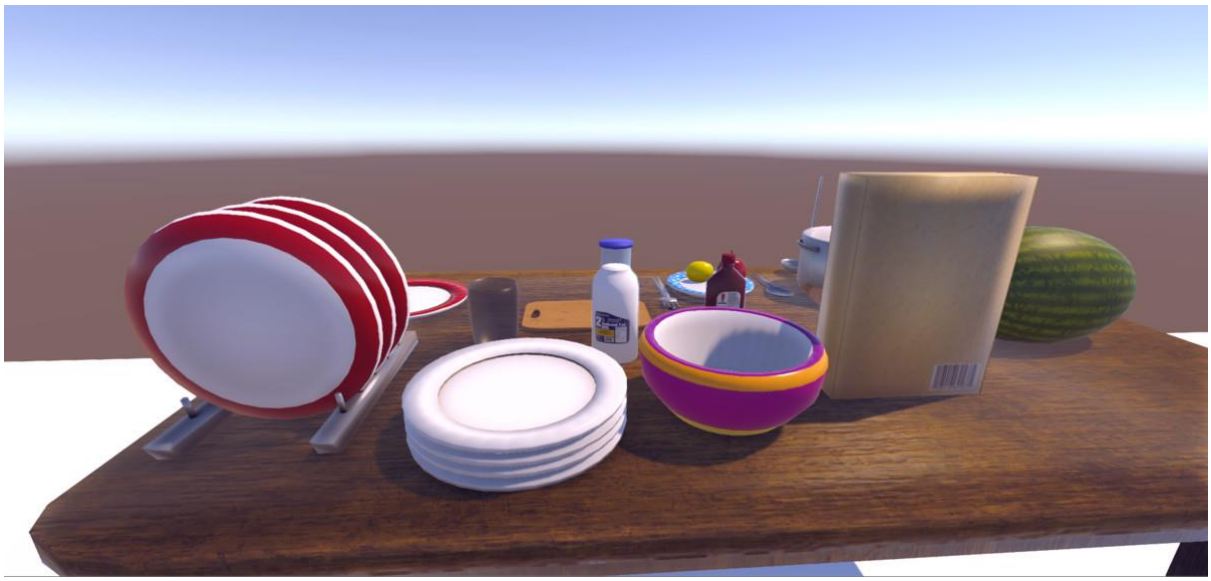
3.3. Game Design

3.3.1. World asset

[사진 2] DevDen ArchViz Vol 1 – Scotland

달고나를 만드는 주방의 느낌과 북유럽 주방의 느낌을 모두 살릴 수 있는 에셋을 사용했다. 단, 플레이시 화면 구도와 게임 용량 축소를 위해 prefab 활용을 최소화했다.

3.3.2. Object asset



[사진 3] **Food and Kitchen Props Pack**

달고나를 만드는 과정을 위해 부엌의 소품을 표현할 에셋을 사용했다.

