자체 물리 엔진을 이용한 유니티 컬링 게임

이용현, 박기범
경희대학교
3246lyh@naver.com, sesbgb@naver.com

Unity curling game based on physics

Yong Hyun Lee, Ki Beom Park
Kyunghee University

요 약

2018년 평창 동계올림픽부터 우리나라 사람들이 컬링이라는 종목에 큰 관심을 갖기 시작하였다. 하지만 이 종목을 직접 체험하거나 경기를 보기 위해서는 빙판이 있어야 하는 특수성과 비싼 장비, 찾아보기 힘든 경기장 등 여러 열악한 조건들 때문에 결국 올림픽 시즌에만 반짝 관심을 가졌다가 시들어버렸다. 이를 해결하기 위해서 우리는 Unity라는 게임 엔진을 사용하여 사람들이 쉽게 접할 수 있는 컬링 게임을 제작하였다. 실제 컬링을 게임으로 만들기 위해 컬링에 필요한 도구들을 이미지로 제작하여 Unity 내부에서 저장 후 오브젝트에 입력하였고 물리 법칙을 구현하기 위해 Unity 상에서 방향, 세기, 속도, 충돌들을 프로그래밍하였으며 대한컬링연맹에 나와있는 컬링 경기 규칙서를 활용하여 게임에 적용하였다. 또한 컬링의 진행이 현실적인 운동과 비슷하게 하기 위하여 스크립트 안의 충돌 및 마찰 관련 계수를 조절하였고 이를 이용하여 반복한 결과값들을 수치화하여 그래프로 작성해보았다. 추가적으로 컬링 게임의 점수판과 카메라 시점 등을 통해서 게임 사용자가 게임 진행에 있어서 도움이 되는 부분을 구현하였고 현실성을 위하여 Arduino를 이용한 게임 패드를 제작하여 직접 게임하는 듯한 느낌을 들도록 하였다. 최종적으로 게임을 이용하여 컬링에 대한 이해도가 증가하고 사람들이 컬링이라는 비인기 종목에 한 걸음 더 접근할 수 있게 되고, 스포츠발전에 조금이나마 기여할 수 있게 될 것이다.

1. Introduction

2018년 평창 동계올림픽에서부터 우리나라는 컬링이라는 새로운 종목에 관심을 가지기 시작하였다. 모든 선수가 김씨라 붙여진 “팀킴(Tram Kim)”이라는 별명과 여론조사 전문기관 리얼미터에서 조사한 2018년 올해의 말 1위로 기록된 경기장에 울려 퍼진 “영미~”라는 밈(Meme), 비인기종목에서의 태양처럼 떠오른 팀이 올림픽에서 얻은 값진 은메달은 그해 우리나라의 유행 거리가 되기 충분했다. 우리는 실제로 겪지 못하는 일들을 게임에서 간접 경험해 볼 수 있다. 예를 들어 비가 오거나 하는 이유로 하고 싶은 축구를 하지 못할 때, 우리는 게임으로나마 그 욕구를 어느 정도 해소할 수 있다. 하지만 컬링은 올림픽 당시 그러한 열기에도 불구하고, 체험해볼 수 있는 게임조차 쉽게 발견하지 못했다.

추가적으로 컬링이라는 종목은 빙판이 있어야 하는 동계스포츠라는 특수성과 비싼 장비, 찾아보기 힘든 경기장 등 여러 열악한 조건들 때문에 결국 올림픽 시즌에만 반짝 관심을 가졌다가 시들어버렸다. 특히 컬링 시트의 얼음 표면의 온도는 영하 5°의 온도를 유지해야 하며 습도도 계속 모니터링을 하며 조정을 해야한다. 이 때문에 컬링 경지장을 만들고 보수 및 유지를 하기 위해서는 큰 돈과 자원이 필요하는데 개발도상국이나 더운 국가의 경우 컬링 시트를

유지하기에는 큰 어려움이 따른다.

이러한 이유로 인해 더운 국가나 개발도상국은 컬링이라는 종목을 접하기 쉽지 않는데 메타버스를 통해 Unity를 이용한 컬링 게임을 만든다면 일반인들이 컬링이라는 비인기 종목에 한 걸음 더 접근할 수 있게 되고, 스포츠 발전에 조금이나마 기여할 수 있게 될 것이다. 또한 컬링을 접하기 어려운 국가들도 게임을 통해서 쉽게 접할 수 있으며 경기장이나 장비 같은 어려운 조건들 없이도 게임을 통해서 연습할 수 있다.

