1월 2주차 레포트

A. Frame 수정

[문제점(가정)]

1. 코드가 돌아갈 때 많은 계산량 때문에 속도가 느려졌다

(1). UnitBox 그리는 방식 문제

```
## 기존의 코드

def Drawunitbox(self):
        glBegin(GL_QUADS)
        glNormal3f(1,0,0)
        glColor3f(0.5,0.5,0.0)
        glVertex3f(0.5,-0.5,-0.5)
        glVertex3f(0.5,-0.5,0.5)
        glVertex3f(0.5,0.5,0.5)
        glVertex3f(0.5,0.5,-0.5)
        glEnd() ....

>> 이런 방식으로 일일이 점들과 거기에 해당하는 Normal Vector를 구해서
>> 사각형을 일일이 만듦

## 바꿀려는 코드(미완성->구현X)

def DrawUnit(self):
        Unit=np.array([
            [1,0,0],
            [0.5, -0.5, 0.5],
            [1,0,0],
            [0.5, -0.5, -0.5],
            [1,0,0],
            [0.5,0.5,-0.5], ...

        glEnableClientState(GL_VERTEX_ARRAY)
        glEnableClientState(GL_NORMAL_ARRAY)
        glNormalPointer(GL_FLOAT,6*Unit.itemsize,Unit)
        glVertexPointer(3, GL_FLOAT, 6*Unit.itemsize,ctypes.c_void_p(Unit.ctypes.data+3*Unit.itemsize))
        glDrawArrays(GL_TRIANGLES,0,int(Unit.size/6))

>> 각각의 점들과 Normal Vector로 이루어진 array 생성
>> 이러한 연속된 array를 통해서 한꺼번에 그리는 방식으로 하면
>> 조금 더 빠르게 시뮬레이션이 진행될 수 있지 않을까 생각
```

(2). Walk BVH 파일에서 Desired Position 추출 문제

```
## 기존의 코드

Z_angle=self.walkfile.frame_joint_channel(self.frame,i,'ZROTATION')*math.pi/180
X_angle=self.walkfile.frame_joint_channel(self.frame,i,'XROTATION')*math.pi/180
Y_angle=self.walkfile.frame_joint_channel(self.frame,i,'YROTATION')*math.pi/180

>> 각각의 Joint에서 Desired Position을 구할 때마다 BVH 파일 읽음
>> 이 과정에서 계산량을 많이 요구했을 것을 보임

## 바꾼 코드

def BVH_read(self):
        f = open("sample-walk.bvh",'r')
        while self.Num < 99:
            lines = f.readline()
            self.Num +=1
        if self.Num >=99:
            while self.Num <298 :
                lines = f.readline()
                lines = lines.split()
```

```
                lines = np.array(list(map(float,lines))).T
                self.BVH_pose = np.vstack([self.BVH_pose,lines])
                self.Num +=1
            self.BVH_pose = np.delete(self.BVH_pose,(0),axis=0)
        f.close()


Z_angle=self.BVH_pose[self.frame][3*i+3]*math.pi/180
X_angle=self.BVH_pose[self.frame][3*i+4]*math.pi/180
Y_angle=self.BVH_pose[self.frame][3*i+5]*math.pi/180

>> 처음에 한번 BVH 파일을 읽어서 각각의 frame에 해당하는 position 정보를 지닌 array 생성
>> Desired Position이 필요할 시, 이때의 array를 가져온다
>> 그나마 기존의 코드보다는 조금 더 빠르게 계산이 진행되는 것으로 보임
```

2. Frame 자체를 잘못 설정하였다.

(기존)

>> 1초당 30번 업데이트, 1번 업데이트할 시 30번 시뮬레이션 진행
>> 1초당 30번 업데이트(pose frame 업데이트)
>> 1초당 900번 시뮬레이션

(변경)

>> 1초당 30번 업데이트, 1번 업데이트할 시 90번 시뮬레이션 진행
>> 1초당 30번 업데이트(pose frame 업데이트)
>> 1초당 2700번 시뮬레이션