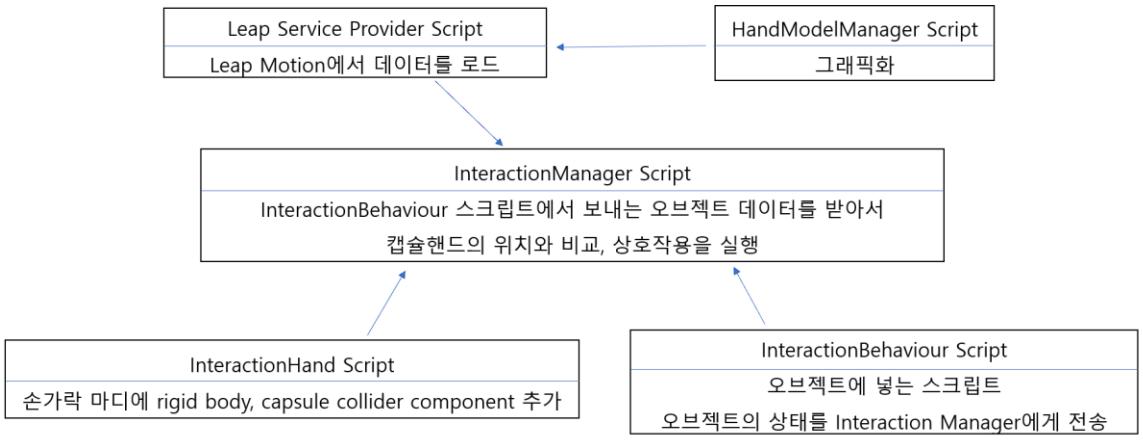
3.3.3. Effect UI asset

[사진 4] **Action RPG FX**

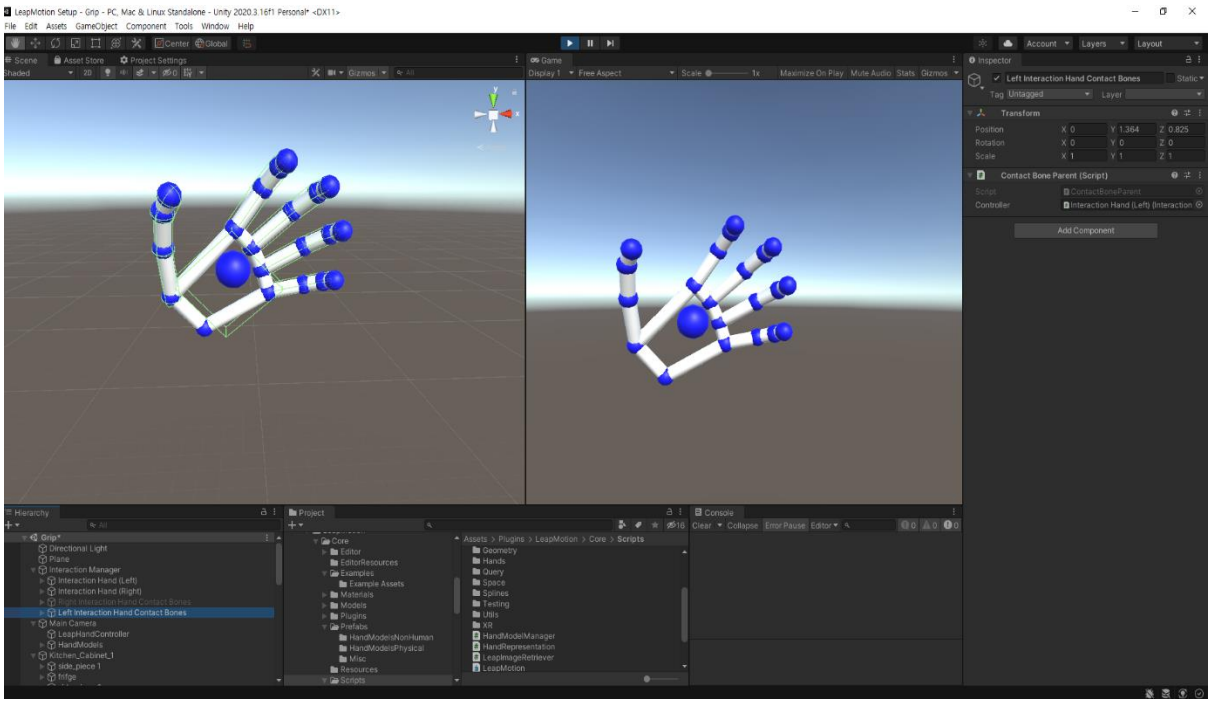
게임 중 플레이어에게 경고 상황을 알리기 위한 이펙트 에셋을 사용했다.

3.4. Game Mechanism

3.4.1. Leap Motion SDK



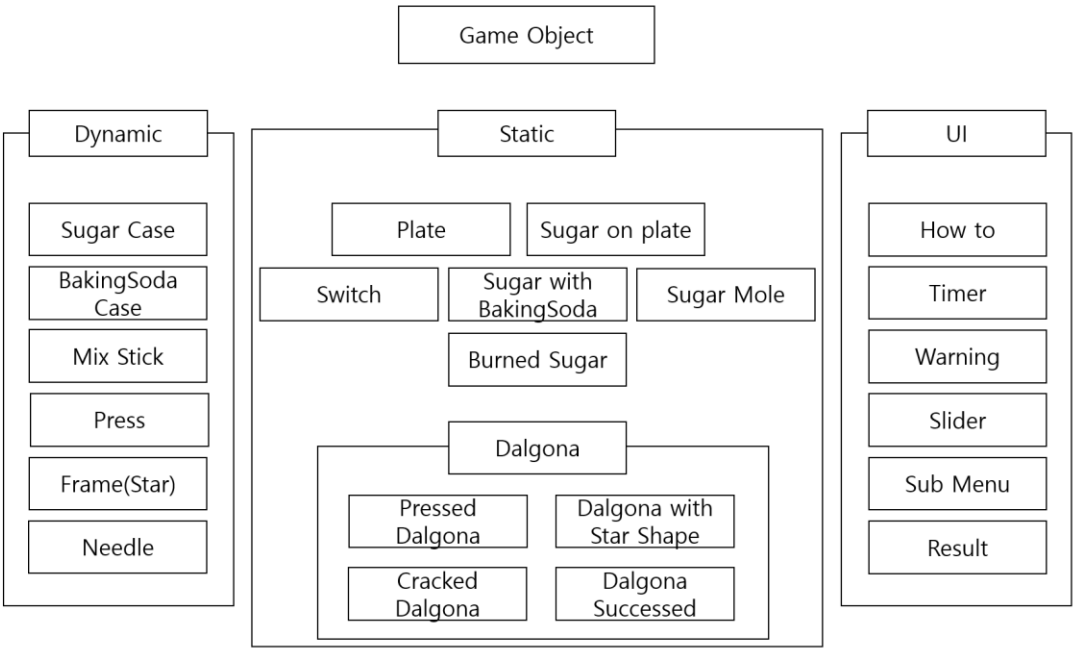
[그림 1] Leap Motion SDK의 작동 간이 구조



[사진 5] 유니티 에디터에서 플레이어의 손을 구현한 모습

플레이어의 손을 그대로 컨트롤러로 사용하기 위해서 Leap Motion 에서 공식적으로 지원하는 유니티용 SDK 를 사용했다. 이 SDK 에서 주로 사용되는 Script 와 역할을 간략하게 정리하면 [그림 1]과 같다.