이를 이용하여 더욱 개발하면 VR 방식을 사용하여 경기에 대한 감각도 연습할 수 있다고 본다.

2. Materials and Method

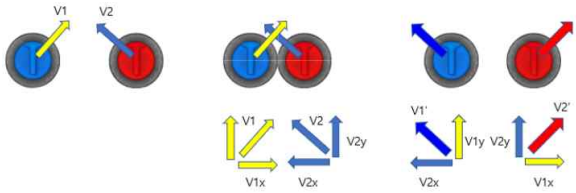
이 연구는 유니티 내에서 물리 엔진을 구현하는 것에 중점을 두고 진행을 하였다. 컬링을 진행하기 위해서는 컬링에 적용되는 회전, 마찰, 물리적 충돌에 관한 법칙들을 확인해야 하고 각 법칙에 적용되는 수치들에 대해서 사용자가 어색한 부분이 없고 실제 컬링과 비슷한 경험을 하도록 조절하여야 한다.

1. 회전 및 마찰력

모든 물체들은 이동하면서 관성을 갖는다. 컬링에서의 스톤도 이 관성에 의해 시트 위에서 더 멀리 나아갈 수 있지만 관성에 의해 방향 전환이 힘들어 지기도 한다. 하지만 컬링 시트 위의 패블이라는 얼음 알갱이들이 뿌려져 있어서 시트는 우둘투둘한 빙판이 되는데 이 때문에 컬링 선수의 스위핑에 따라 마찰력의 세기와 스톤의 회전을 조절할 수 있다. 즉, 스톤의 진행 방향에 회전을 주고자 하는 지점까지는 스위핑을 통해 낮은 마찰을 유지해 스톤의 속도를 유지시키고 그 지점부터는 스위핑을 조절해가며 스톤의 속도를 낮춘다. 또한 스톤의 진행방향의 왼쪽이나 오른쪽을 선택적으로 스위핑을 조절하여 스톤의 회전력에 추가적인 변화를 줄 수도 있다. 이를 유니티 내에서는 Object의 Rigidbody 2D와 Friction Joint 2D 두가지 컴포넌트를 사용하여 구현하였다. Friction Joint 2D를 사용하여 자연스러운 기본 마찰정도를 고정시켰고, Rigidbody 2D의 LinearDrag(선형마찰, 공기저항)의 값을 조절하여 현실과 비슷한 느낌이 나도록 조절하였다.

2. 충돌

완전 탄성일 경우 스톤의 충돌 직전에 각 스톤들의 속도를 접하는 부분의 벡터와 같은 방향과 수직 방향으로 나누고 충돌 후와 비교한 결과 접하는 벡터와 수직한 방향의 벡터들만 교환되는 것을 알 수 있다.



수식으로 설명하면 위의 그림처럼 V1과 V2를 접하는 부분을 기준으로 수평, 수직 방향으로 나눌 수 있는데 이를 식으로 표현하면 충돌 전 속도 벡터는

$$V_1 = V_{1X} + V_{1Y},\ V_2 = V_{2X} + V_{2Y}$$

와 같다.

충돌 후 속도 벡터는

$$V_1' = V_{2X} + V_{1Y},\ V_2' = V_{1X} + V_{2Y}$$

로 표현할 수 있다. 이를 사용하여 충돌을 구현하였다.

위 이론을 직접 적용을 한 물리충돌 구현을 위해 내장된 물리엔진을 사용하지 않고 최대한 근접하게 Unity의 충돌 함수만을 사용하여 직접 구현하였다.

Unity의 충돌 함수에는 Collision과 Trigger 두가지 방식이 존재하는데, 서로의 단점을 보완하기위해 두가지 방법 모두를 사용하였다. 먼저 Trigger방식에선 할 수 없는 Continuous 적인 충돌감지를 위해 Stone의 크기와 동일한 Collider를 사용하였다. OnCollisionEnter2D함수에서는 “Collider와 충돌이 일어나는순간”을 뜻하지만 이미 물리적 연산이 끝나고나서 실행되는 함수이다. 따라서 기존 충돌 물리엔진이 작동되지 않게 Stone의 크기보다 조금 더 큰 Trigger Collider를 사용하였다.

