2023학년도 1학기

컴퓨터애니메이션실습

컴퓨터 애니메이션

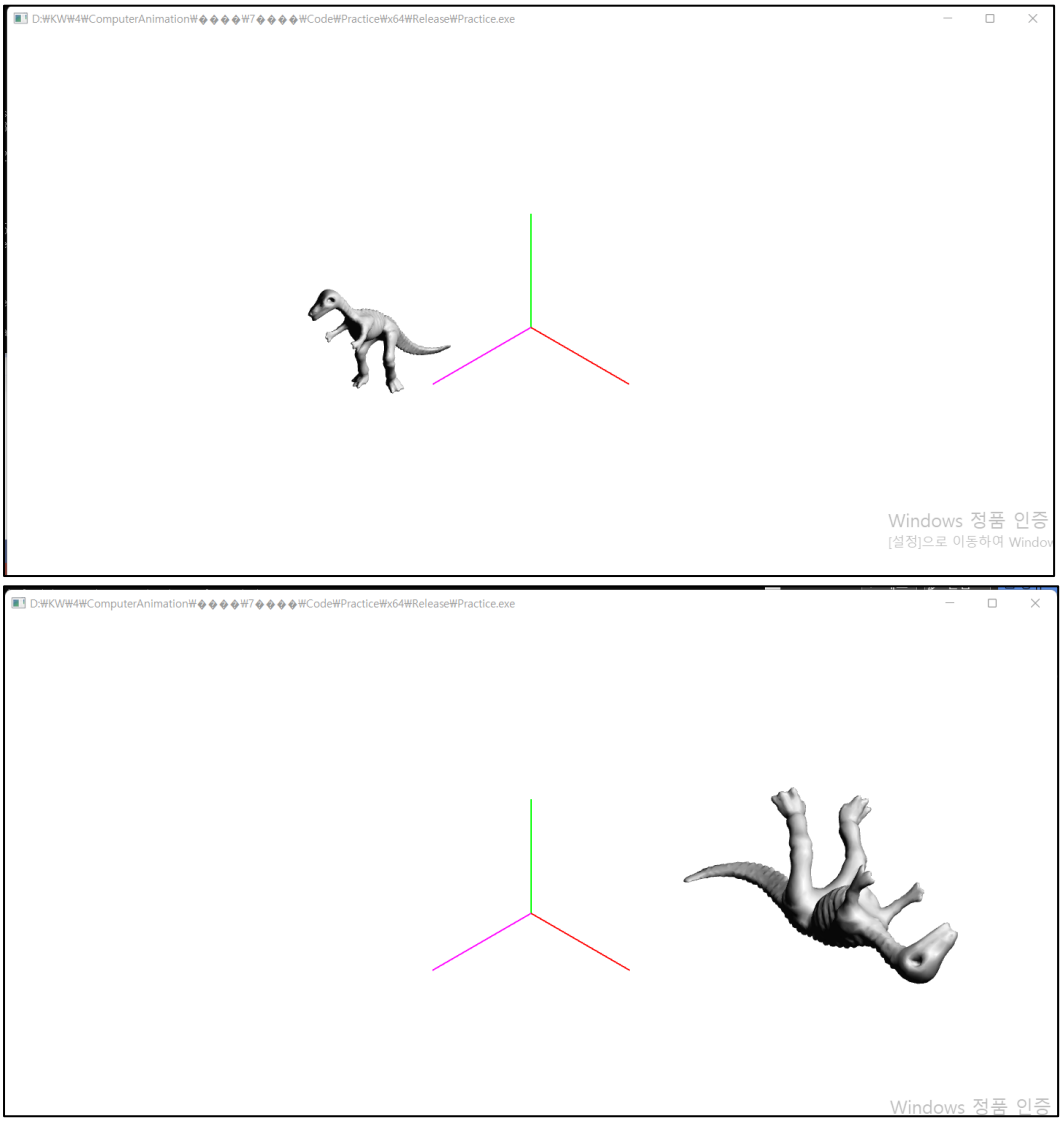
실습 보고서

Self-Scoring Table

	P1	P2	P3	E1
Score	1	1	1	1

정보융합학부 2018204058 김민교

P1 - Linear interpolation of rotation matrices

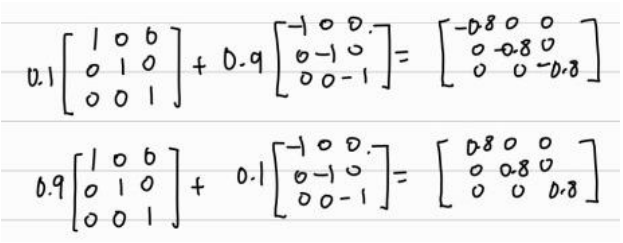


현재 회전하는 각도가 거의 180도에 수렴한다. (theta = 181)