3.4.2. Game Object



[그림 2] 필요한 게임 오브젝트 목록

[그림 2]는 게임에 필요한 오브젝트를 정리한 것으로, 유저가 상호작용하며 물체를 움직여야 하는 경우 Dynamic 으로, 상호작용은 하지만 물체의 Transform 은 변화하지 않는 경우 Static 으로 구분하였다. 또한 게임 진행을 위한 주요 UI 를 정리했다.

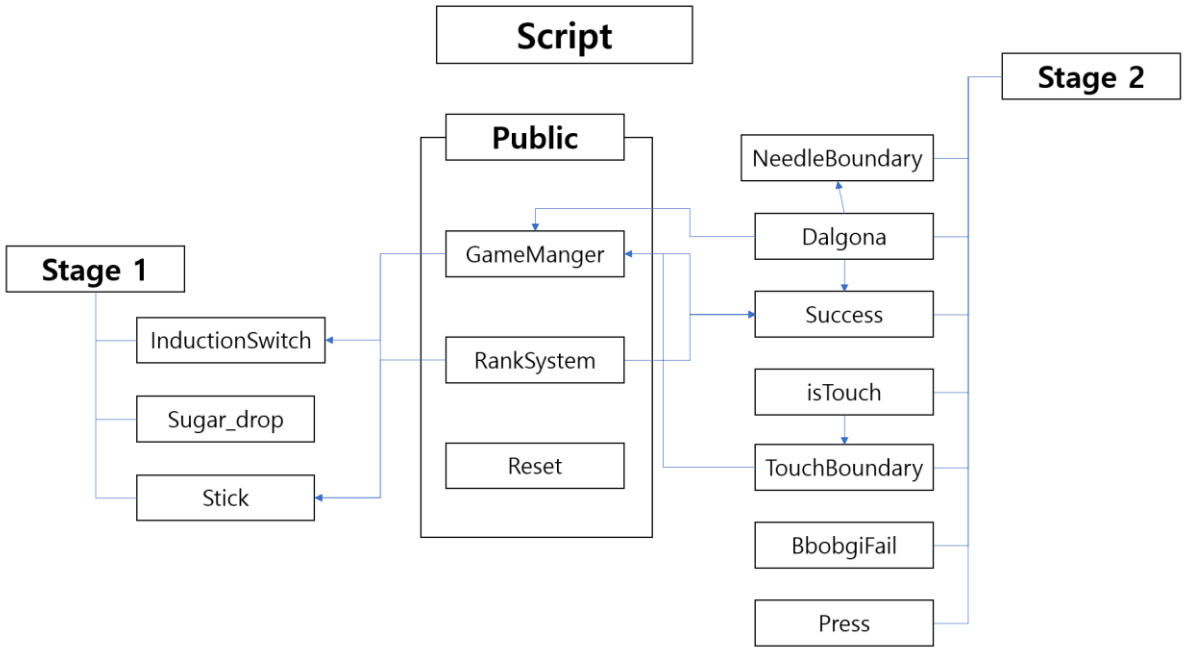
Dynamic 성격을 가진 오브젝트 중에서 Sugar Case 와 BakingSoda Case 는 달고나 반죽을 만들기 위해 플레이어가 집어서 그릇 위에 각각 설탕과 베이킹 소다를 뿌리는데 사용된다. 이에 따라 그릇을 인식하기 위한 boundary 를 제작하고, 각 오브젝트에는 Sugar 태그와 BakingSoda 태그를 설정한다. Mix Stick 은 그릇의 설탕을 젓는데 사용한다. 막대를 젓는 것을 인식하기 위해서 boundary 를 따로 만들고, Stick 태그를 설정한다. Press 는 완성된 달고나 반죽을 누르는데 사용한다. Press 를 달고나가 인식하게 만들기 위해 Press 태그를 추가한다. Frame 오브젝트는 별모양의 틀로, press 로 누른 달고나에 모양을 찍기 위해 사용한다. 달고나가 Frame 을 인식하게 만들기 위해 Frame 태그를 설정한다. 마지막으로 Needle 오브젝트는 달고나에서 모양을 뽑아내는데 사용한다. Frame 오브젝트로 달고나를 누르고 남은 자국과의 접촉을 인식하기 위해서 Needle 태그를 설정한다.