따라서 충돌과정은 일반적으로 OnTriggerEnter2D - OnCollisionEnter2D - OnCollisionExit2D - OnTriggerExit2D의 순서로 일어난다.

뒤에 작성될 문제로 인한 추가구현은 후에 서술하고 기본적인 원리를 순서대로 설명하자면,

(1) CollisionEnter시 강제로 일어나는 물리엔진 계산을 개선하기 위해 TriggerEnter일때의 Velocity를 저장한다. Trigger와 Collider 크기차이로 인해 사이에서 일어나는 자그마한 마찰감속은 무시하며 이후 적용할 탄성계수조절로 구현하였다.

(2) TriggerEnter시에는 위에 기술된 물리법칙을 계산하여 저장한다. 충돌시에 발생하는 법선을 기준으로 수직인 방향은 X축, 평행한 방향은 Y축으로 가정하며 함수 특성상 모든 충돌함수는 양쪽 Stone에서 동시에 일어나는데 먼저 실행되는 함수에서 상대 Stone의 Velocity까지 모두 계산하여 종료한 후 상대 Stone의 함수가 실행될 때는 계산하지 아니하게 한다. 충돌 직후 먼저 실행된 함수의 Stone 기준으로 크기가 1인 UnitX(X축 단위벡터), UnitY(Y축 단위벡터)를 기저벡터로 양쪽 Stone 모두에 저장한다. 각각의 모든 현재 Velocity(기존

Rigidbody2D 물리엔진으로 계산된 Velocity)는 무시하고, TriggerEnter시에 저장된 Velocity와 기저벡터들을 내적하여 재정의된 좌표에서의 벡터들을 계산한다(각각 NewX, NewY). 이때 CollisionExit시 계산된 NewX와 NewY로 Velocity를 설정해야 하는데 기존 물리엔진에 의해 Stone이 탄성있게 튕기지 않고 찰흙처럼 붙어서 진행되는 현상이 일어나 매우 작은거리를 벌어지는 방향으로 이동(이때 반드시 transform이 아닌 Rigidbody2D의 Position변수를 조정해야한다)하여 강제로 OnCollisionExit2D를 실행하도록 하였다.

(3) CollisionExit에서는 CollisionEnter때 계산된 NewX와 NewY를 Velocity에 할당해주어 원래의 물리법칙대로 진행하도록 해주었다.
위의 3단계를 기본으로 구현하였으나 의도치 않게 Collision과 Trigger가 각각 서로 충돌하였을때 함수가 발생하여 Stone내의 Empty Object를 Child로 두어 Tag를 구분하여 Collision과 Trigger Collider가 서로 인식하지 않게 하였다. TriggerEnter - CollisionEnter가 일어나는 사이에 또다른 Stone의 Trigger Collider가 TriggerEnter되는 현상이 발생하는 특정한 경우를 처리하기위해, 위의 방법에서 값이 아닌 List를 두어 처리하도록 하였다. 1단계에서는 List.Add를 사용, 값 계산시에는 List에 저장된 이름들중 일치하는 이름 발견시 값계산, 충돌 발생시 바뀐 Velocity를 OnTriggerStay2D에서 처리하고 OnTriggerExit2D에서 List.Remove을 사용 하였다.

3. Results and Discussion

컬링 경기에서 사용되는 물리적 법칙을 게임 진행 순서에 맞추어 구성하였다. 또한 게임적인 정서를 넣기 위하여 여러 게임적 요소를 추가하였다.

1. 컬링의 방향 설정

Arduino로 제작된 조이스틱을 사용하거나 키보드의 왼쪽, 오른쪽 방향 키를 사용하여 스톤의 방향을 설정할 수 있는데 원하는 만큼의 방향으로 조절 후 버튼으로 결정한다.

2. 컬링의 속도

0부터 호그라인 까지 스톤이 움직일 수 있는 최소 세기까지 추가적으로 표시하고 세기가 90이 넘어가게 되면 PowerShot으로 설정하였다. Arduino로 제작된 게임패드를 사용하는 경우 A로 설정된 버튼을 누르면 버튼을 누르는 동안 세기를 조절하고 원하는 만큼의 세기의 순간에 다시 버튼을 누르면 그 만큼의 스톤의 속도를 얻을 수 있다.

이를 시각적으로 표현하기 위하여 게임 화면의 오른쪽 중간 위치에 게이지 바를 만들어 확인할 수 있다.

