**GOLDBERG MACHINE**

TEAM GPS

김진수, Dept. of Computer Science and Engineering, POSTECH

문민재, Dept. of Computer Science and Engineering, POSTECH

이화윤, Dept. of Computer Science and Engineering, POSTECH

1. **Project Description**

Explanation of Goldberg machine and project

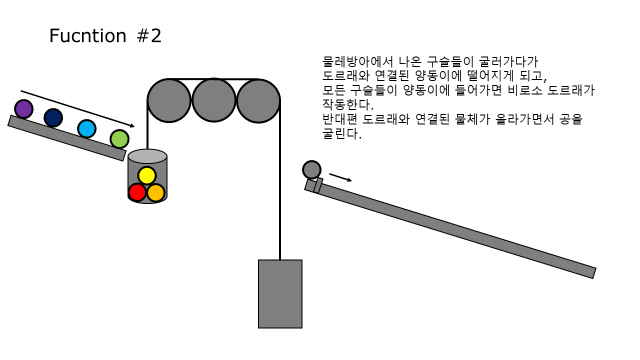
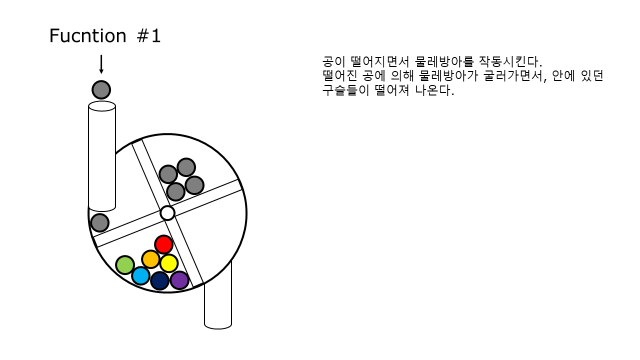
1. **Programming Environment**
2. **Design and Implementation**
   1. **Object Generation**

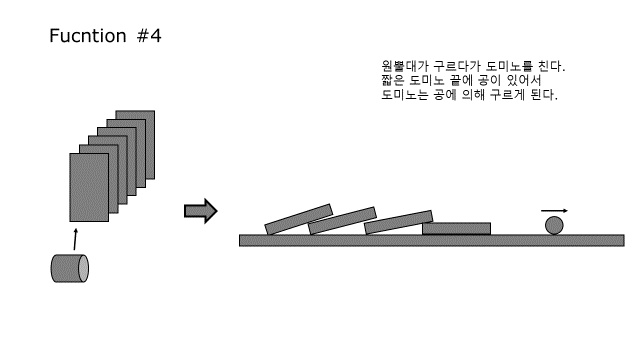
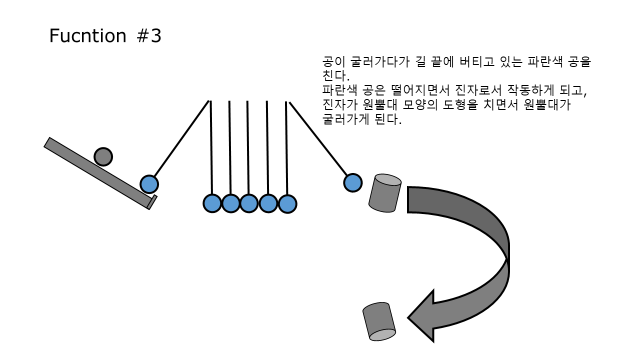
진수

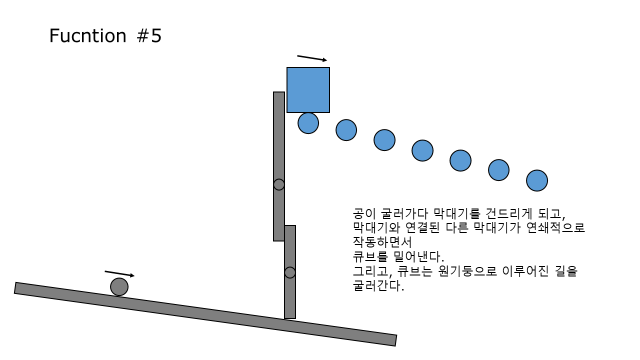
* 1. **Magnetic Force (Part 1)**

화윤

* 1. **Complex Objects (Part 2)**
     1. Design







Function 1:

Function 2:

Function 3:

Function 4:

Function 5:

Function 6:

* + 1. Implementation

Function 1: 물레 오브젝트는 우리가 원하는 디자인의 적절한 애셋을 찾기 어려워 Blender를 이용하여 직접 만들었고, 기능 자체는 디자인과 동일하게 구현되었다.

