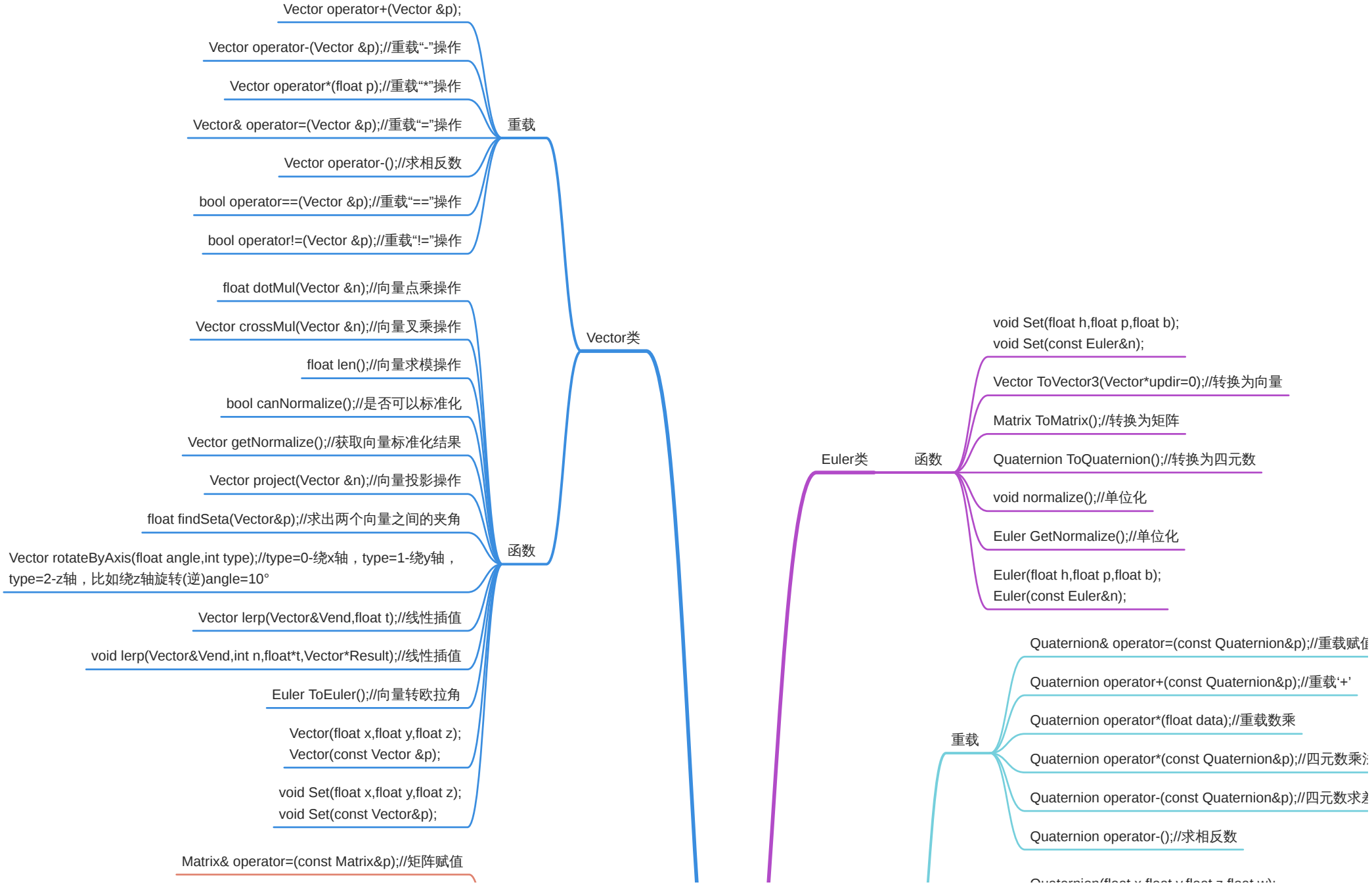
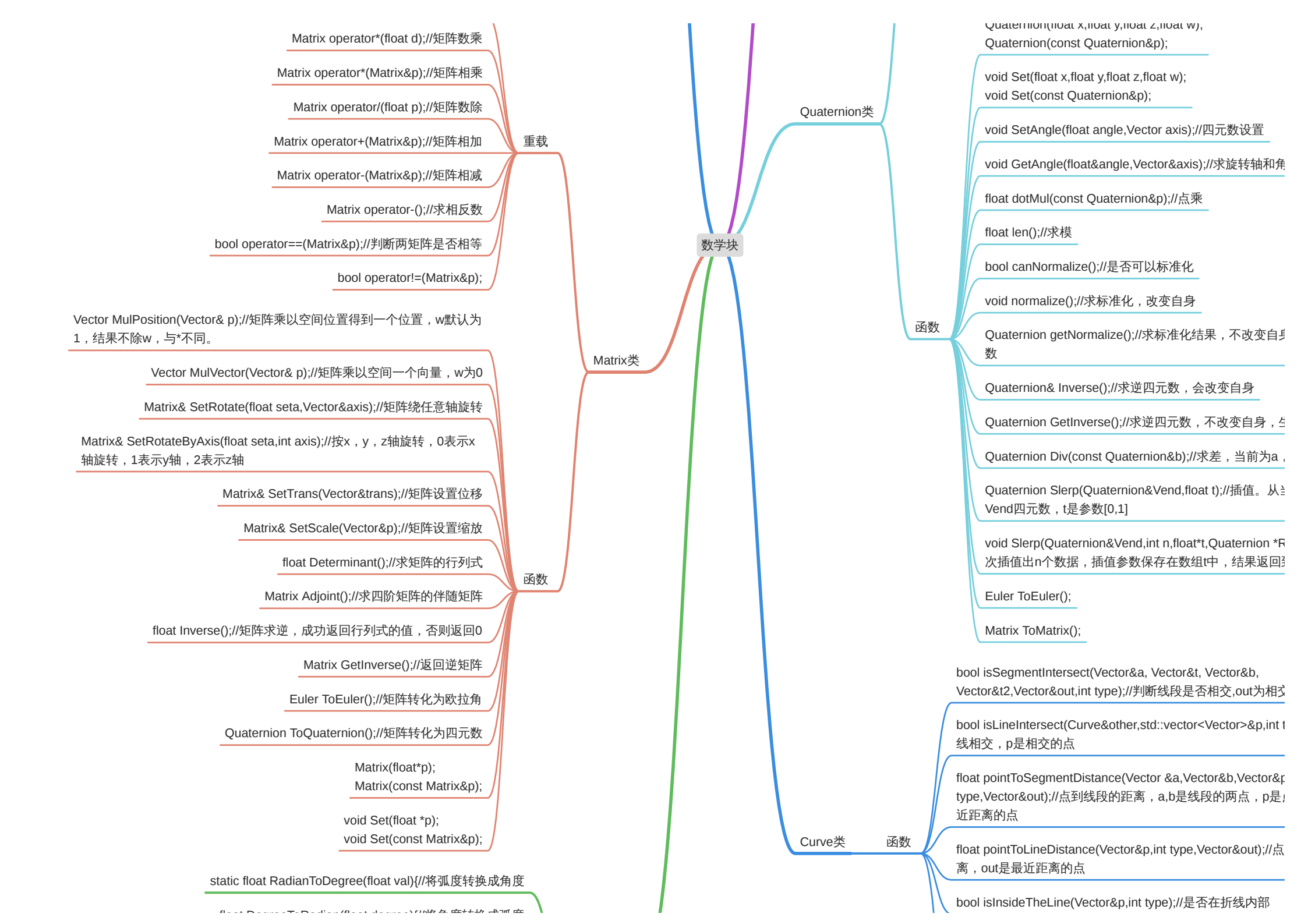
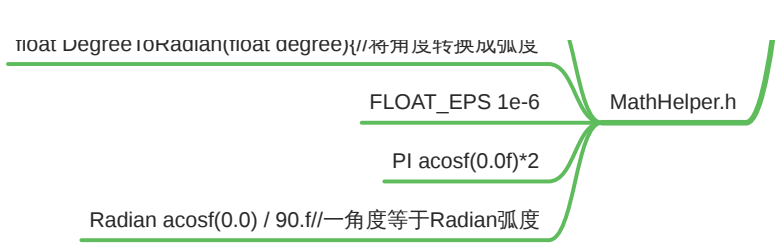

Vector operator+(Vector &p);
Vector operator-(Vector &p);//重载“-”操作
Vector operator*(float p);//重载“*”操作
Vector& operator=(Vector &p);//重载“=”操作
Vector operator-();//求相反数
bool operator==(Vector &p);//重载“==”操作
bool operator!=(Vector &p);//重载“!=”操作
重载
float dotMul(Vector &n);//向量点乘操作
Vector crossMul(Vector &n);//向量叉乘操作
float len();//向量求模操作
bool canNormalize();//是否可以标准化
Vector getNormalize();//获取向量标准化结果
Vector project(Vector &n);//向量投影操作
float findSeta(Vector&p);//求出两个向量之间的夹角
Vector rotateByAxis(float angle,int type);//type=0-绕x轴，type=1-绕y轴，type=2-z轴，比如绕z轴旋转(逆)angle=10°
Vector lerp(Vector&Vend,float t);//线性插值
void lerp(Vector&Vend,int n,float*t,Vector*Result);//线性插值
Euler ToEuler();//向量转欧拉角
Vector(float x,float y,float z);
Vector(const Vector &p);
void Set(float x,float y,float z);
void Set(const Vector&p);
函数
Vector类
Matrix& operator=(const Matrix&p);//矩阵赋值
void Set(float h,float p,float b);
void Set(const Euler&n);
Vector ToVector3(Vector*updir=0);//转换为向量
Matrix ToMatrix();//转换为矩阵
Quaternion ToQuaternion();//转换为四元数
void normalize();//单位化
Euler GetNormalize();//单位化
Euler(float h,float p,float b);
Euler(const Euler&n);