3. 스톤의 회전

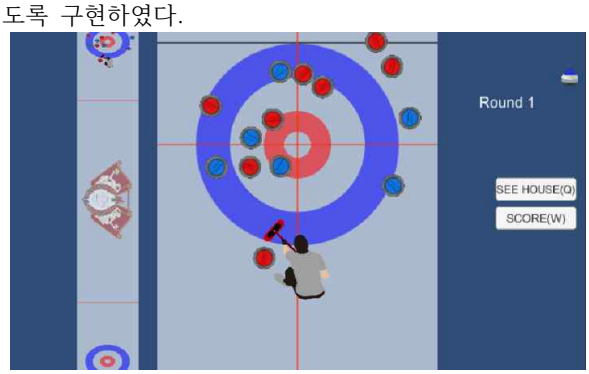
속도와 마찬가지로 A로 설정된 버튼을 누르면 화면 중간 우측 부분에 빨간 원이 생기며 시계방향 또는 반시계방향으로 원이 채워지게 되는데 시계방향의 경우 오른쪽 회전 반시계방향의 경우 왼쪽 회전으로 움직이도록 설정하였고 원하는 만큼의 회전량이 결정되면 그 순간에 버튼을 떼면 적용되게 된다.

4. 스위핑

Arduino로 제작된 게임패드의 조이스틱을 왼쪽 또는 오른쪽으로 움직이거나 키보드의 왼쪽, 오른쪽 방향키를 누르게 되면 스위핑을 1회 실시한 것으로 설정하여 스톤과 컬링시트 사이의 마찰을 줄이고 회전을 늘려 더 멀리 나아가도록 구현하였다.

5. 스톤과 스톤 사이의 충돌

스위핑을 하고 나서 스톤이 앞으로 전진하게 되면 전에 던진 스톤들이 하우스 근처에 있게 되는데 이를 던진 스톤이 충돌하는 것을 구현하였다. 이때 마찰, 분리각, 회전을 적용하여 실제 컬링 스톤들이 충돌하는 것 같은 효과가 보이도록 구현하였다.



충돌 전

충돌 후

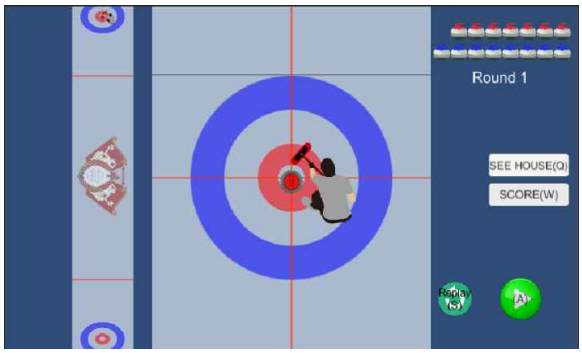
앞서 설명과 같이 완전 탄성일 경우로 가정하여 구현하였고 반발계수를 정하기 위하여 Redstone과 Bluestone을 각각 하나씩 배치한 후에 충돌시켜 Bluestone의 속도 벡

터의 스칼라 값을 비교한 결과 아래의 그래프와 같이 나왔으며 이를 통해 반발계수의 값이 0.75인 경우 가장 현실성이 있다고 판단하였다.

반발계수		
0.9	0	49.72702932
0.85	0	46.96441533
0.8	0	44.20175119
0.75	0	41.43913719
0.7	0	38.67660972
0.65	0	35.91399572
0.6	0	33.15134161
0.55	0	30.38872988
0.5	0	27.62611815
0.45	0	24.86350641
0.4	0	22.10089468
0.35	0	19.33828294
0.3	0	16.57567121
0.25	0	13.81305948

6. 마찰

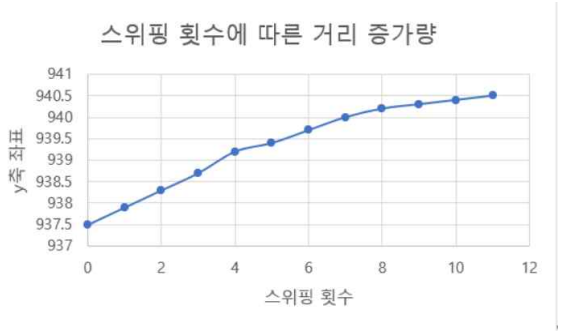
마찰의 경우 코드 내에서 스위핑을 하는 경우와 하지 않는 경우로 나누어서 마찰을 조절하였는데
토크와 힘을 고정시킨 후 스톤의 위치를 비교하였을 때 아래의 그림과 같이 스위핑을 한 경우 더 멀리 나아가는 것을 확인할 수 있었다.