Function 2: 양동이에 공이 가득 차면 반대편에 달려 있던 오브젝트가 올라오면서 공을 건드려 다음 function으로 넘어가게 하는 대신, 양동이에 공이 가득 차면 케이블이 끊어지면서 반대편에 매달린 오브젝트가 떨어져 다음 기능으로 넘어가게끔 구현되는 등 구현의 한계로 인해 디자인했던 바와 일부 다르게 구현되었다. 도르래에 사용되는 케이블의 구현은 Unity의 Hinge joint 기능을 사용하여 구현되었다.

Function 3: 진자 여러 개가 연속되어 붙어있는 뉴턴의 요람 대신, 진자 한 개만을 사용하여 구현되었고, 마지막에 굴러가는 도형이 원뿔대에서 원기둥으로 바뀌는 등 디자인했던 바와 일부 다르게 구현되었다. 진자에 사용되는 케이블의 구현은 Unity의 Hinge joint 기능을 사용하여 구현되었다.

Function 4: 유니티의 기본 도형들을 사용하여 디자인과 동일하게 구현되었다.

Function 5: 유니티의 기본 도형들을 사용하여 디자인과 동일하게 구현되었다.

Function 6: 디자인과 동일하게 구현되었다. 마지막 대포 및 포탄의 경우, 직접 구현하기보다는 외부 애셋을 사용하여 더 멋있게 구현되었다.