>> 1-(2) 방법을 사용할 시
>> 업데이트당 시뮬레이션 진행이 3배 늘었지만 오히려
>> 계산량이 빨라진 것을 확인할 수 있었음

B. SPD Control - Dart Tutorials (C++)

1. SPD Control 코드

```
## Dart 홈페이지에서 Tutorials C++ Biped 부분에 나온
## SPD 구현 함수

void addSPDForces()
{
    Eigen::VectorXd q = mBiped->getPositions();
    Eigen::VectorXd dq = mBiped->getVelocities();

    Eigen::MatrixXd invM = (mBiped->getMassMatrix() + mKd * mBiped->getTimeStep()).inverse();
    Eigen::VectorXd p = -mKp * (q + dq * mBiped->getTimeStep() - mTargetPositions);
    Eigen::VectorXd d = -mKd * dq;
    Eigen::VectorXd qddot = invM * (-mBiped->getCoriolisAndGravityForces() + p + d + mBiped->getConstraintForces());

    mForces += p + d - mKd * qddot * mBiped->getTimeStep();
    mBiped->setForces(mForces);
}
```

```
## 위의 addSPDForces에서 사용되는 수식에 대한 간단한 설명

Body 15개, Joint 형식, free 1개, ball 14개
>> free요소 6개 / ball요소 3개
>> 6 * 1EA + 3 * 14EA = 48
```

```
Skeleton.getCoriolisAndGravityForces()
>> [ .... ]
>> shape : (48,)

Skeleton.getMassMatrix()
>> [ [ .... ], [ .... ] , .... , [ .... ]]
>> shape : (48,48)

Skeleton.getConstraintForces()
>> [ .... ]
>> shape : (48,)


## 코드 해석(변환 >> C++ >> Python)

Eigen::VectorXd q = mBiped->getPositions();
>> q = Skeleton.getPositions()

Eigen::VectorXd dq = mBiped->getVelocities();
>> dq = Skeleton.getVelocities()

Eigen::MatrixXd invM = (mBiped->getMassMatrix() + mKd * mBiped->getTimeStep()).inverse();
>> invM = np.linalg.inv(Skeleton.getMassMatrix() + mKd * World.getTimeStep())

Eigen::VectorXd p = -mKp * (q + dq * mBiped->getTimeStep() - mTargetPositions);
>> p = -mKp * (q + dq * World.getTimeStep() - mTargetPositions

Eigen::VectorXd d = -mKd * dq
>> d = -mKd * dq

Eigen::VectorXd qddot = invM * (-mBiped->getCoriolisAndGravityForces() + p + d + mBiped->getConstraintForces());
>> qddot = invM @ ( -Skeleton.getCoriolisAndGarvityForces() + d + Skeleton.getConstraintForces())

mForces += p + d - mKd * qddot * mBiped->getTimeStep();
>> Torque = p + d - mKd * qddot * World.getTimeStep()

mBiped->setForces(mForces);
>> Skeleton.setForces(Torque)
```

```
## 코드 구현

def SPD_control(self):
        Kp=10
        Kd=0.5

        D_position=np.zeros(48)

        for i in range(48):
            D_position[i] = self.BVH_pose[self.frame][i]*math.pi/180

        position = self.Human.getPositions()
        velocity = self.Human.getVelocities()

        invM = np.linalg.inv(self.Human.getMassMatrix() + Kd * self.world.getTimeStep())
        PP = -Kp * (position + velocity * self.world.getTimeStep() - D_position)
        DD = -Kd * velocity
        QDDOT = invM @ (-self.Human.getCoriolisAndGravityForces() + PP + DD + self.Human.getConstraintForces())
        Torque = PP + DD - Kd * QDDOT * self.world.getTimeStep()

        for i in range(6):
            Torque[i] = 0

        self.Human.setForces(Torque)
```

2. 구현 동영상

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/69e6c65c-360c-4daa-ab9f-5372208f2b32/SPD__Kp__10__Kd__0.4__Update__30_HZ___Simulation__2700_Hz.mp4

(1). 문제점

>> 팔이 앞뒤로 움직이는 것이 아닌 위아래로 흔들린다

>> 다리는 앞뒤로 움직이는 것이 아닌 X자 형식으로 꼬였다

>> 아마도 Pose의 X,Y,Z 좌표를 잘못 넣은 것으로 보인다

>> Kp : 10, Kd : 0.4 정도인데도 움직임이 심하다

>> Kd : 0.5만 되어도 코어가 덤프된다

>> SPD는 PD보다 Gain이 크다

>> 전반적인 코드 수정 필요

(2). 의문점

>> Github에 들어와있는 다른 SPD를 참고할 시,

>> 대부분의 경우 Joint에 대한 Position과 Velocity가 아닌

>> Skeleton에 대한 Position과 Velocity를 구하던데

>> 차이가 없는 것인가?

```
Skeleton.getPositions()
>>
[ 1.00781660e-06 -1.11816046e-14  1.02148752e-12  3.37320223e-13
  1.90589865e-06 -8.98823862e-08 -1.27087349e-06  1.11817203e-14
 -3.86229395e-13  4.08307416e-07  9.24027153e-20 -7.69456816e-13
  2.63056890e-07  2.04007946e-08  8.71113517e-06  4.88697421e-16
 -2.85666886e-08 -1.22928351e-05  4.68772959e-15  1.53950656e-08
  7.15316683e-06  2.63056890e-07 -2.04007946e-08 -8.71113775e-06
  4.88703065e-16  2.85666886e-08  1.22928369e-05  4.68773100e-15
 -1.53950656e-08 -7.15316790e-06 -5.09291767e-06  1.11941094e-14
 -1.59268775e-11  6.43223157e-06 -1.03250307e-10  6.63239699e-11
 -2.34857224e-06  1.03273439e-10 -7.34233972e-10 -5.09291757e-06
  1.11943202e-14 -1.62006317e-11  6.43223214e-06 -1.05402278e-10
  6.76574948e-11 -2.34860946e-06  1.14958182e-10 -7.66307011e-10]

Skeleton.getJoint(0).getPositions()
>>
[ 1.00781660e-06 -1.11816046e-14  1.02148752e-12  3.37320223e-13
  1.90589865e-06 -8.98823862e-08]

>> Skeleton에서 getPositions로
>> 전체 Joint에 대한 getPositions가 이뤄지는 것으로 보인다.
>> 차이 없다!!
```