Static 오브젝트 중에서 Plate 오브젝트는 설탕이 담기는 오브젝트이며, 설탕을 뿌렸을 때 Sugar on plate 오브젝트를 활성화하여 설탕이 그릇에 담긴 모습을 표현한다. Switch 오브젝트는 부엌 에셋을 활용하되, 플레이어의 손을 인식할 수 있도록 설정한다. Mix Stick 을 이용하여 일정 횟수 이상 설탕을 저으면, Sugar Mole 오브젝트를 활성화시켜서 설탕의 상태를 플레이어에게 직관적으로 전달한다. 이 오브젝트는 알갱이의 효과를 위해 particle system 을 활용한다. 베이킹 소다를 넣은 후에는 색이 변하므로 Sugar with BakingSoda 오브젝트를 활성화해서 설탕의 상태를 전달한다. 또한 플레이어가

설탕을 젓거나 일정 시간이 지나면 자연스럽게 크기를 조정하여 반죽이 부풀어 오르고 있음을 표현한다. Burned Sugar 는 설탕을 태웠을 경우 활성화되도록 검은 색의 설탕을 표현한다. 완성된 달고나 반죽 오브젝트는 Press 로 눌렀을 때 눌린 모습을 표현하는 Pressed Dalgona, Frame 으로 모양이 찍힌 상태를 연출할 Dalgona with Star Shape, 뽑기가 실패했을 때 금이 간 달고나의 모습을 표현하기 위한 Cracked Dalgona, 성공적으로 달고나의 모양을 뽑은 것을 표현할 Dalgona Successed 오브젝트의 4 가지 형태를 제작한다.

UI 는 각각의 상황에서 플레이어가 어떤 행동을 해야 하는지 설명할 How to 가 필요하다. Static 오브젝트와 dynamic 오브젝트의 활성화 상태를 통해 안내하는 UI 를 변경할 수 있어야 한다. Timer UI 는 게임 시작과 함께 플레이 시간을 기록하도록 제작한다. Warning UI 는 플레이어가 달고나를 만들면서 실패할 수 있는 상황에 경고를 주기 위해 만든다. 실패 상황은 설탕을 태우는 경우, 달고나 반죽이 그릇에서 흘러 넘치는 경우, 달고나 뽑기를 할 때 달고나가 부서지는 상황이 있다. Slider UI 는 플레이어가 미션의 진척도를 직관적으로 알 수 있게 전달하는 용도로 제작한다. Stick 으로 설탕을 저을 때 얼마나 저었는지 Mix slider 로 표기하고, 설탕이 타는 정도를 Burning slider 를 표기하며, 달고나 뽑기를 할 때는 눌린 자국을 검은색으로 변하게 해서 유저의 진척도 이해를 돕는다. Sub Menu 는 게임 일시정지, 재시작, 게임 종료를 선택할 수 있게 제작한다. Result 는 게임 결과를 전달하도록 클리어 시간과 그 시간에 따른 랭크, 최고 기록을 표기하며, 이전/다음 스테이지로 넘어가는 것과 재시작, 게임 종료를 선택할 수 있게 제작한다.

3.4.3. Scripts



[그림 3] Stage별 Script 및 Script간 참조 관계도

게임 개발에 필요한 Script 는 [그림 3]과 같이 stage 와 공통 script 를 기준으로 구분한다. 먼저 공통 script 에는 Timer 와 같은 게임 공통의 기능들을 동작 시키기 위한 코드를 담는다. GameManager script 는 Update 함수를 통해 매 프레임마다 Timer 의 시간을 조정하고, 언제든지 서브 메뉴를 불러올 수 있도록 menu 오브젝트를 on/off 할 수 있게 작성한다. 그리고 menu 에서 재시작, 게임 종료를 위한 button 의 on click event 를 함수로 제작한다. 또한 일부 UI 의 출력을 위해 Dalgona script 와 TouchBoundary script 를 참조하며, slider 역시 본 script 를 통해 관리한다. Rank System script 는 게임 클리어 시간에 따라 랭크를 구분하는 기능을 담당하며, 랭크는 S, A, B, C, D 로 나눈다. Reset script 는 Dynamic object 가 플레이어와 상호작용 과정에서 손에 닿지 않는 먼 곳으로 갈 경우 위치를 초기화 시키는 코드를 작성한다.

Stage 1 에서 InductionSwitch script 는 Induction 가열 스위치의 on/off 기능을 담당한다. 또한 설탕이 그릇 위에 있을 때 스위치가 켜지면 burning count 를 증가시키고, 일정 수치를 넘기면 경고를 안내하고 이후에는 타버린 설탕 오브젝트를 활성화하여 게임을 재시작하게 만든다. Sugar_drop script 는 SugarCase 또는 BakingSoda Case 가 drop boundary 에 충돌했을 때 각각의 count 를 증가시키고, 일정 횟수를 넘겼을 때 설탕 오브젝트 또는 Sugar with Baking Soda 오브젝트를 활성화시킨다. Stick script 는 기본적으로 mix count 를 통해 플레이어가 설탕을 얼마나 저었는지 판정하는 역할을 한다. 내부적으로 count 를 계산해서 설탕의 상태와 UI 를 변화시키며, 베이킹 소다를 넣은 후에 일정 count 에 도달하면 게임 clear 를 판단하고 게임 결과창을 출력한다.