* 1. **Wind and Air Resistance (Part 3)**

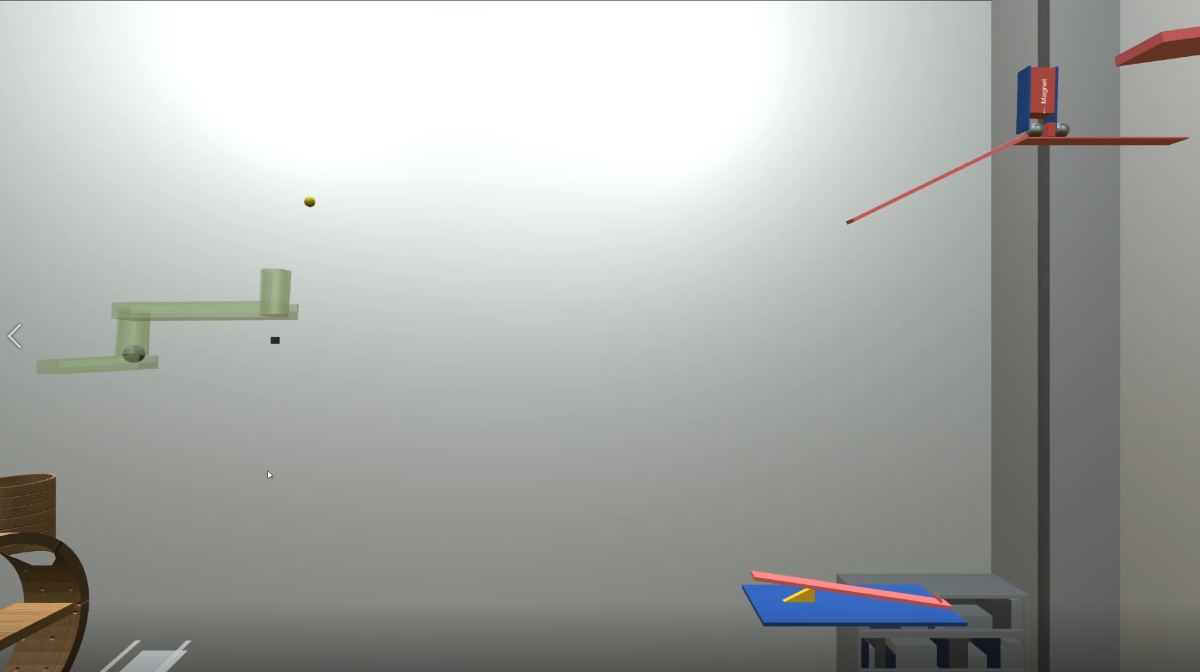
진수

* 1. **Camera Movement**

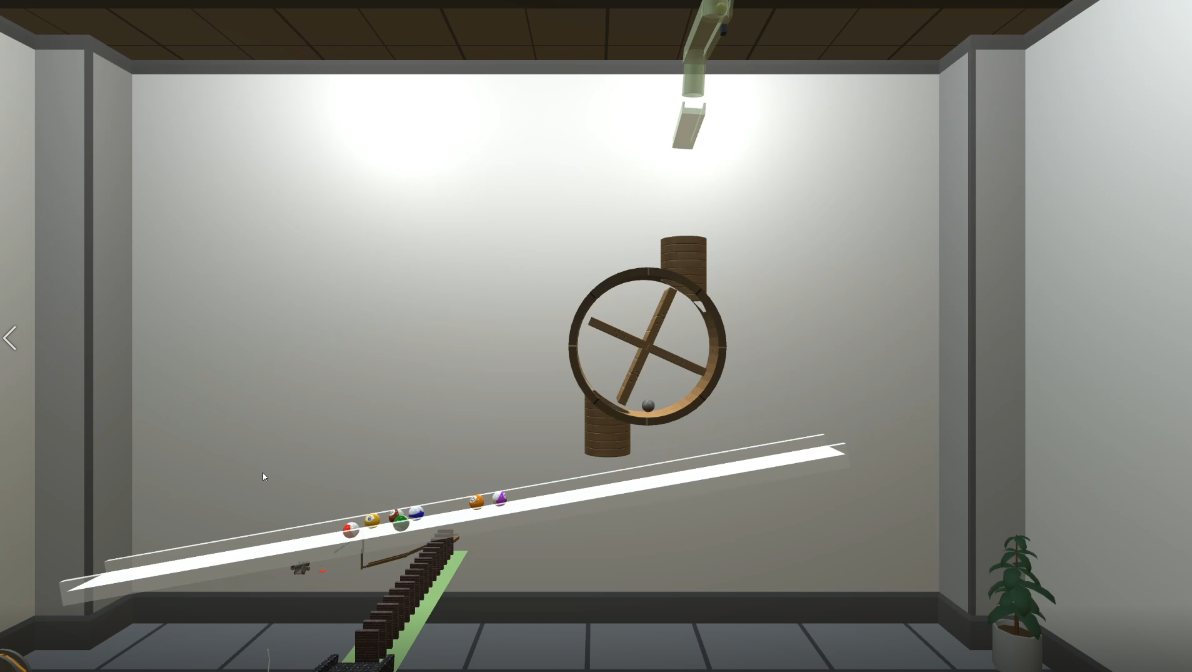
1. **How to Run**

진수

1. **Example**
   1. **Part 1**
   2. **Part 2**



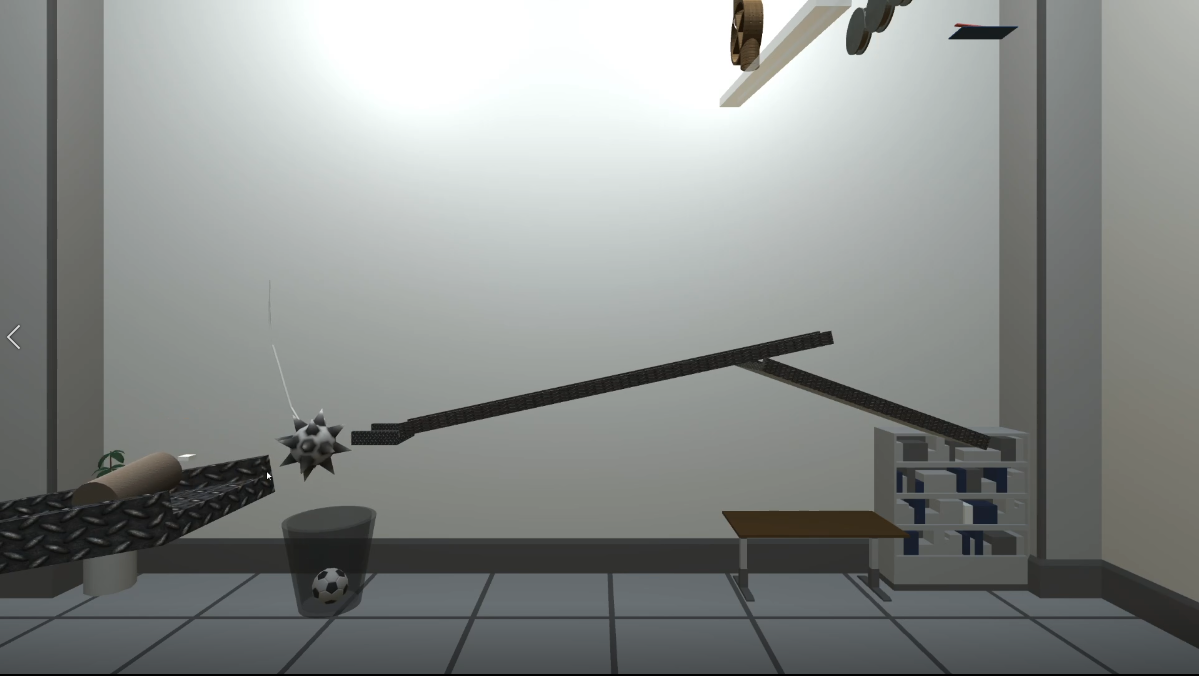
Part 1과 Part 2의 연결부분이다. Part 1에서 Part 2로 좀 더 자연스럽게 연결되게 