[일반물리및시뮬레이션 과제 5]

소프트웨어학과 20011738 민재홍

[1. 배경]

피파온라인4 5/25일 9차 next field 패치에는 페널티박스 안의 다이렉트 슛(땅볼슛)과 중거리슛이 소폭 하향되었다(일명 너프). 하지만 소폭 하향이라 하더라도 “슛파워” 및 “슈팅자세에 따른 골결정력 능력치 하향” 등 공격수들이 골을 넣기 위해 필요한 부가적인 능력치들이 모두 하향되어 페널티박스 안에서 일반적인 루트로 넣었던 골들을 이제는 넣기 어렵게 되었다.

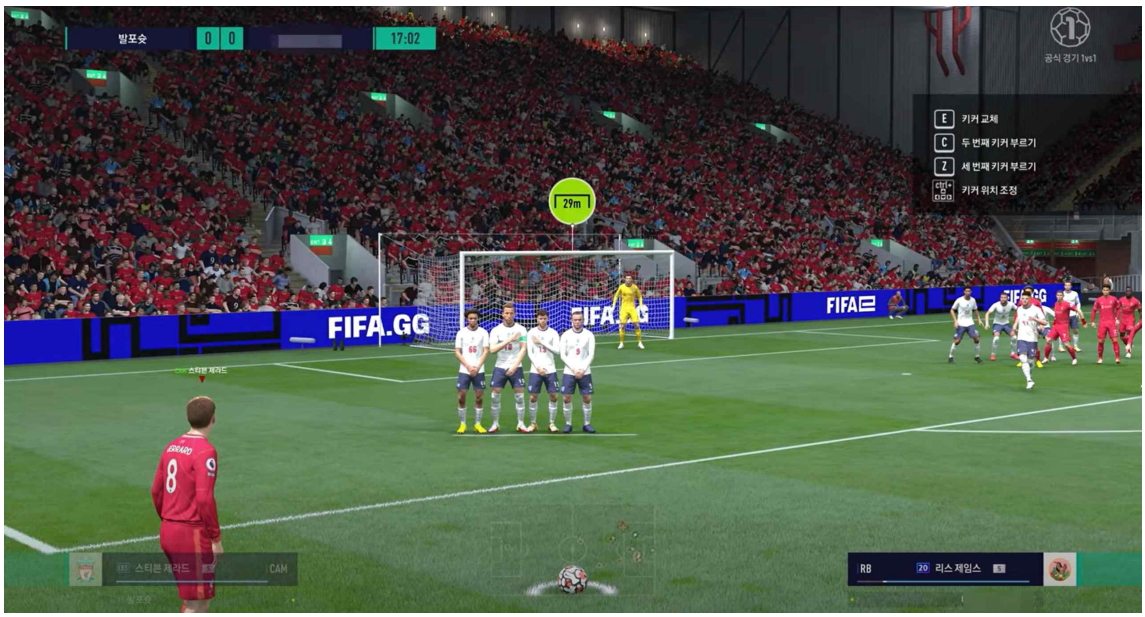
따라서 현재 가장 가성비 있는 득점루트는 세트피스이다. 그중에서도 코너킥은 게임 특성 상 수비수 AI가 공격수 AI(위치선정, 몸싸움 등)를 뛰어넘기에 성공확률이 매우 낮다. 따라서 프리킥으로 득점하는 방법을 익히고 연습하는 것이 가장 현명한 전략이다.

피파온라인을 연습하기 위해서는 피파온라인4에 접속을 해야 하는데, 상당히 높은 그래픽카드를 요구하는 게임인 만큼 노트북으로는 게임을 실행하기 어렵기에 집이나 PC방에 가지 않는 이상 프리킥 연습을 하기 어려운 현실이다.

따라서 이 기회에 VPython을 통해 피파온라인4 프리킥 연습을 할 수 있는 프로그램을 만들어보았다.

[2. 상황설명]

인게임 상 프리킥을 얻게 되면 다음과 같은 상황이 된다.