반면 베이킹 소다를 넣고 일정 시간이 경과할 동안 요구 count 를 달성하지 못하면 경고를 출력하고, 일정 시간 후에 overflow 로 인한 게임 오버 상태를 출력한다.

Stage 2 는 Press script 를 가장 먼저 활용하는데, 이 script 는 달고나와 press 가 충돌할 때 달고나 반죽이 서서히 눌리도록 코드를 작성한다. 이후 눌린 달고나 반죽에 Frame 오브젝트를 올리면 달고나 위의 boundary 가 isTouch script 와 TouchBoundary script 를 통해 오브젝트를 인식하고, Frame 의 Transform 에 맞게 별 모양을 남긴다. Dalgona script 는 이 상태에서 달고나를 손가락으로 터치했을 때 화면의 구도를 바꾸게 만든다. 뽑기 과정의 편의를 위해 달고나의 크기는 키우고 카메라의 위치와 각도를 바꾼다. 그리고 Needle 오브젝트를 활성화시킨다. NeedleBoundary script 는 바늘이 달고나에 남은 별 모양의 자국과 충돌할 때마다 stack 을 증가시켜서 일정 횟수에 도달하면 자국이 검은색으로 변하게 만든다. 동시에 BbobgiFail script 에는 failstack 이라는 변수를 만들어서, 달고나의 어떤 부분이든 바늘과 충돌할 때마다 stack 이 쌓이게 만든다. 그래서 너무 많은 충돌이 일어날 경우 Cracked Dalgona 오브젝트를 활성화시키고 게임 오버 상태를 출력하게 만든다. 마지막으로 Success script 는 달고나의 별 모양 자국이 모두 검은 색이 되었을 때 다시 카메라를 조정하고, Dalgona successed 오브젝트를 활성화한 후에, 게임 결과창을 출력하도록 작성한다.

3.5. 개발 기간

총 개발 기간은 2021 년 9 월 30 일부터 2021 년 12 월 13 일까지로, 10 월 동안은 stage 1 의 개발, 11 월 동안은 stage 2 를 개발했다. 12 월에는 두 stage 의 버그 수정과 UI, 편의성 향상을 개발했다.

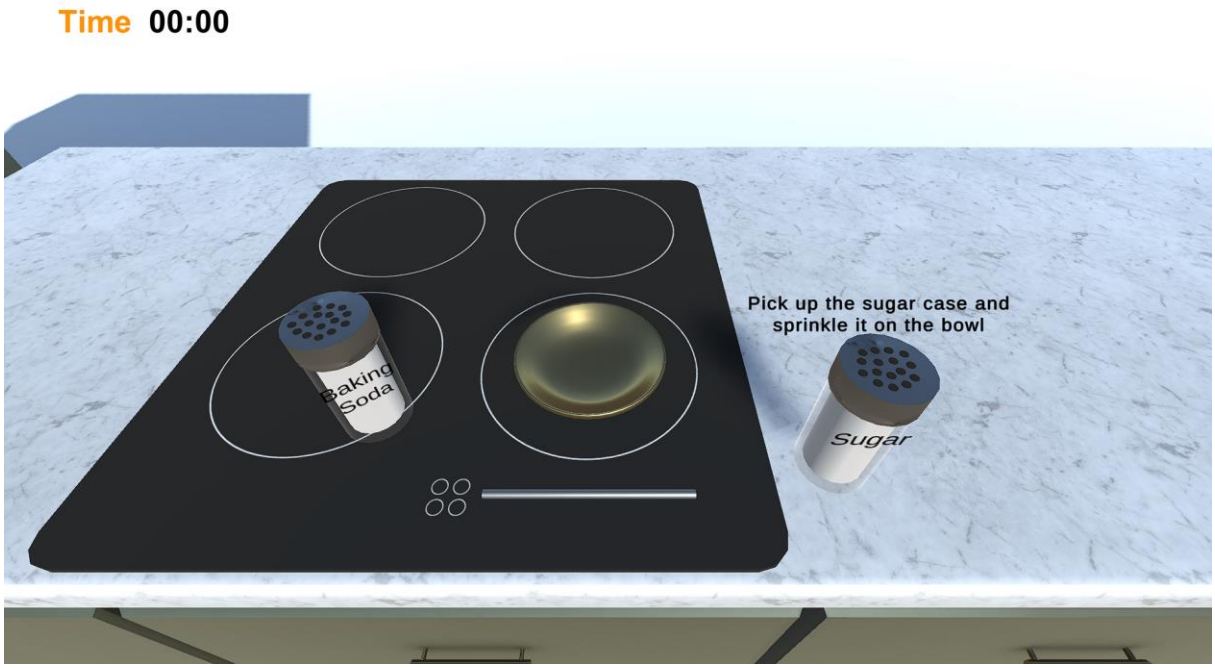
3.6. 기타 사항

게임의 밸런스는 게임에 있어서 중요한 요소지만, 본 연구는 Leap Motion 을 이용한 조작과 편리한 인터페이스 및 게임 환경 제공에 중점을 두기 때문에, 밸런스는 게임 진행에 방해가 되지 않는 수준으로만 조정하고 상세한 레벨까지는 고려하지 않기로 한다.