하기 위해 작은 구조물을 하나 추가하였다. Part 1의 마지막 시소로부터 공이 날아와 컵에 들어가면, 반대쪽에 있던 큰 쇠공이 굴러가면서 Part 2로 이어진다.



Function 1의 작동 모습이다. Part 1과 2의 연결부에서 공이 떨어지며 물레를 작동시키고, 물레 안에 원래 들어있던 7가지 색의 공들이 떨어지면서 굴러간다.



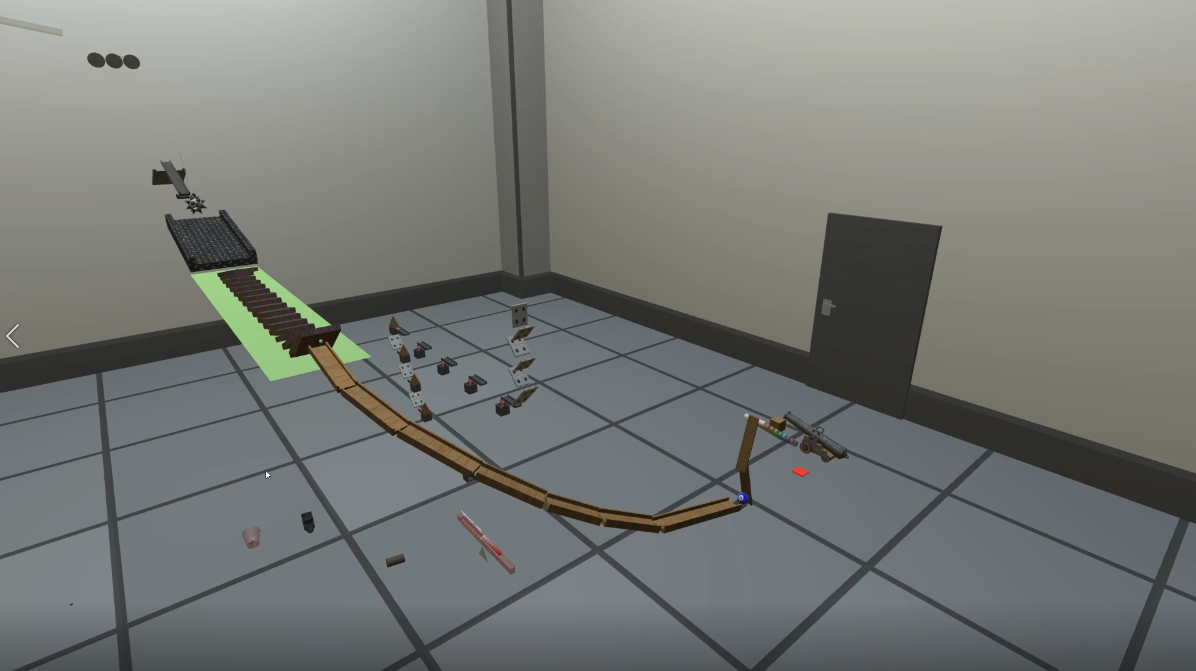
Function 2의 작동 모습이다. Function 1의 공들이 양동이로 모두 떨어지면, 모루와 연결된 케이블이 끊어지면서 양동이와 케이블로 연결된 모루가 떨어진다.