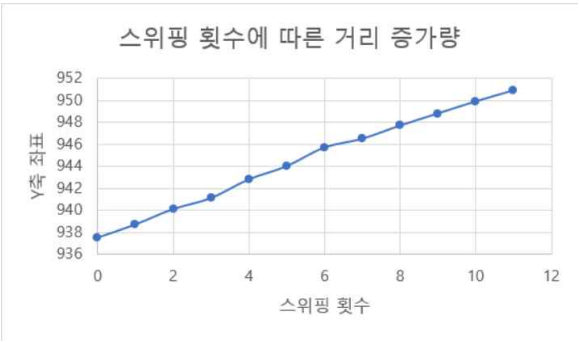
스위핑을 하지 않는 경우

스위핑을 한 경우

또한 스위핑에 관한 실험을 하면서 스위핑 횟수에 따라 거리가 조금 멀어질 수도 있고 많이 멀어질 수도 있다는 것을 확인하였다. 이를 정확히 확인하기 위하여 코드 내에서의 스위핑 관련 계수를 0.02, 0.03의 경우로 나누고 스위핑 횟수를 0에서 11회까지 측정하여 그래프로 확인한 결과 아래와 같았다.



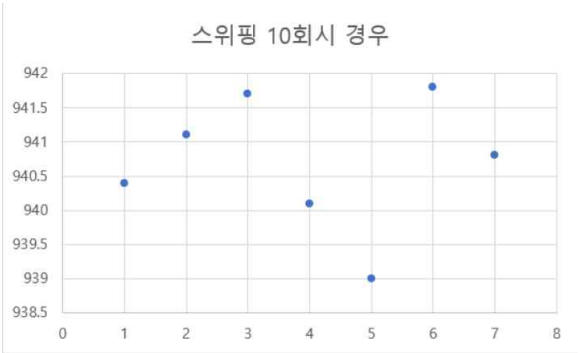
계수가 0.02인 경우



계수가 0.03인 경우

이를 통해 스위핑 횟수가 증가할수록 더 멀리 나아가는 것을 확인할 수 있었다.

추가적으로 스위핑을 하면서 같은 횟수의 스위핑을 하는 경우 매번 동일한 거리만큼 나아가는 것이 아니고 매번 달라지는 것을 확인하였는데 이를 판단하기 위하여 스위핑 횟수를 10회로 고정하고 측정한 결과 아래와 같았다.



스톤이 진행하면서 스위핑을 하게 되는데 스톤이 출발하자마자 바로 스위핑을 하는 경우와 어느 정도 스톤이 앞으로 진행한 후에 스위핑을 하는 경우를 비교해 보니 출발 직후에 스위핑을 하는 것이 스톤이 더 멀리 나아가는 것을 확인할 수 있었다. 물론 섬세한 컨트롤을 위해서는 앞서 진행한 스톤들의 위치와 예상 경로를 예측하여 본인이 감각적으로 적절한 위치에서 스위핑을 하는 것이 좋다고 본다. 컬링 경기의 경우 매번 경우의 수가 달라지기 때문에 명확한 해답은 없다고 생각한다.

추가적인 게임 요소

1. 카메라 시점 추가

스톤의 전체적인 진행 경로를 확인하면서 스위핑 여부와 스위핑 횟수를 결정하기 위하여 시점을 추가하였다. 기존에 진행하는 스톤을 따라가는 메인 Camera 시점을 제외하고 전체적인 시점을 추가하여 해당 시점을 작게 비출 수 있는 UI를 추가하였다. 게임 화면에서 왼쪽부분에 위치한다.

2. 점수 계산 및 점수판 구현

스톤이 라운드가 끝날 때마다 점수를 컬링 경기 규칙서에 기반하여 계산을 하고 점수판에 점수가 나타나도록 구현하였다. 만약 마지막 라운드까지 동점이 나오는 경우 라운드 하나를 추가하여 마지막 경기를 진행할 수 있도록 하였다.

3. 하우스 버튼 및 하우스 시점 구현

4. 리플레이 버튼 및 구현

리플레이 버튼 실행 시 바로 이전에 스톤이 진행하는 영상을 실행시킨다. 스톤의 방향을 선택하는 것부터 스톤이 충돌 후 정지하는 것까지 보여준다.