4. 프로젝트 결과

4.1. 게임 구현

4.1.1. Stage 1



[사진 6] Stage 1 시작화면

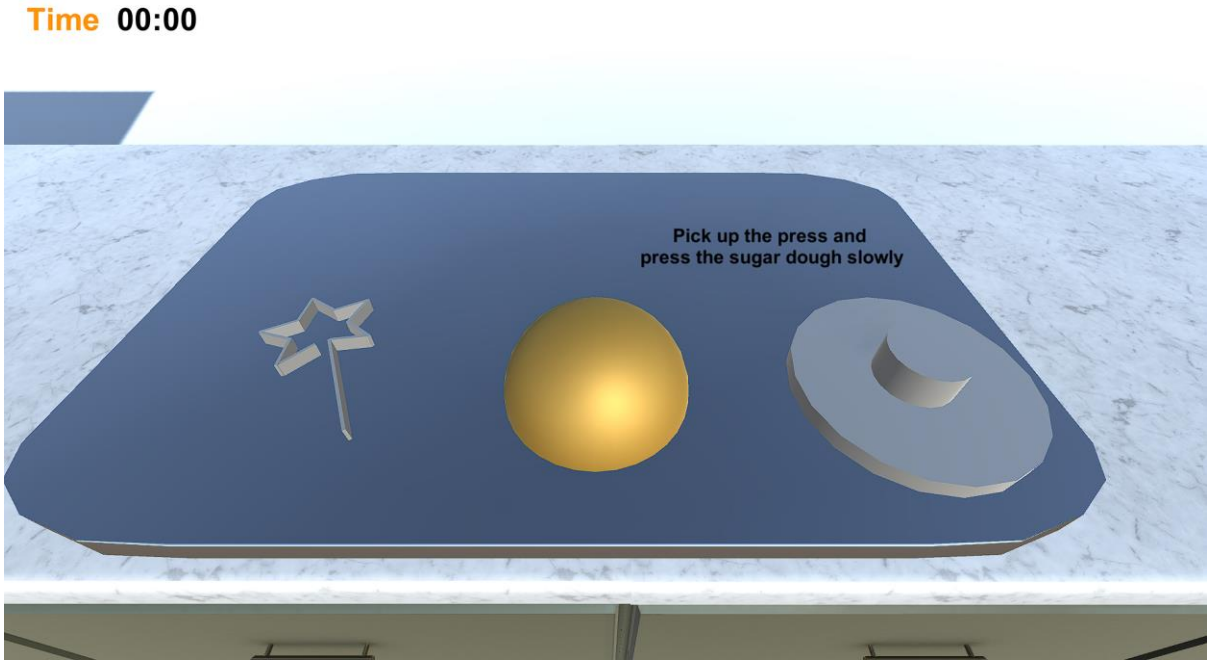
Stage 1 을 시작하면 먼저 UI 에 맞게 Sugar Case 를 집어서 그릇에 설탕을 뿌린다. 설탕을 뿌릴 때마다 미리 설정된 drop boundary 에서 설탕을 뿌리는 이펙트가 나오며, 5 회 이상 설탕을 뿌리면 그릇의 설탕이 활성화된다. 설탕이 활성화되면 UI 가 변하며 스위치를 켜야 한다. 스위치는 원래 고정되어 있지만, 플레이어의 손가락을 인식하기 위해서 InteractionBehaviour script 를 component 로 넣어야 했다. 그 뿐만 아니라 Leap Motion SDK 의 InteractionHand script 의 손가락의 특성을 부여하는 initContactBone 함수에서 손가락에 Finger 태그를 지정해주는 코드를 추가했다. 스위치를 켜고 난 후에는 Burning Count 와 Mix count 가 게이지 UI 를 통해 안내되며, 막대를 집어서 Burning 게이지가 완전히 채워지기 전에 설탕을 저어 Mix 게이지를 채워야 한다. 그릇의 바닥 부분에는 4 개의 mix boundary 가 있어서, 막대가 각각의 boundary 를 충분히 터치해야 Mix 게이지가 완전히 채워지도록 구현했다. 이것은 그릇의 한 부분만 젓는 것이 아니라 전체적으로 저어서 실제로 설탕을 젓는 모습을 연출한 것이다. 이후에는 스위치를 끄고, Baking Soda Case 를 집어서 설탕에 뿌린다. 이것 역시 drop boundary 를 통해 collider 의 충돌을 인식한다. 베이킹 소다를 4 회 이상 뿌리면 설탕이 갈색으로 변하면서 Mix 게이지가 나오고 다시 막대로 저어야 한다. 설탕을 저어서 Mix 게이지가 1/3 씩 채워질 때마다 설탕의 높이가 상승하며, 최대 높이에서 스테이지가 클리어 된다. 단, 충분히 설탕을 젓지 않아도 시간이 경과하면서 설탕의 높이는 올라간다. Mix

게이지를 시간 내에 채우지 못하더라도 2 번의 기회를 더 주어 플레이어가 빠르게 설탕을 저으면 게임을 클리어할 수 있게 했다. 그러나 시간이 지나 설탕의 높이가 충분히 올라간 상황에서도 Mix 게이지를 완전히 채우지 못하면 overflow 로 인한 게임 오버가 된다.