Function 3의 작동 모습이다. 모루가 떨어지면서 판이 기울게 되고, 그로 인해 축구공이 굴러가게 된다. 축구공이 구르다가 철퇴와 부딪히면, 줄에 매달린 철퇴가 떨어지면서 진자 운동을 통해 반대편의 원기둥을 쳐 구르게 한다.



Function 4의 작동 모습이다. 원기둥이 굴러가면서 도미노를 넘어뜨리고, 짧은 도미노 끝에 위치한 파란색 당구공이 마지막 도미노가 넘어짐으로 인해 굴러가게 된다.



Function 5의 작동 모습이다. 파란색 당구공이 굴러가다 나무 판을 건드리면, 나무 판이 회전하면서 나무 큐브를 밀게 되고, 원기둥으로 이루어진 길을 굴러가게 된다.



Function 6의 작동 모습이다. Function 5에 의해 굴러가는 나무 큐브가 떨어지며 버튼에 닿게 되면 버튼이 파란색으로 변하고, 잠시 후 대포에서 포탄이 날아가며 Part 3으로 이어진다.

* 1. **Part 3**

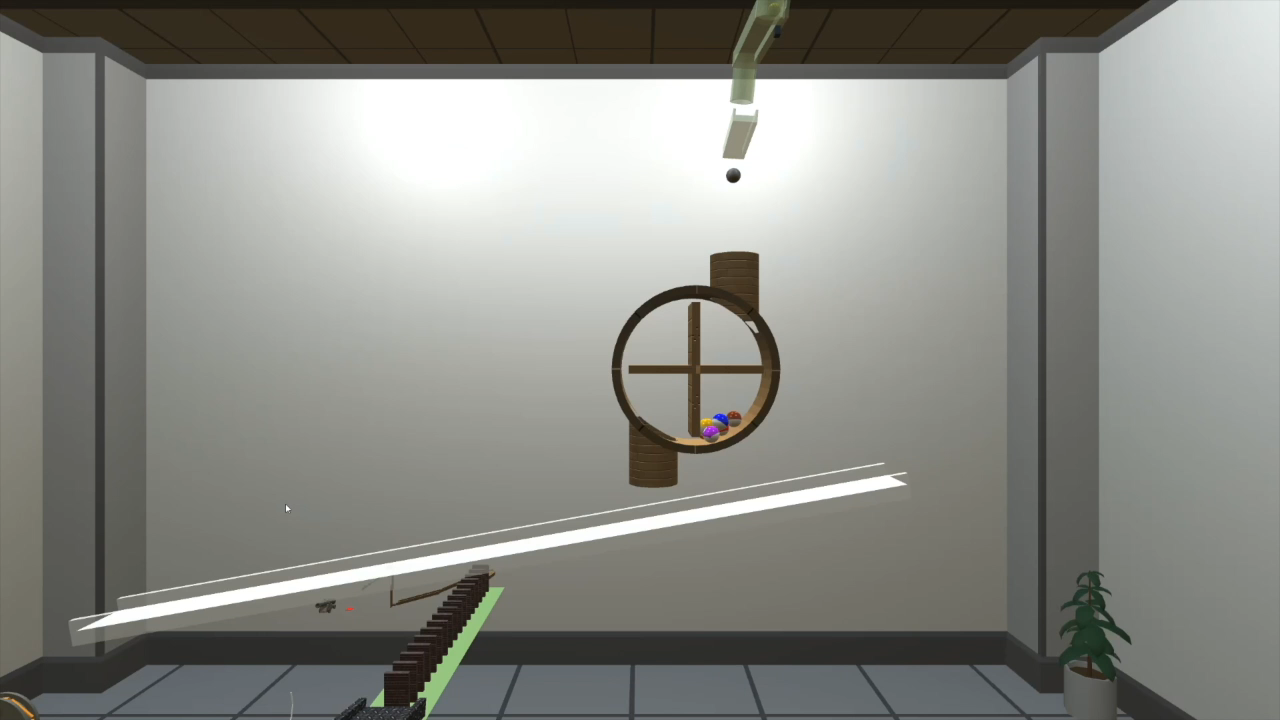
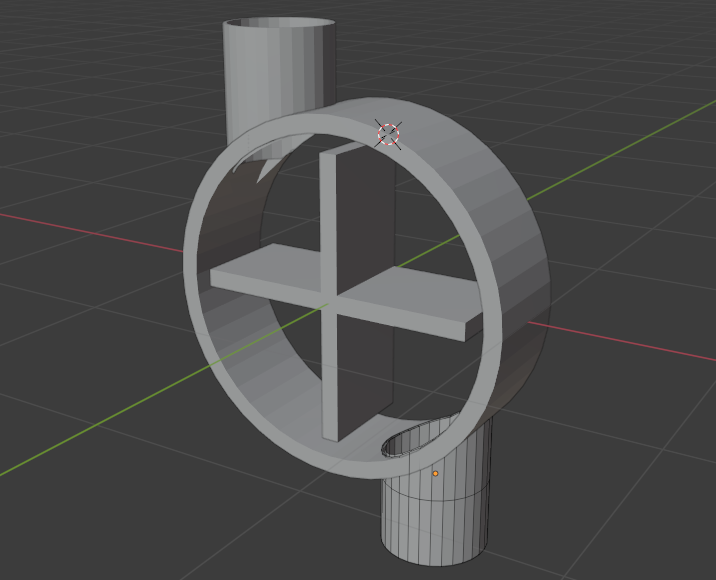
1. **DISCUSSION**
   1. **Part 1**
   2. **Part 2**
      1. 복잡한 오브젝트(Function 1의 물레 등) 구현