여기서 프리킥 차는 선수 아래에 있는 게이지바가 (거리 30m-40m 기준) 3칸-4칸 정도 차면 골대 상단 구석으로 빨려 들어가는 코스가 나온다. 이때 게이지바 1칸 채우는데 필요한 시간은 약 1초라고 보면 된다.

잘 들어가는 프리킥 종류로는 2가지가 있다.

1. 무회전 슛 (Q+D+위쪽방향키)

2. 감아차기 (Z+D+대각방향키)
   2-1) 위쪽 + 오른쪽 -> 오른쪽 감아차기
   2-2) 위쪽 + 왼쪽 -> 왼쪽 감아차기

괄호 안에 있는 키를 모두 눌러야 해당 프리킥이 나가게 된다.
이렇게 두 개의 프리킥을 연습하기 위해 두 가지 프리킥을 구현했다.

보통 프리킥은 잘 차기만 하면 (수비벽을 넘기고 골대 구석으로 차면) 골키퍼가 얼마나 좋은 선수이던 간에 들어가는 구조이다. 상대방이 사기 골키퍼를 사용하지 않는 이상 무조건 들어간다고 보면 된다. 하지만 완벽하게 구석으로 차는게 매우 어렵다. 인게임 내에서 게이지를 맞추는 것도 어렵고 거리랑 각도에 따라 무회전/감아차기 중 어떤 프리킥을 구사할 것인지 판단력과 감이 굉장히 중요하기 때문이다.

인게임 내에서는 QD슛(무회전슛)은 게이지 3.5칸-4칸 사이, ZD슛(감아차기슛)은 게이지 4.0-4.5칸을 채우면 거리가 비정상적이지 않는 이상 들어간다고 보면 된다. 따라서 이 시뮬레이션에서도 QD슛은 power가 35-40, ZD슛은 power가 40-45 정도되면 들어가게 구현했다.

[3. 코드 설명]

우선 코드 설명을 하기 전에 간략하게 구현한 요소를 정리하고 넘어가겠다.

1. 다양한 위치와 각도에서 연습하기 위해 특정 범위 내 랜덤한 프리킥 위치 설정
2. 차기 전 슛 방향 설정 기능
3. 커맨드 입력에 따라 구사할 프리킥 결정 기능
4. 인게임과 동일한 슛게이지 및 타이밍 적용
5. 인게임과 동일한 환경 적용 (수비벽, 커맨드입력, 감아차기 시 감기는 정도)

참고로 코드 구현에서 xz 평면으로만 바라보는 구도가 많이 나온다. 그래서 2dpos 라는 변수명이 많은데 이 변수들은 객체들을 xz평면으로 정사영시킨 도형의 중심좌표를 나타낸다. 이 2dpos를 바탕으로 정해지는 벡터들이 많다.

1. 구현 환경 설정

scene 기본설정 및 물리법칙 구현을 위한 상수값 설정

```
scene = canvas(width = 1400, height = 600, title = "피파온라인4")
scene.background = color.white

# constants
g = 9.8
rho = 1.204 # air density
Cd = 0.3 # air resistance
Cm = 1 # air magnus
w = 15*pi #각속도
```

2. 축구장 및 골대 설치

축구장이랑 골대를 만든다. left_post right_post top_post를 만들어 합치면 골대가 된다.

```
ground = box(pos=vec(0,0,0),size=vec(50,0.1,50),color=color.green)

# box center = mid position -> if height changes pos.y have to change
post1 = box(pos=vec(-4,2,-20),axis=vec(1,0,0),size=vec(0.5,4,2),color=color.red) # left post
post2 = box(pos=vec(4,2,-20),axis=vec(1,0,0),size=vec(0.5,4,2),color=color.red) # right post
post3 = box(pos=vec(0,4.25,-20),axis = vec(1,0,0),size=vec(8.5,0.5,2),color=color.red) # top post
```