函数
Euler类
Quaternion& operator=(const Quaternion&p);//重载赋值
Quaternion operator+(const Quaternion&p);//重载‘+’
Quaternion operator*(float data);//重载数乘
Quaternion operator*(const Quaternion&p);//四元数乘法
Quaternion operator-(const Quaternion&p);//四元数求差
Quaternion operator-();//求相反数
重载


数学块
Matrix类
重载
Matrix operator*(float d);//矩阵数乘
Matrix operator*(Matrix&p);//矩阵相乘
Matrix operator/(float p);//矩阵数除
Matrix operator+(Matrix&p);//矩阵相加
Matrix operator-(Matrix&p);//矩阵相减
Matrix operator-();//求相反数
bool operator==(Matrix&p);//判断两矩阵是否相等
bool operator!=(Matrix&p);
函数
Vector MulPosition(Vector& p);//矩阵乘以空间位置得到一个位置，w默认为1，结果不除w，与*不同。
Vector MulVector(Vector& p);//矩阵乘以空间一个向量，w为0
Matrix& SetRotate(float seta,Vector&axis);//矩阵绕任意轴旋转
Matrix& SetRotateByAxis(float seta,int axis);//按x，y，z轴旋转，0表示x轴旋转，1表示y轴，2表示z轴
Matrix& SetTrans(Vector&trans);//矩阵设置位移
Matrix& SetScale(Vector&p);//矩阵设置缩放
float Determinant();//求矩阵的行列式
Matrix Adjoint();//求四阶矩阵的伴随矩阵
float Inverse();//矩阵求逆，成功返回行列式的值，否则返回0
Matrix GetInverse();//返回逆矩阵
Euler ToEuler();//矩阵转化为欧拉角
Quaternion ToQuaternion();//矩阵转化为四元数
Matrix(float*p);
Matrix(const Matrix&p);
void Set(float *p);
void Set(const Matrix&p);
static float RadianToDegree(float val){//将弧度转换成角度
Quaternion类
Quaternion(const Quaternion&p);