Part 2의 구현에서 대부분의 기능들은 간단한 모양의 오브젝트의 적절한 배치 및 rigidbody, collider 등 간단한 물체 특성 조절로 구현되었지만, 그렇지 않고 Part 2의 Function 1의 물레와 같이 복잡한 모양을 가진 오브젝트도 있었다. 물레의 경우, Unity에서 제공하는 기본적인 도형만으로 구현할 수 없었고, 유니티 애셋 스토어에도 물레 기능을 하는 일부 애셋들이 있었지만, 우리의 골드버그 머신 설계에 부합하는 모양을 가진 물레는 찾을 수 없었다.

이 문제 해결하기 위해서는 유니티의 기본 제공 도형만으로 구현할 수 있게 하거나 외부 애셋을 끌어와 사용할 수 있게끔 디자인을 변경하거나 우리의 목적에 부합한 물레를 직접 구현해야 했는데, 디자인을 변경하는 것보다는 디자인에 맞는 물레를 구현하는 것이 이번 그래픽스 프로젝트의 목적에 더 부합하다고 생각하여 Blender를 이용하여 물레를 직접 구현하였다.

이번 골드버그 머신의 구현에 필요한 물레는 가로로 놓인 속이 빈 원기둥 위아래에 일부 잘린 원기둥이 붙어있고, 물레 안쪽에는 회전하는 날개가 달려 있는 모습을 하고 있다. 물레의 입구와 출구 부분에 구멍을 뚫고 입구와 출구에 해당하는 원기둥을 붙이는 작업은 Blender의 knife 기능을 이용하여 구현했고, 물레 안쪽에 Box 4개를 붙여서 만든 날개를 배치하여 물레 모양을 완성시켰다.