3. 랜덤한 프리킥 위치 설정

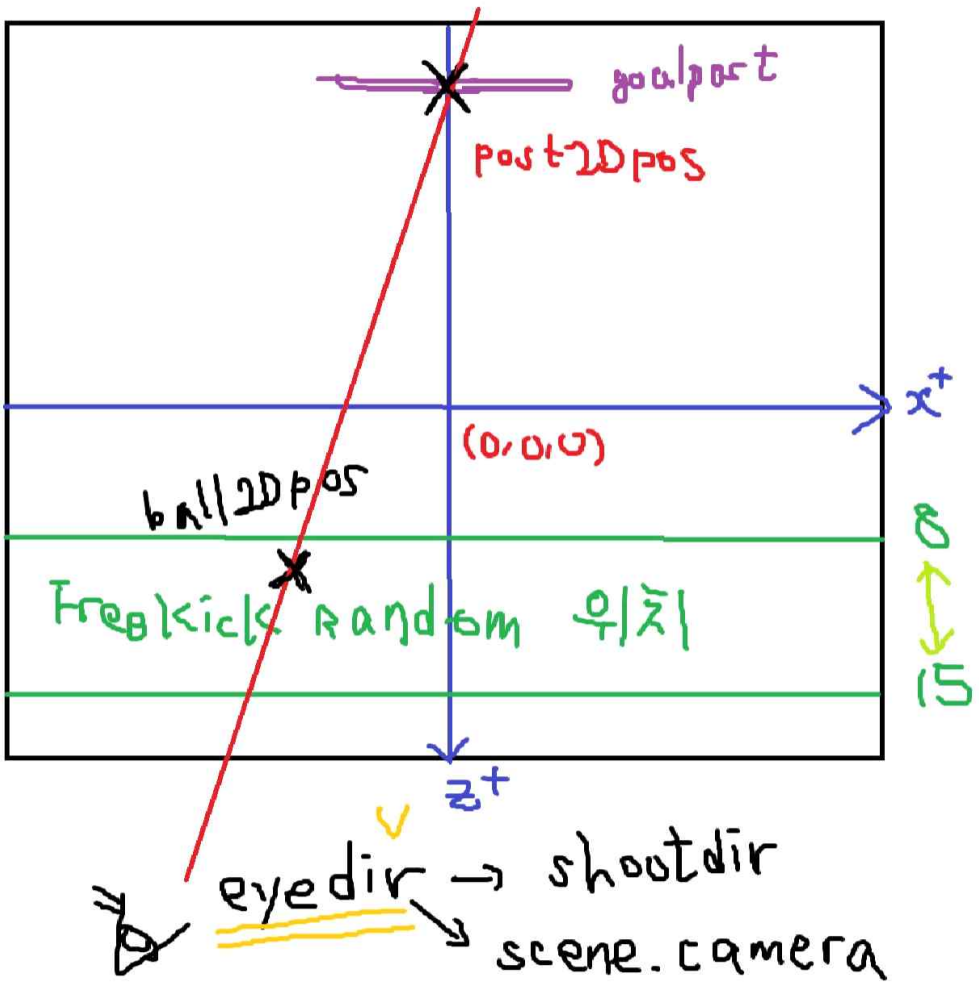
ball을 -15<=x<=15, 8<=z<=15 인 축구장 범위 내에 지정한다. ball2dpos는 ball의 xz평면에서의 위치를 말한다. 이 ball2dpos 변수는 게임 내에서 중요한 변수 중 하나이다. 이 변수는 camera view와 shootdir을 결정짓는데 중요한 역할을 하는 변수이다.

```
ball=sphere(pos=vec(random.randint(-15,15),0.3,random.randint(8,15)),radius=0.25,color=color.white)
ball.m = 1
ball.r = 0.2

ball2dpos = ball.pos - vec(0,0.25,0)
```