이런 경우,

q1 to Rotation = $\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ q2 to Rotation = $\begin{matrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{matrix}$ 라고 대략 가정했을 때,

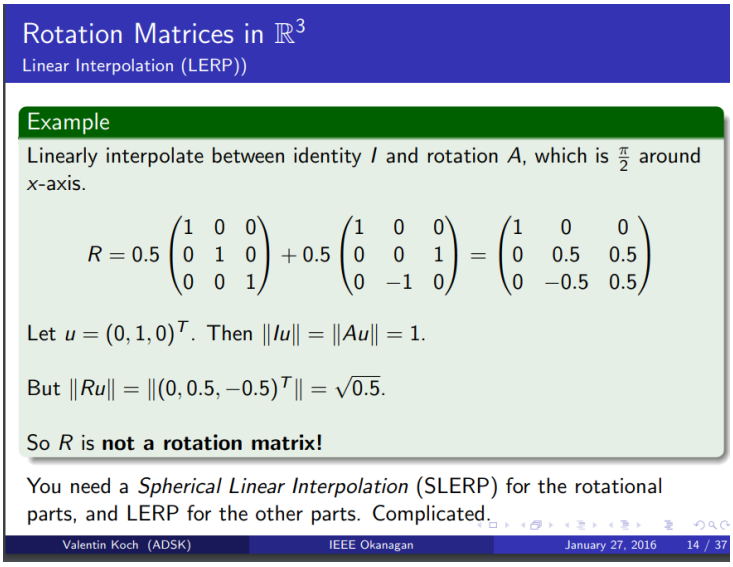
Linear Interpolation을 하게 되면 t가 0~0.5 일 때는, q1 Rotation의 회전만 살아있고, 회전행렬의 determinant의 크기가 1보다 작아진다. 그래서 그림과 같이 공룡이 작아진다. t가 0.5~1일 때에는 q2 Rotation의 회전만 살아있고, 회전행렬의 determinant 크기가 -1에 점점 가까워진다. 그래서 그림과 같이 공룡이 커진다.



theta가 180에 가까울수록, Orientation이 극단적으로 두 가지만 나타난다. 만약 theta을 180보다

멀게 설정한다면, 여러 Orientation이 나오게 되겠지만, $R(t) = (1-t)R_1 + (t)R_2$ 의 결과는 회전행렬을 반환하지 않는다. (Orthogonal 하지 않다)

ex) 계산 예시



*) 참고 자료

http://cau.ac.kr/~kjpark/courses/2008_2_ca/Lec5.pdf

http://www.okanagan.ieee.ca/wp-content/uploads/2016/02/GimbalLockPresentationJan2016.pdf

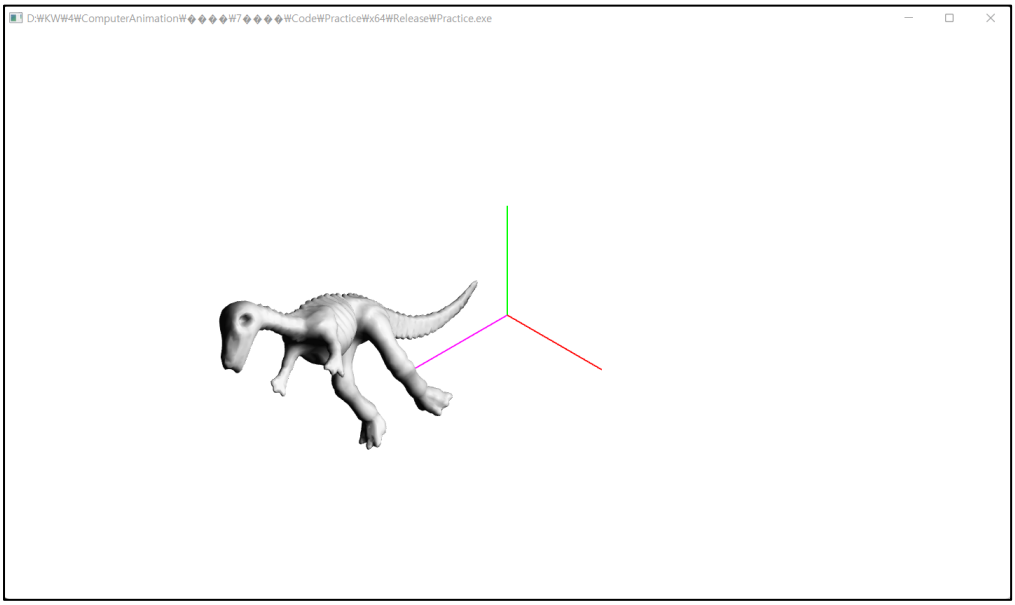
https://courses.cs.duke.edu/compsci344/spring15/classwork/11_rotations/