Blender로 물레 모양을 완성시킨 후에는, unity로 이 물레를 가져와서 우리가 원하던 물레의 물리적 특성을 가지게 만들어야 했다. 첫째로, 물레 자체가 외부의 충격이 있어도 제자리를 유지하며 회전되지 않아야 하고, 둘째로, 물레의 날개 또한 공이 떨어지더라도 위치가 변경되지 않고, 날개의 회전 또한 우리가 원하는 축으로만 회전해야 했다. 마지막으로, 물레의 비어있는(투명한) 옆면을 통해 공이 흘러내리지 않아야 했다. 이러한 물레의 물리적 요구사항을 충족시기키 위해, 첫째로 물레의 겉부분에는 rigidbody를 적용시키지 않고 mesh collider만 적용시켜 외부의 충격에 영향이 없도록 설정했다. 둘째로, 물레의 날개 부분에는 rigidbody를 설정하는 대신 중력의 영향을 받지 않게 하고, 날개의 이동과 회전축에 constrinat를 추가하여 우리가 원하는 방향으로만 회전할 수 있게 설정했다. 마지막으로, 물레의 비어있는 옆면에 투명하고 얇은 box collider를 추가하여 물레 안에 들어있는 공들이 물레 옆면을 통해 새어나올 수 없도록 만들었다.



좌: Blender에서 구현한 모습. 우: 실제 골드버그 머신에 사용된 물레

물레 외에도, 양동이나 속이 빈 원기둥처럼 유니티의 기본 제공 도형으로 구현하기 곤란한 일부 다른 오브젝트들도 Blender를 이용하여 구현하였다.

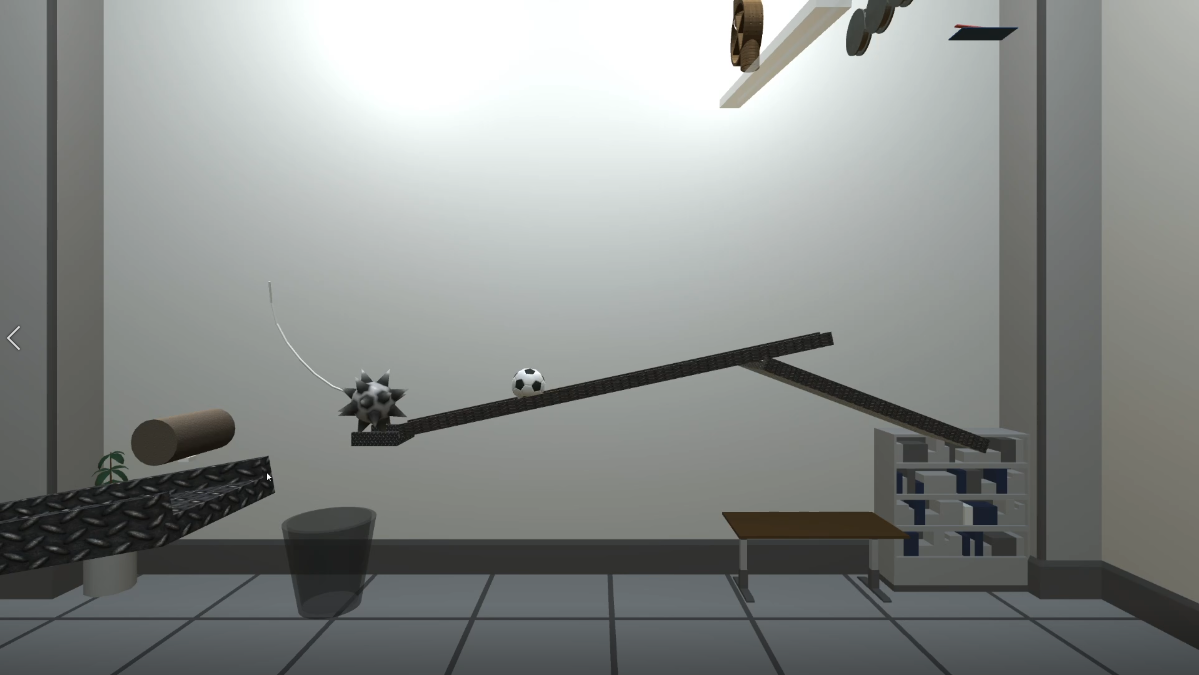
* + 1. 도르래 및 진자 기능을 위한 케이블 구현

Part 2의 구현에서 어려움을 겪었던 다른 부분은 Function 2, 3의 도르래 기능과 진자 기능 구현에 사용되는 케이블을 만드는 것이었다. 이 케이블의 경우, 도르래의 작동을 위해 양쪽 끝에 물체가 매달려 있으면 양 끝에 달린 물체의 무게에 따라 어느 한 쪽으로 쏠려야 한다. 그리고, 진자의 작동을 위해 케이블에 매달린 물체가 낙하하면 진자 운동을 해야 했다.