5. 게임 이미지 제작

게임 시작 및 정보 화면을 제작하여 게임을 하는 듯한 느낌을 만들어 보았다.

또한 시작 버튼을 누르는 경우 게임 라운드 수를 선택하거나 빨간 스톤 또는 파란 스톤의 선공을 정하기 위한 화면도 제작하였다.

게임을 처음 접하는 사람들을 위해 게임에 대한 설명이 담겨있는 게임 방법을 만들었다. 이때 GIF로 이미지를 넣어 영상과 같은 효과를 볼 수 있고 이전 버튼(Q), 다음 버튼(W)를 설정하여 이전 및 다음 화면을 볼 수 있도록 하였고 처음화면(E) 버튼을 구현하여 게임 초기 화면으로 돌아갈 수 있도록 하였다.

게임의 승패에 관한 이미지도 제작하였다. RedTeam이 이긴 경우, BlueTeam이 이긴 경우, 무승부의 경우 3가지로 제작하여 구현하였다.

6. Arduino 게임패드 제작

키보드로만 버튼을 누르면 현실성이 떨어진다고 생각하여 직접 게임패드를 제작하였다. 왼쪽엔 조이스틱 오른쪽엔 버튼 6개를 두었고 왼쪽 위에서부터 순서대로 q,w,e,a,s,d로 설정하였다.

4. Conclusion

이번 연구는 Unity 게임 엔진과 Arduino 회로를 사용하여 사람들이 쉽게 컬링이라는 비인기 종목에 한걸음 더 접근할 수 있으며 스포츠 발전에 기여할 수 있고 직접 만든 컬링 게임을 사용하는 사람들이 Unity라는 게임 엔진을 통해 현실적인 부분을 느낄 수 있고 Arduino를 사용한 게임 패드를 통해 직접 게임하는 듯한 느낌을 갖도록 하였다. 이를 위해 컬링과 관련 규칙서를 찾아보며 컬링 규칙과 사용된 장비들에 대해 공부하였고 Unity 상에서 직접 이미지를 제작하고 구현하여 위와 같은 게임을 만들었다. 또한 Arduino Leonardo 보드를 사용하여 게임 패드를 제작하였다.

게임을 만들면서 스톤에 영향을 주는 방향 설정, 힘, 회전, 마찰력, 충돌 등을 구현하였으며 방향의 경우에는 조이스틱 또는 키보드 방향키를 이용하여 조절할 수 있도록 하였고 힘(게이지)를 주는 경우에는 게임패드(A 버튼)을 누르고 떼면서 원하는 정도의 힘을 줄 수 있도록 하였다. 회전의 경우에는 힘을 설정 후에 진행하게 되는데 힘과 마찬가지로 게임패드(A 버튼)을 누르는 동안 원하는 만큼의 토크를 줄 수 있도록 하였다. 마찰력의 경우 기본적으로 코드 상에서 여러 번 조절하여 눈으로 보았을 때 가장 알맞은 경우의 수치를 넣어서 만들었으며 컬링 경기의 경우 스위핑이라는 동작을 진행하게 되는데 이를 통해 마찰력과 회전에 영향을 줄 수 있다.

스위핑을 함으로써 스톤이 진행하는 동안 마찰력의 줄이는 효과를 가져오는데 이를 스위핑 횟수를 비교하여 측정하였더니 스위핑 횟수가 증가할수록 마찰력이 줄어들어 스톤이 더 멀리 나아가는 것을 확인할 수 있었다. 또한 스톤이 출발한 후에 스위핑을 언제 진행하는지에 따라 차이가 발생했는데 이는 출발 직후 바로 진행하는 경우가 나중에 진행하는 경우보다 스톤이 더 멀리 나가는 것을 확인할 수 있었다. 하지만 스톤의 섬세한 컨트롤을 위해서는 호그라인 직전에 스위핑을 하여 조절하는 것이 더 좋다는 판단을 하게 되었다.

충돌의 경우 실제 컬링과 비슷하게 하기 위하여 비탄성 충돌인 경우로 가정하고 구현을 하였지만 물리적인 부분을 코딩안에서 구현하기 쉽지 않아서 완전 탄성인 경우로 수정하여 구현하였다.