게임을 성공적으로 클리어하면 결과창이 나온다. Stage 1 의 랭크는 클리어 시간이 각각 25 초 이내에서 S 랭크, 40 초 이내에서 A 랭크, 1 분 이내에서 B 랭크, 1 분 20 초 이내에서 C 랭크, 그 이상은 D 랭크를 받게 된다.

4.1.2. Stage 2

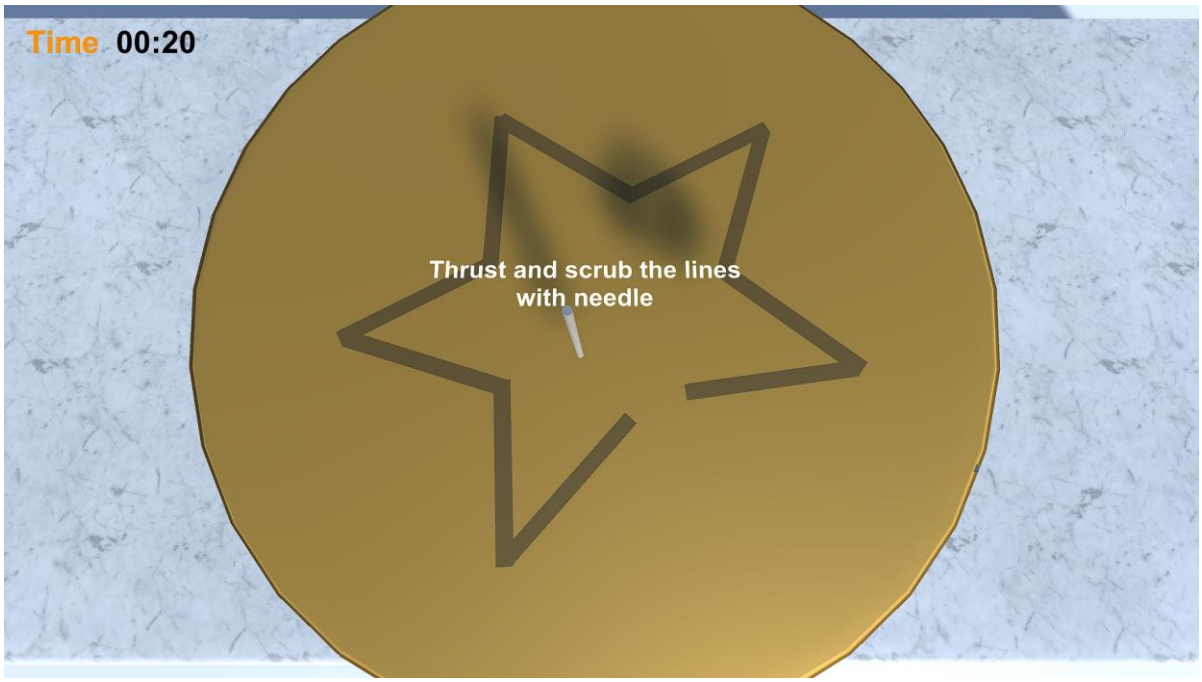
4.1.2.1 반죽 누르기



[사진 7] Stage 2 시작화면

Stage 2 는 먼저 press 오브젝트를 들어서 달고나 반죽을 눌러야 한다. 달고나 반죽은 press 가 닿으면 미리 설정해둔 애니메이션에 따라 서서히 크기가 작아진다. 특히 애니메이션의 재생 속도를 조절하여 달고나의 크기가 작아질수록 press 를 충분히 누르고 있어야 달고나가 눌리도록 구현했다. 그 다음엔 별 모양의 frame 을 들어서 달고나를 누른다. 별 모양이 달고나에 제대로 새겨지는 것을 유도하기 위해, frame 이 달고나 내부에 완전히 들어올 때에만 모양이 새겨지도록 touch boundary 의 구간을 나누고 위치를 조정했다. 이후에는 UI 안내에 따라 손가락으로 달고나를 터치하여 화면 구도를 바꿀 수 있게 구현했다.

4.1.2.2 모양 뽑기



[사진 8] 뽑기 과정 초기화면

화면 구도가 바뀐 후에는 바늘로 모양을 뽑아내야 한다. 다만 바늘은 크기가 작아서 플레이어와의 상호작용에 에러가 나거나 상호작용 그 자체를 제대로 인식하지 못한다. 따라서 바늘보다는 막대의 형태로 만들고 크기를 키워서 상호작용에서의 난이도를 조정했다.