이러한 기능을 지원하는 케이블을 구현하기 위한 방법은 두 가지가 있었다. 첫째는 애셋 스토어에서 적절한 애셋을 찾아 사용하는 것이고, 두 번째 방법은 직접 구현하는 것이었다. 애셋 스토어에서 우리가 원하는 기능에 부합해 보이는 케이블 애셋을 찾을 수는 있었지만, 가격이 상당히 비쌌고, 애셋을 사용하기보다는 직접 구현하는 것이 그래픽스 프로젝트의 목적에 좀 더 부합하다고 생각했기에 직접 구현하는 것을 선택하였다.

유니티에서 밧줄을 직접 구현하기 위해 활용할 수 있는 적절한 기능을 탐색했고, 그 결과 유니티의 Hinge joint라는 기능을 사용해서 케이블을 구현하였다. 이 hinge joint를 이용한 케이블은 진자 기능에는 유효하게 작동했으나, 도르래 기능에는 올바르게 작동하지 못했다. 어느 한 쪽에 달린 물체가 더 무겁더라도 그쪽이 끌려 내려가지는 않았다.

이 문제를 해결할 방법을 찾기 어려웠고, 결국 도르래 기능을 구현하는 것은 어렵다고 판단하여 어쩔 수 없이 function 2의 디자인을 일부 수정하였다. 케이블 끝에 매달린 양동이에 공이 가득 차면 반대편에 달려 있던 모루가 올라오면서 공을 건드려 다음 function으로 넘어가게 하는 대신, 양동이에 공이 가득 차면 케이블이 끊어지면서 반대편에 매달린 모루가 떨어져 다음 기능으로 넘어가게끔 디자인을 일부 수정하였다. 이 기능은 Hinge joint에 일정 이상의 힘을 주면 joint가 끊어지게 하는 Break Force의 값을 조절하고, 스크립트를 사용하여 양동이에 공이 가득 차면 그 이상의 힘을 가해주게 하여 구현되었다.

좌: function 2에 사용된 케이블의 모습. 우: function 3에 사용된 케이블의 모습

* 1. **Part 3**

1. **CONCLUSION**
   1. **Part 1**
   2. **Part 2**

이번 프로젝트 Part 2를 구현하는 과정에서 복잡하게 생긴 오브젝트를 Blender를 사용하여 직접 만들어보면서 복잡한 오브젝트의 구현 방식에 대해서 알게 되었다. 복잡한 오브젝트의 구현은 유니티 내에서 이루어지기보다는 Blender, 3DMax, Maya와 같은 별도의 3D 모델링 프로그램을 사용하여 복잡한 오브젝트를 모델링한 후, 이러한 모델을 Unity로 임포트하여 복잡하게 생긴 오브젝트를 구현함을 알게 되었다. 또한, 이러한 복잡한 오브젝트의 물리적 특성의 경우에는 3D 모델링 툴에서 물리적 특성 설정을 지원하는 경우도 있고, Unity에서 임포트한 후 Unity 내에서 rigidbody 및 collider 조절을 통해 구현이 가능하다는 점 또한 알게 되었다.

* 1. **Part 3**

1. **DIRECTION OF IMPROVEMENT**

**References**

Part 2의 function 6에서 사용된 대포 및 포탄 Asset - <https://assetstore.unity.com/packages/3d/props/cannon-on-a-platform-57534>

Part 2의 function 2에서 사용된 모루 Asset - <https://assetstore.unity.com/packages/3d/props/weapons/cartoon-heavy-weights-2857>

배경 장식에 사용된 오피스 테마 Asset - <https://assetstore.unity.com/packages/3d/environments/snaps-prototype-office-137490>