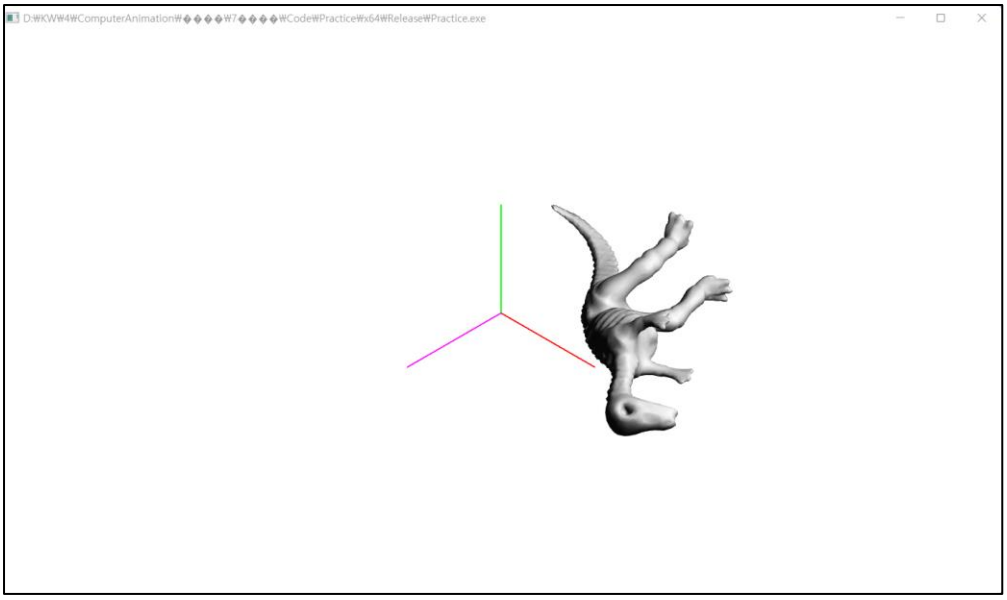
https://thenumb.at/Exponential-Rotations/

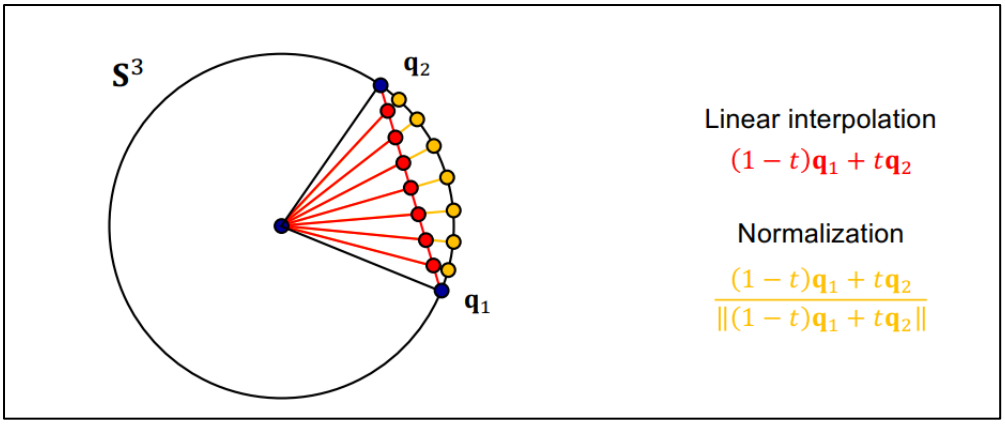
https://users.aalto.fi/~lehtinj7/CS-C3100/2020/slides/6.1.rotation.representations.pdf

https://gitcgr.hanyang.ac.kr/courses/2021-spring-cg/lecture-slides/9-Orientation&Rotation.pdf