그럼에도 막대의 크기가 작은 탓에, 상호작용이 가능한 capsule collider 를 크게 키우고 물리 영역이 작용하는 capsule collider 는 크기에 맞게 조정했다. 이런 이유로 달고나와 충돌 시 fail stack 이 쌓이는 속도가 2 배가 되었고, 그에 따라 게임 오버의 failstack 기준 역시 대폭 증가시켰다. 달고나가 깨지지 않게 모든 선을 찌르거나 긁으면 검은색으로 선이 변하는데, 모든 선이 검은색이 되면 게임이 클리어 되며 완성된 별 모양의 달고나를 볼 수 있게 구현했다. 완성된 달고나는 약간의 입체감을 살릴 수 있도록 material 을 적용했다. 이후에는 결과창이 나오며 이전 스테이지로 돌아갈지, 재도전할지, 게임을 종료할지 고를 수 있다.

Stage 2 의 랭크는 클리어 시간이 각각 1 분 이내에서 S 랭크, 1 분 20 초 이내에서 A 랭크, 1 분 40 초 이내에서 B 랭크, 2 분 이내에서 C 랭크, 그 이상은 D 랭크를 받게 된다.

4.2. 게임 테스트 및 평가

본 연구에서 세부 목표로 제시했던 3 가지는 짧은 플레이 타임, 정적인 게임, 플레이어의 손을 컨트롤러로 활용하는 것이었다. Leap Motion SDK 를 활용하여 플레이어의 손을 게임에 이식했기 때문에 3 번째 목표를 달성했고, S 랭크 기준 1 스테이지는 25 초, 2 스테이지는 1 분 내에 완료할 수 있으므로 짧은 플레이 타임의 목표를 달성했다. 또한 카메라의 전환이 별도로 존재하지 않는 정적인 게임으로 개발하여 목표를 모두 달성하였다.

5. 결론

5.1. 기대효과

Leap Motion 을 활용하여 게임을 즐길 수 있다. 특히, 기존의 모션 인식 기기들과는 달리 역동적인 활동이 요구되지 않아 육체적인 피로의 부담, 번거로움 없이 게임을 켜고 끌 수 있으며, 3D 멀미로 인해 가상-현실 상호작용 기기를 이용한 게임 이용이 어려운 사용자도 비교적 쉽게 모션 인식 기술을 체험할 수 있다.

또한 기존에 사용하는 PC 와 모니터 디스플레이를 사용해서 게임을 즐길 수 있어 Leap Motion 만 있으면 쉽게 게임을 즐길 수 있다. 또한 Leap Motion 외의 추가적인 비용을 필요로 하지 않는다. 그 외에도 도트 그래픽 사용으로 인해 저사양 환경에서도 게임을 구동할 수 있다는 점 역시 사용자에게 편리한 점으로 작용하여, 누구나 쉽게 환경에 상관없이 게임을 즐길 수 있다.

5.2. 추후 연구 방향

달고나 메이커는 물체와의 상호작용을 허용하지만, 물리적인 충돌이 완벽하지는 않다. Leap Motion 이 손의 움직임을 트래킹하고, 그 데이터를 유니티 프로그램에 보내주는 과정에서 게임 내의 물리 충돌로 인해 게임과 실제 플레이어의 손의 위치가 어긋나는 것을 방지하기 위해서이다. 이것은 플레이어의 몰입도를 해칠 수 있으므로, 오브젝트와 충돌했을 때 오브젝트를 뚫고 들어가지 않도록 물리적 특성을 수정할 필요가 있다.

추가적으로 Leap Motion 의 카메라가 플레이어의 손을 완벽하게 인식하지 못하여 왼손과 오른손을 구별하지 못하거나, 손이 뒤집히는 상황이 간헐적으로 나타난다. 이 경우를 완화하기 위해 여러 번의 테스트를 걸친 결과, 양손을 모두 Leap Motion 기기 위에 노출하면 인식률이 높아지므로 플레이어가 양손을 사용해야하는 상황을 연출하거나 UI 를 통한 안내를 추가할 필요가 있다.

또한 제작한 게임은 별 모양의 프레임 밖에 없지만, 다른 모양의 프레임을 더 추가하여 플레이어에게 다양한 모양의 달고나 뽑기를 시도할 수 있도록 지원한다.

6. 참고문헌

[1] Azure Kinect DK: https://azure.microsoft.com/ko-kr/services/kinect-dk/#overview

[2] Nintendo Switch – Ring Fit: https://nintendo.co.kr/software/switch/ring/about/index.html

[3] PlayStation Camera: https://www.playstation.com/ko-kr/accessories/playstation-camera/

[4] Leap Motion: https://www.ultraleap.com/tracking/

[5] https://assetstore.unity.com/packages/3d/environments/urban/devden-archviz-vol-1-scotland-158539

[6] https://assetstore.unity.com/packages/3d/props/food-and-kitchen-props-pack-85050

[7] https://assetstore.unity.com/packages/vfx/particles/action-rpg-fx-38222