An introduction to coordinate system objects

Abstract: *Extensive use of matrix operations in coordinate system transformation operations, or tensor operations more generally. The former*
It belongs to the "meta" mathematical object, which is not conceptually designed for coordinate system transformation, and there are redundancy and convenience problems in the calculation process; The latter is difficult to grasp and more difficult to use numerically. This article describes a structural object encapsulated specifically for *coordinate system transformations as an alternative to matrix and tensor operations.*

1. Introduction

Coordinate systems have always been the most abstract and difficult mathematical physical objects, and after the development of tensor operations, they are even more abstract, and must be mastered after years of training. So try to use a simplified method to deal with the coordinate system operation problem. Coordinate system transformation is often carried out using matrices, but as mathematical objects, matrices are not customized for coordinate system changes, and matrices are too mathematical and ambiguous. Tensors are too abstract to be mastered not only by computers, but also by quantification, requiring concepts specifically designed for coordinate system transformation.

2. Design

Design a mathematical object called coord, and design four operations around it based on group theory.

a) Definition of coordinate system structure:

- The coordinate system in three-dimensional space consists of an origin plus three direction axes and three scaling components, corresponding to the three transformations of **displacement, rotation, and scale** (C++ version).

```
struct coord
{
    vec3 ux, uy, uz; Three unit basis vectors
    vec3 scale; zoom
    vec3 o; Origin Position
}
```

Construct a coordinate system object (C++ version).

Through three axes

coord C(vec3 ux, vec3 uy, vec3 uz)

coord C(vec3 ux, vec3 uy);

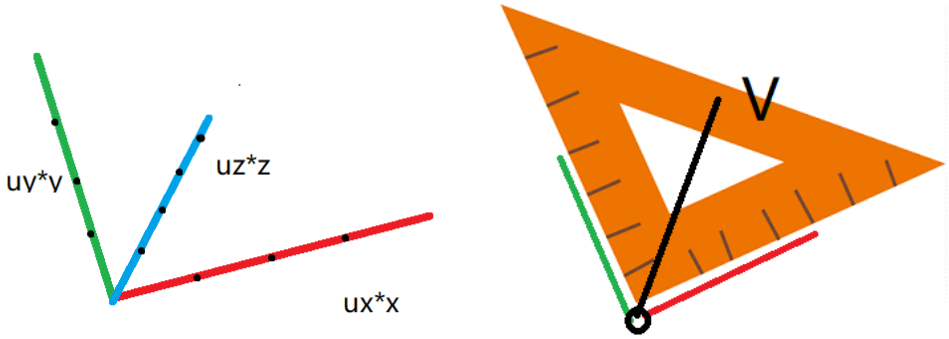
Through Cape Euler

coord C(float angle, vec3 axis);

coord C(float pitch, float yaw, float roll);

(The origin and zoom can be set directly)

b) Multiplication: Define a vector in a coordinate system



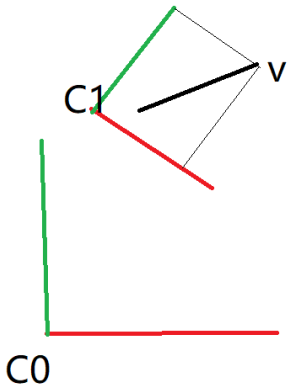
Use multiplication to realize the transformation of vectors from the local coordinate system to the parent coordinate system and the merge operation of the coordinate system

For example, V0 is defined in the world coordinate system C0, and V1 is defined in the C1 coordinate system

$$V0 = V1 * C1$$

C1, C2, and C3 are the local coordinate systems under the three scene nodes, parent-> child direction:

$$V0 = V3 * C3 * C2 * C1 = V3 * (C3 * C2 * C1)$$



Multiplication after swapping with vectors:

$$V * C != C * V$$

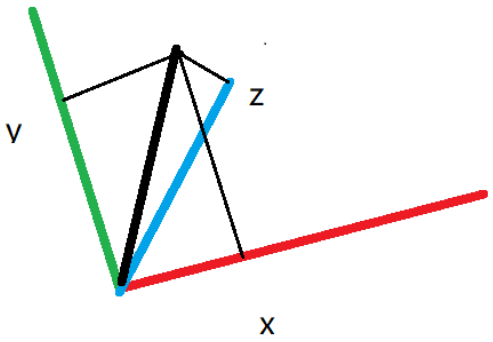
Define the coordinate system and multiply the vector:

$$C * V = Lerp(ONE, C, V.dot(UX))$$

where Lerp is a linear interpolation function and ONE = coord. ONE,

UX = vec3. UX

c) Division: Measure a vector with a certain coordinate system



Use division to project from the parent coordinate system to the local coordinate system, for example:

V0 is defined under the world coordinate system C0, and V1 is defined in the C1 coordinate system

V1 = V0 / C1, V0 = V1 * C1

V2 is defined in the C2 coordinate system:

V2 = V0 / C2 = V1 * C1 / C2

C1, C2, and C3 are the local coordinate systems under the three scene nodes, parent-> child directions, respectively

V3 = V0 / C1 / C2 / C3 = V0 / (C1 * C2 * C3)

d) Typical usage scenarios

1. For example, the vector Vw in a world space is transformed into the local coordinate system C

   VL = Vw / C

   Conversely: Vw = VL * C

2. The world coordinate system is converted to the local coordinate system

   C = C3 * C2 * C1

   Vw = VL* C, VL = Vw / C

3. Used under the multi-node hierarchy

   Define the vector V5 in the coordinate system of node 5, under the parent node of node 2 hierarchy:

   V2 = V5 * C5 * C4 * C3

   Each of these coordinate systems is local

   Conversely: V3 = V2 / C3 / C4 / C5

4. Swap between flat coordinate systems

   C0{ // C0 under the parent coordinate system

   C1, C2; Two flat sub-coordinate systems

   }

The vector is converted from C1 to C2

V2 = V1 * C1 / C2

5. More operations

Scalar multiplication:

C*k = C.scale * k

k is a floating-point number

Quaternion multiplication

C*q = C.ux(q.angle, q.axis);

C.uy(q.angle, q.axis);

C.uz(q.angle, q.axis);

q is a quaternion

Addition and subtraction

C1+o = C1+ o, C1-o = C1- o

o is a three-dimensional vector (VEC3).

e) Addition and subtraction

When calculating the difference between two vectors:

DV = V1 – V2

设 V1 = v * C1, V2 = v * C2

DV = v * (C1 – C2)

Here defines an addition and subtraction operation for basic addition and

subtraction of vectors:

V1 + V2 = v * (C1+ C2)

V1 - V2 = v * (C1- C2)

3. Applications in the field of differential geometry

a) The movement of the vector

Define a natural coordinate system (assuming it is a straight space, the vector can move at will without changing) under V, observe V under the curved space coordinate system, V is different at different points, so the coordinate system is related to the position, take the adjacent two points (1), (2) There are vectors V1, V2 at the point, corresponding to the coordinate system C1, C2, then:

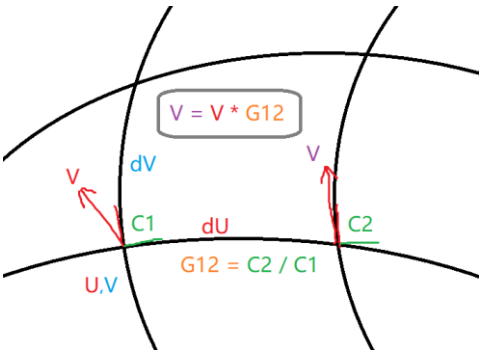
V = V1 * C1 = V2 * C2 =>

V2 = V1 * C1 / C2, 令 G12 = C1 / C2 =>

V2 = V1 * G12

To move in any direction of the differential vector:

Arbitrary differential vector: dW = Cw * dU



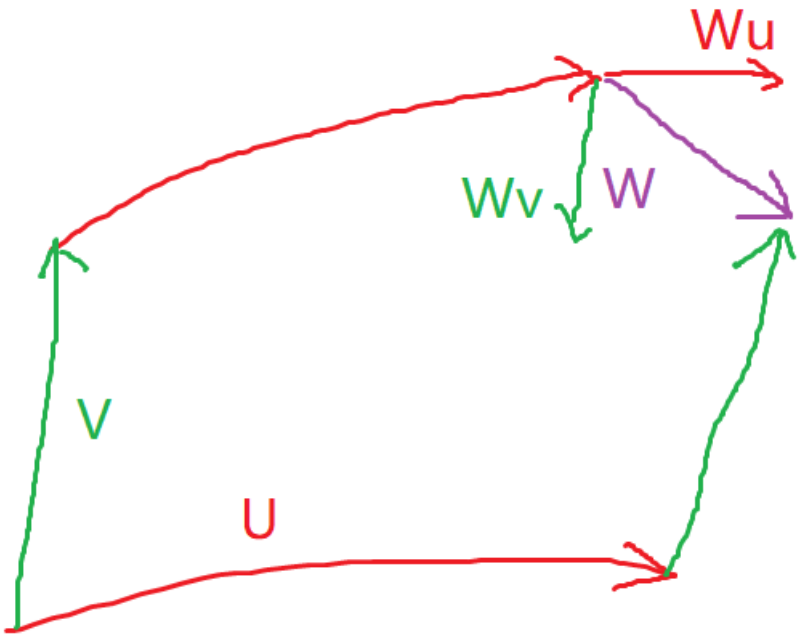
So move in any direction of the differential vector:

V2 = V1 * Gw = V1 * Gu * Cw

where: Gu = G12 = C1 / C2

The quotient of two coordinate system objects in the direction of the unit differential vector d U.

b) Curvature calculation



Under the curved coordinate system, the g12 curve obtained by the parallel line projection of the x,y axis of the natural coordinate system corresponds to Gu and Gv in two directions, respectively, from (u1,v1) To (u2,v2) to calculate the difference between the two paths plus the correction term, the curvature formula is:

Ruv = Gu*Gv - Gv*Gu*Gu*Wu*Gv*Wv

其中：W = Wu + Wv = Gu * Wu + Gv * Wv

4. Combination with Lie groups, Lie algebras

The rotation matrix R is an element of the Lie group, and the multiplication operation of our coordinate system is equivalent to the rotation matrix, so our coordinate system C is also an element of the Lie group, multiplication is the operation, and the 0 element is ONE:

$$C \in \{ONE, (*)\}$$

The vector fork multiplies the corresponding coordinate system

$$v1 \times v2 = v*C1 \times (v * C2)$$

Let v = vec3. ONE is vec3(1,0,0), then $C * v = C$, so the above equation can be written as:

$$v1 \times v2 = v*(C1 \times C2)$$

We define a coordinate system fork multiplication operation , expressed in Lie algebra brackets :

$$[C1, C2] = C1 * C2 - C2 * C1;$$

5. Conclusion

The coordinate system object can simplify the linear operation in local space, and some operations can be appropriately designed when calculating vector movement and rotation operations, and a unified form can be formed according to the rules of group theory. It can be applied to situations that require a lot of linear operations, such as constraint calculations. Vectors, matrices, and tensors are combined through coordinate system objects, and

Lie groups and Lie algebraic objects are unified into a ring, forming a unified and coherent form.