P2 - Linear interpolation of unit quaternions with normalization



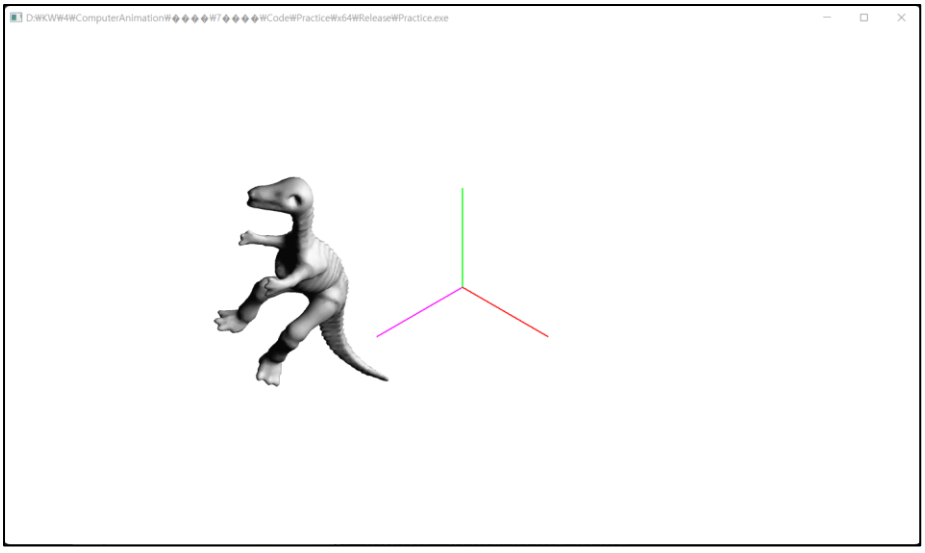




Linear interpolation 하면 균s등하게 나뉘어지지 않는다. 그래서 천천히 회전하다가 빨리 회전하고 또 다시 천천히 회전하게 된다.

antipodal equivalence가 적용되지 않았다. 그래서 P3과 다르게 회전방향이 반대이다. 즉 회전축이 반대이다.

P3 - Spherical linear interpolation using Eigen's slerp



slerp을 적용한 결과이다. Linear interpolation의 균등하게 각도가 나뉘지 않는다는 문제점이 해결된다. Antipodal equivalence가 적용되었다.

```
q2 from theta = 181 and axis = 0.57735 0.57735 0.57735
q2 = -0.00872665, 0.577328 0.577328 0.577328
```

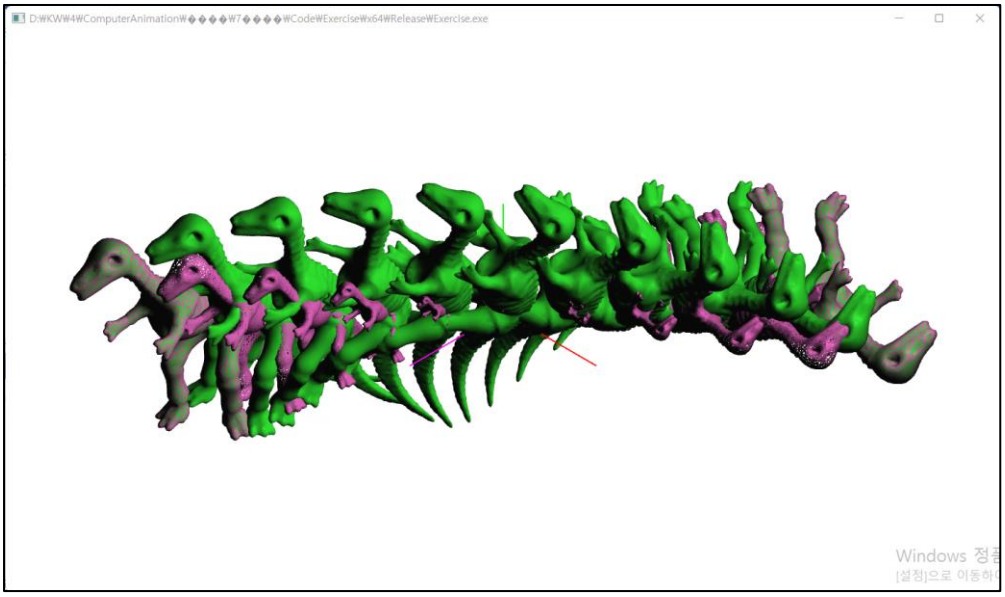
q2의 원래 회전축은 1,1,1이다. 이는 외적 오른손법칙에 의해서 공룡이 반시계 방향으로 회전함을 의미한다.

```
q1 -> q2 : Angle = 179 degree, Axis = -0.57735 -0.57735 -0.57735
```