4. shootdir 및 camera view 설정

게임 내 가장 중요한 변수 및 설정들이다. 이들은 post2dpos와 랜덤으로 설정된 ball의 ball2dpos 로 결정된다. 아래 그림을 참고하면서 코드를 해석하면 된다.




```
eyedir = hat(ball2dpos-post2dpos)
eye2dpos = vec(ball.pos.x/(ball.pos.z+20)*(18+20),0,18)
eyepos = eye2dpos + vec(0,2,0)

shootdir = arrow(pos=ball2dpos,axis=4*hat(ball2dpos-eye2dpos),shaftwidth=0.15,color=color.black)

scene.camera.pos = eyepos
scene.camera.axis = shootdir.axis
```

post2dpos 는 골포스트의 xz평면에서의 중심좌표이다. 우리는 ball에서 post 중심을 바라보는 시점을 계속 유지하고 싶으므로 시점의 방향벡터인 eyedir를 구해놔준다. 이 eyedir를 통해 shootdir과 camera.pos 까지 설정해준다.

5. shootbtn

shoot button이다. 이 버튼을 누르면 찰 준비가 되었다는 뜻이므로 시야가 고정되고(shootvec가 고정되고) 프리킥을 차기만 하면 된다.

```
def shootbtn(b) :
    b.disabled=True
    return b.disabled
```

6. check_hit() functions

ground, goalpost에 맞았을 때 ball의 속성을 바꿔주는 함수들이다. 이 함수들은 각 shoot 함수들 내에서 해당 함수가 종료될때까지 계속 hit 유무를 확인해준다.

```
def check_groundhit() :
    if ball.pos.y < 0.3 :
        ball.pos.y = 0.3
        ball.v.y = -0.8*ball.v.y

def check_goalposthit() :
    # right left post hit
    if 3.75 <= abs(ball.pos.x) <= 4.25 and 0 <= ball.pos.y <= 4 and -21 <= ball.pos.z <= -19 :
        ball.v.z = -0.3*ball.v.z
    # top post hit
    elif -3.75 < ball.pos.x < 4.25 and 4 < ball.pos.y < 4.5 and -21 <= ball.pos.z <= -19 :
        ball.v.y = 0.8*ball.v.y
        ball.v.z = -0.5*ball.v.z
```

7. check_goal()

골 판단을 해주는 함수이다. 5번의 check_hit 함수들과 같이 각 shoot 함수 내 while문에서 반복해서 확인해준다. 만약 골대 내부로 들어갔으면 미리 준비되어있는 goal_label을 표시해준다.

```
def check_goal() :
    if abs(ball.pos.x) < 3.5 and 0 <= ball.pos.y < 4 and -21<= ball.pos.z <=-19 :
        goal_label.visible = True
```

8. label 설정

골 유무를 화면에 나타내는 label 설정이다. 시작 전에 미리 만들어놓고 초기설정으로 visible = False로 놓는다. 추후에 한 번의 프리킥이 종료되면 골 유무에 따라 visible을 True로 바꿔놓는다.

```
goal_label = label(pos=post2dpos+vec(0,10,0),box=False,height = 30,text='GOAL!',color=color.blue)
goal_label.visible = False
miss_label = label(pos=post2dpos+vec(0,10,0),box=False,height=30,text='프리킥더연습하세요',
color=color.blue)
miss_label.visible = False
```

9. shoot_directd()

무회전 프리킥 함수이다. 인자로 무회전슛에 대한 shoot_power값을 받아 해당 shootpower만큼 무회전 슛을 차게한다. 무회전 슛은 공을 감는 것이 아닌 일직선의 궤도로 가게하는 것이므로 “ball.v = direct_d * hat(shootvec) + vec(0,direct_d/5,0)” 즉, 정해진 방향인 hat(shootvec)에 인자로 들어온 direct_d shootpower값만큼 곱해주고 y값으로 direct_d/5 만큼 더해준다.

```
def shoot_directd(direct_d) :

    # direct_d have no deceleration in the given power
    ball.v = direct_d*hat(shootvec)+vec(0,direct_d/5,0)

    dt = 0.01
    t = 0

    while t<2.5 :
        rate(1/dt)
        ball.v+=vec(0,-g*ball.m,0)*dt
        ball.pos += ball.v*dt

        check_wallhit(wall)
        check_groundhit()
        check_goalposthit()
        check_goal()

        t+=dt
```