게임적인 부분을 추가하기 위하여 게임 시작 이미지와 끝 이미지를 추가하였고 게임을 처음 접하는 사람들을 위하여 게임방법을 추가하여 게임에 대한 이해도를 높일 수 있었다. 또한 게임의 라운드 수를 고르거나 팀의 선공을 정하기 위한 동전 돌리기를 넣어서 게임을 하는 듯한 느낌을 받도록 하였다.

현재 아두이노 보드의 크기 때문에 컨트롤러의 크기를 일정 수준 아래로 줄이는 것이 불가능 했기 때문에

초기 단계에서 기획했던 자이로 센서가 들어간 소형 컨트롤러 제작을 고려하였지만 다른 방법을 사용하였다. 또한 메타버스라는 주제를 더 살릴 수 있도록 VR 기기와 연동한 게임 플레이 제작을 시도했지만 3D 이미지 제작 및 프로그래밍 단계에서 한계점이 많아서 2D로 제한하여 제작하였다. 추후에 시간이 충분하다면 위의 문제들을 개선하고 싶고 네트워크를 사용하여 온라인 게임으로 개발하고 싶다.

이 연구를 통해서 사람들이 컬링에 대한 관심과 스포츠 발전에 조금이나마 기여하였으면 좋겠다고 생각하였다. 또한 빙판이 있어야 하는 특수성, 비싼 장비, 찾아보기 힘든 경기장과 같은 어려운 조건들 때문에 경험해보기 힘든 더운 국가들이나 개발도상국의 사람들이 이 연구를 통해 더 많은 관심과 게임을 통한 컬링에 대한 이해도를 높이는 경험을 하였으면 좋겠다.

5. References

[1] 대한컬링연맹, “컬링경기규칙서”, http://www.koreacurling.co.kr/rule.html , 5-6

[2] HaraldNyberg SaraAlfredson StureHogmark StaffanJacobson (2013), The asymmetrical friction mechanism that puts the curl in the curling stone, WEAR vol. 301 Issues 1-2, April–May 2013, Pages 583-589

[3] Norikazu Maeno, Dynamics and curl ratio of a curling stone, Sports Engineering 17,33-41(2014), 30 July 2013, Erratum 30 August 2013

[4]Dina Spector, Why curlers sweep the ice, 마지막 수정 Feb 14, 2018, 6:04 PM, 마지막 접속 2022/3/22 20:00, https://www.businessinsider.com/why-curlers-sweep-the-ice-2014-2

[5] scienceworld, curling rocks, 마지막 수정 Monday, February 22, 2010, 마지막 접속 2022/3/22 20:00, https://www.scienceworld.ca/stories/curling-rocks/

[6] real world physics problems, The Physics Of Curling, 마지막 접속 2022/3/22 20:00, https://www.real-world-physics-problems.com/physics-of-curling.html

[7] Jennifer Ouellette, Physicists on Ice: Exploring the Physics of Curling, 마지막 수정 September 21, 2014, 마지막 접속 2022/3/22 20:00, https://blogs.scientificamerican.com/cocktail-party-physics/physicists-on-ice-exploring-the-physics-of-curling/#

[8] 고재협, 방정원. (2017). 현실감 높은 게임 구현을 위한 2차원 공간상의 원형 물체의 탄성충돌 모델 연구. 한국컴퓨터정보학회 학술발표논문집, 25(1), 175-178.

[9] “과학을 알면 컬링의 전략이 보인다고?”, 사이언스레벨업, 2018년02월14일, https://sciencelevelup.kofac.re.kr/contents/guide/detail?pageIndex=1&class_type=guide&searchCol=&guide_idx=53&searchText=

[10] 육기승. (2002). 컬링. 스포츠과학, 79(0), 45-52.

[11] 권영일, 김태완, 최상협. (2021). 컬링 호그 투 호그 각 구간 스위핑 시 근 동원 양상 및 스톤의 정지 위치 분석. 체육과학연구, 32(1), 170-179.

[12] 이동일, 송오영. (2018). 컬링 스톤의 빙상 마찰력 분석을 통한 컬링 시뮬레이터 구현. 한국컴퓨터그래픽스학회 학술대회, () ,59-60.

[13] Turn your Arduino Pro Micro into a USB Keyboard - 아두이노 프로 마이크로 USB 키보드, https://s-engineer.tistory.com/202.

[14]“Arduino - Compare”, Arduino, 2022년04월15일 접속, https://www.arduino.cc/en/products.compare