void Set(float x,float y,float z,float w);
void Set(const Quaternion&p);
void SetAngle(float angle,Vector axis);//四元数设置
void GetAngle(float&angle,Vector&axis);//求旋转轴和角
float dotMul(const Quaternion&p);//点乘
float len();//求模
bool canNormalize();//是否可以标准化
void normalize();//求标准化，改变自身
Quaternion getNormalize();//求标准化结果，不改变自身
数
Quaternion& Inverse();//求逆四元数，会改变自身
Quaternion GetInverse();//求逆四元数，不改变自身，
Quaternion Div(const Quaternion&b);//求差，当前为a，
Quaternion Slerp(Quaternion&Vend,float t);//插值。从
Vend四元数，t是参数[0,1]
void Slerp(Quaternion&Vend,int n,float*t,Quaternion *R
次插值出n个数据，插值参数保存在数组t中，结果返回
Euler ToEuler();
Matrix ToMatrix();
函数
Curve类
bool isSegmentIntersect(Vector&a, Vector&t, Vector&b,
Vector&t2,Vector&out,int type);//判断线段是否相交,out为相交
bool isLineIntersect(Curve&other,std::vector<Vector>&p,int t
线相交，p是相交的点
float pointToSegmentDistance(Vector &a,Vector&b,Vector&p
type,Vector&out);//点到线段的距离，a,b是线段的两点，p是
近距离的点
float pointToLineDistance(Vector&p,int type,Vector&out);//点
离，out是最近距离的点
bool isInsideTheLine(Vector&p,int type);//是否在折线内部



//沿折线运动，begind是运动前的索引位置,begdis是到运动前
距离float，movedis是要沿折线运动的距离，输出 pos(当前
curind(当前索引) curdis(到当前索引的距离)
void runByLine(int begind,float begins,float
movedis,Vector&pos,int&curin,float&curdis);