10. shoot_rightzd()

오른쪽 방향 감아차기이다. 감아차는 것이므로 정해진 shootvec로 차는 것이 아닌 살짝 오른쪽 대각선 방향으로 차야한다. 또한, 감아차기의 shootpower는 공을 때리는 power라기보단 공을 감는 정도를 나타내므로 프리킥에 인자로 들어온 shootpower인 right_zd가 100% 반영되지는 않는다. 따라서 ball.v에 인자로 들어온 shootpower인 right_zd/2 만큼 곱해주고 vec(right_zd/3,right_zd/3,-right_zd/4)을 더해주어 감아차기를 구현해준다.

```
def shoot_rightzd(right_zd) :

    ball.v = right_zd/2*hat(shootvec)+vec(right_zd/3,right_zd/3,-right_zd/4)

    dt = 0.01
    t = 0

    while t<2.5 :
        rate(1/dt)

        Fg = vec(0,-g*ball.m,0)
        Fd = -0.5*Cd*rho*(pi*ball.r**2)*mag(ball.v)**2*hat(ball.v)
        Fm = 0.5*Cm*rho*(pi*ball.r**2)*ball.r*mag(ball.v)*w*cross(vec(0,1,0),hat(ball.v))

        Fnet = Fg+Fd+Fm
        ball.a = Fnet/ball.m

        ball.v+=ball.a*dt
        ball.pos+=ball.v*dt

        check_wallhit(wall)
        check_groundhit()
        check_goalposthit()
        check_goal()

        t+=dt
```

11. shoot_leftzd()

왼쪽방향 감아차기. 오른쪽방향 감아차기와 코드논리는 똑같다. 감아차기이므로 슛파워가 입력된 슛파워만큼 반영되지 않고 정해진 shootvec 보다 살짝 왼쪽 대각선으로 차져야한다.

```
def shoot_leftzd(left_zd) :

    ball.v = left_zd/2*hat(shootvec)+vec(-left_zd/3,left_zd/3,-left_zd/4)

    dt = 0.01
    t = 0

    while t<2.5 :
        rate(1/dt)

        Fg = vec(0,-g*ball.m,0)
        Fd = -0.5*Cd*rho*(pi*ball.r**2)*mag(ball.v)**2*hat(ball.v)
        Fm = 0.5*Cm*rho*(pi*ball.r**2)*ball.r*mag(ball.v)*w*cross(vec(0,-1,0),hat(ball.v))

        Fnet = Fg+Fd+Fm
        ball.a = Fnet/ball.m

        ball.v+=ball.a*dt
        ball.pos+=ball.v*dt

        check_wallhit(wall)
        check_groundhit()
        check_goalposthit()
        check_goal()

        t+=dt
```

12. camera control

인게임 내에서 프리킥 방향을 left right 방향키로 조절할 수 있듯이 해당 시뮬레이션 내에서도 keysdown() 함수로 user가 입력한 방향키를 받고 이를 토대로 scene.camera.rotate() 함수를 통해 조절가능하게 만들었다. camera control은 shoot 버튼이 눌리기 전에만 가능하다.

```
angle = radians(1)
axis = vec(0,1,0)
origin = ball.pos

while btnShoot.disabled==False :
    rate(100)
    if btnShoot.disabled : break

    s = keysdown()

    if 'left' in s:
        scene.camera.rotate(angle=radians(1), axis=axis, origin=origin)
        shootdir.axis=6*hat(scene.camera.axis)
    if 'right' in s:
        scene.camera.rotate(angle=radians(-1), axis=axis, origin=origin)
        shootdir.axis=6*hat(scene.camera.axis)
```