>> 차이가 있다면, Skeleton에 setForce를 하는 것과
>> Joint에 setForce를 하는것에서 혼동이 생겨 오류가 생긴 것인가?

```
## 출처 :
## https://github.com/j-rivero/dart_unified/blob/1673b0be51fb370023df7490dc49706b590d8f72/python/examples/biped_stand/main.py
for i in range(6):
    self.torques[i] = 0

self.skel.setForces(self.torques * 0.8)

>> SPD Control에서 제일 마지막부분을 보게 되면
>> Skeleton에 setForces를 하지만, 처음에 들어가는 6개
>> 즉, free joint를 지닌 것들에 대하여 torque가 없어야하므로
>> Skeleton에 setForces는 하는 것과
>> Joint 각각에 setForeces는 하는 것은 동일한 것으로 보임
```

(3). 해결

>> 아마도 Pose의 X,Y,Z 좌표를 잘못 넣은 것으로 보인다

>> Z,X,Y 순으로 되어있던 BVH의 Desired Position
>> X,Y,Z 순으로 바꾸고 SPD 구현 시 제대로 작동

```
def Change_SPD(self,D_position):
        ## BVH 파일에서의 ZROT, XROT, YROT 순서를
        ## XROT, YROT, ZROT 순서로 바꾸어야 한다
        ## D_position은 (48,)
        ## (3 ~ 45까지 바꿔야 한다.)
        ## i= 1 ~ 15까지 진행한다.
        for i in range(1,16):
            Num = D_position[3*i]
            D_position[3*i] = D_position[3*i+1]
            D_position[3*i+1] = D_position[3*i+2]
            D_position[3*i+2] = Num

>> Z,X,Y순으로 배치되어있던 요소들
>> X <- Y <- Z 순으로 재배치
```

>> Kd : 0.5만 되어도 코어가 덤프된다

>> Kd는 0.4보다 커질 시 코어가 덤프된다
>> 하지만 높은 Kp에서도 안정적으로 구현된다
>> Kp가 올라갈 수록 다리와 팔다리 움직임이 안정적
>> 하지만 처음에 움직임 시작 시 진동 심함

(4). 보완점

>> Kp가 커지니 안정시점을 지난 걷기 동작은 제대로 구현

>> 하지만 처음 시뮬레이션 시작 시 진동 심함
>> 그리고 Kp가 크다보니 캐릭터가 하늘 위로 떠올라감

>> 카메라로 캐릭터에 고정되게 관찰하면 좋을 것

```
def Skel_Camera(self):
        R_camera = np.zeros(3)
        Normal = np.array([0,0,3])
        Vertical = np.array([1,0,0])

        focus = np.array(self.Human.getBodyNode(0).getWorldTransform().translation())
        angle = np.array(self.Human.getJoint(0).getPositions())

        Z_angle=angle[0]
        X_angle=angle[1]
        Y_angle=angle[2]

        Z_ROT=np.array([[np.cos(Z_angle),-np.sin(Z_angle),0],
                        [np.sin(Z_angle),np.cos(Z_angle),0],
                        [0,0,1]])
        X_ROT=np.array([[1,0,0],
                        [0,np.cos(X_angle),-np.sin(X_angle)],
                        [0,np.sin(X_angle),np.cos(X_angle)]])
        Y_ROT=np.array([[np.cos(Y_angle),0,np.sin(Y_angle)],
                        [0,1,0],
                        [-np.sin(Y_angle),0,np.cos(Y_angle)]])

        Normal = Z_ROT @ X_ROT @ Y_ROT @ Normal
        Vertical = Z_ROT @ X_ROT @ Y_ROT @ Vertical

        for i in range(3):
            R_camera[i] = focus[i] + Normal[i]
```

```
        Upvec=np.cross(Vertical,-Normal)

        glLoadIdentity()
        gluLookAt(R_camera[0],R_camera[1],R_camera[2],focus[0],focus[1],focus[2],0,1,0)

>> 카메라의 Focus는 Heap BodyNode의 World 좌표계
>> 카메라의 Eye는 Heap 정면부의 Normal Vector + Focus Vector
>> 카메라의 Up은 Heap 정면부의 Normal Vector와 Vertical Unit Vector의 외적
  >> 하지만 제대로 구현되지 못함
```

>> 제일 좋은 것은 안 올라가고 잘 걷는 것