Antipodal equivalence가 적용되면 회전축이 반대방향이 된다. 이는 외적 오른손법칙에 의해서 공룡이 시계방향으로 회전함을 의미한다.

공룡의 회전방향을 통해 Antipodal equivalence의 적용 유무를 알 수 있다.

E1 - Visual comparison between LERP of rotation matrices and SLERP of quaternions



```
68
69      Matrix4f Ts[11] = {};
70      Matrix4f Q_Ts[11] = {};
71
```

TS[11] : LERP T matrix를 0초부터 1초까지 0.1 간격으로 담는 배열이다.

Q_Ts[11] : SLERP T matrix를 0초부터 1초까지 0.1 간격으로 담는 배열이다.

<getTimeperT 함수>

```
247     void getTimeperT()
248     {
249         // Transformation
250         Matrix4f TTrans;
251         Matrix4f QTrans;            // Current orientation and position
252
253         // Initial transformation of the mesh
254         TTrans.setIdentity();
255         QTrans.setIdentity();
256
257         for (int i = 0; i <11; i++) {
258
259             float t = 0.1 * i;
260             Vector3f p = (1 - t) * p1 + t * p2;
261             TTrans.block<3, 1>(0, 3) = p;
262
263             // Linear interpolation of the given two positions in all cases
264             // Set the translational part in the 4x4 homogenous matrix
265             Vector3f p2 = (1 - t) * Q_p1 + t * Q_p2;
266             QTrans.block<3, 1>(0, 3) = p2;
267
268
269             // Linear interpolation of the given two rotation matrices
270             // Set the rotational part in the 4x4 homogenous matrix
271             TTrans.block<3, 3>(0, 0) = rlerp(t, q1, q2);
272
273             Quaternionf q;
274             // Slerp provided by Eigen
275             q = Q_q1.slerp(t, Q_q2);
276             // Set the 3x3 rotational part of the 4x4 homogeous matrix
277             QTrans.block<3, 3>(0, 0) = Matrix3f(q);
278
279             Ts[i] = TTrans;
280             Q_Ts[i] = QTrans;
281         }
282     }
283
284
285     // Draw a sphere after setting up its material
```

t값이 0, 0.1, 0.2…1 까지 11번 반복해주고, 이 때마다 Slerp T matrix와 LERP matrix를 구한다. 그리고 Ts와 Q_Ts배열에 각각 저장한다.

getTimeperT 함수는 main함수에서 한 번 호출한다.

<Render 함수>

```
344     // 계쏙 그린다.
345     for (int i = 0; i < 11; i++)
346     {
347         glPushMatrix();
348         glMultMatrixf(Ts[i].data());
349         glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); // Polygon을 Line으로 그린다.
350         drawMesh(vertex2, normal2, face2,true);
351         glPopMatrix();
352 
353         glPushMatrix();
354         glMultMatrixf(Q_Ts[i].data());
355         glPolygonMode(GL_FRONT_AND_BACK, GL_FILL); // Polygon을 면을 채우게끔 그린다.
356         drawMesh(vertex1, normal1, face1,false);
357         glPopMatrix();
358 
359     }
360 }
```

Render함수에서 0~1초 까지 0.1초씩의 LERP T Matrix를 현재 행렬에 곱해주고 메시를 그린다. 또 SLERP T Matrix를 현재 행렬에 곱해주고 메시를 그린다.