13. before freekick setting

이제 shoot버튼이 눌려서 shootdir이 결정되었으면 shootdir.axis를 shootvec 변수에 저장한다. 이 변수는 이제 변하면 안되는 상수여야하는 벡터이다. right_zd left_zd direct_d 는 각각 해당 프리킥에 대응하는 shootpower 값이다. dpower는 미소power이다. 누르는 만큼 +dpower가 된다.

```
# the most important variable
# used in final shooting
shootvec = shootdir.axis

right_zd = 0
left_zd = 0
direct_d = 0

dpower = 1 # power+=dpower

scene.waitfor('keydown')
```

14. shootpower 입력받기

btnShoot이 눌렸다는 것으로부터 shootpower를 입력받을 준비가 되었다는 것을 확인하고, 피파온라인4 커맨드를 그대로 따라 어떤 종류의 프리킥을 구사할지가 결정된다. 결정된 프리킥 종류를 바탕으로 해당 프리킥 함수를 호출한다.

```
while btnShoot.disabled :
        rate(100)  # Limit the loop rate for smooth animation
        s = keysdown()  # Get the keys that are currently pressed

        # right zd
        if 'z' in s and 'right' in s : right_zd += dpower
        # left zd
        else if 'z' in s and 'left' in s : left_zd += dpower
        # direct_d
        else if 'q' in s and 'd' in s : direct_d +=dpower
        else if s==[] :
            break
# 프리킥 구사하기
if right_zd > 0 :
    shoot_rightzd(right_zd)
else if left_zd > 0 :
    shoot_leftzd(left_zd)
else if direct_d > 0 :
    shoot_directd(direct_d)
```

15. after freekick

프리킥 함수가 종료되면 골인지 아닌지가 결정되어있다. 따라서 프리킥 함수가 종료되면 이에 따른 결과를 화면에 나타내야한다. 기존에 미리 생성된 goal_label과 miss_label을 상황에 맞게 화면에 보이게 한다. .visible 속성을 사용한다.

그 이후에는 한번의 프리킥이 끝났다는 것이므로 해당 시뮬레이션의 모든 객체를 .visible=False를 사용해 안보이게 하고 새로운 시뮬레이션을 할 수 있도록 한다. 무한 플레이가 가능하도록 전체 프리킥 차는 코드를 while문으로 감싸주었다.

```
    if !goal_label.visible :
        miss_label.visible = True

    scene.waitfor('click')
    goal_label.visible = False
    miss_label.visible = False
    btnShoot.disabled = False
    ball.visible = False
    shootdir.visible = False
    wall.visible = False
```

16. 수비벽

상대 수비벽을 세운다. 벽을 구성하는 선수들은 프리킥을 차는 사람쪽을 바라보고 서있으므로 벽의 axis 벡터는 차는 방향인 shootvec와는 수직이여야한다. 벽의 axis 벡터를 cross 함수를 통해 구해준다. 또한 수비벽과 차는위치의 거리도 15m로 결정지어 놓는다.

```
linepos = ball.pos + hat(shootdir.axis)*15
wall = box(pos=linepos+vec(0,1,0),axis=cross(vec(0,1,0),shootdir.axis), size = vec(3,2,0.2),
texture=textures.wood)
```

17. 수비벽과의 충돌

프리킥으로 찬 공이 수비벽에 맞으면 ball.v 속성값을 바꿔준다. 3차원에서 공과 벽의 충돌을 정확하게 계산해보려고 했지만 구글링과 Chatgpt를 동원해도 해답이 보이지 않아 일단 구와 구가 충돌하는 코드로 적어놨다. 이 코드도 위에 ground post hit함수들과 같이 프리킥 함수가 종료되기 전까지 지속적으로 호출된다.

```
# 수비벽 충돌
def checkwallhit() :
    if mag(ball.pos - wall.pos) < 1.2 :
        ball.v.y = 0.8*ball.v.y
        ball.v.z = -0.5*ball.v.z
```