作业3和作业4：

说明：由于作业3和作业4的区别仅仅在于MyController.cpp里面对于world的设置稍有不同，而作业4用到了作业3的所有技术，故合写一篇文档。

作业3的原理：通过首先将旋转轴上的一点移动至原点，再经过两次旋转，使得旋转轴与y周重合，然后采用绕y轴旋转矩阵，最后实施前两步的逆变换将图形移回原处即可。

作业4的原理：将（已经经过旋转等变换的）三维物体投射到某一平面（xoy, yoz, zox）上，成为一个3维空间内的2维图形，再对于整个场景用轴二侧（或者其他用于显示3维场景的矩阵）变换映射到2维绘出（绘制3维场景的常规方式）。

关于作业4中的长方体消隐：

长方体只需要取消在视线方向上走得最远的一点即可，所以只要找出视线向量和顶点的坐标向量的点积（dot product）的值最大的那点（因为视向量的模不变，而电积的结果事实上就是该向量在视向量上的投影长度乘以视向量的模，故点积最大点就是视向量方向上的最远点），和那点有关的直线不画即可。

实现上的一些说明：

这两个作业的实现采用了一个自制的粗糙的3维框架。本框架是为了大作业做准备而做的一个粗糙的原形，但已经能使得作业3和作业4在MyController.cpp中的工作量减少不少：仅仅在于设置旋转变换矩阵和透视矩阵，以及向场景中添加物件，响应鼠标操作以及时钟到点操作变换场景中的物件。与该框架相关的文件在src3/Utilities下（src4下也有）。

下面对于这个框架作一个大致的介绍。

基础类：如Matrix4D等，支持了矩阵的基本操作， Primitive3D.h定义了用于构造其他3维图形的基础，此处只提供点和线段，不是十分合理，但对于此处已经足够（事实上，点可以用退化的线段表示，且本次作业中没有用到）

Painter3D：提供了绘制Primitive3D中元素的interface，可以有各种实现。此处给出了一个实现：ProjectPainter3D，需要用到一个Painter2D，将3D基础元素经过预先制定的投影矩阵变换（不要求第三维一定被映射为0，但只取前2维坐标）后，用该painter2D在2D画面中画出。（Painter2D是一个差不多的东西，此处略去）

Shape3D：一个interface，用以派生出各种3D物件。有两个方法：draw和transform，其中draw将传给该3D物件一个Painter3D使之可以primitives来表示本物体（比如本作业中采用线框图），另外还有一个WorldInfo参数提供外界的信息，比如视方向、外界的旋转矩阵等（主要用于自消隐）。

World：3D场景管理类，可以添加/删除Shape3D，本身具有一个transformer，意在当输出场景中的对象时对每个对象作该变换，具有absolute position的除外。有一个draw函数，传入一个Painter3D即可画出场景内的所有物件（其实就是foreach object: object.draw(); ）

有了上述基础设施，剩下的就是派生出一些Shape3D了。这两个作业中用到了以下几个：

SLine3D: （前缀S用于区别Primitive3D里的Line3D）3维线段

SurfaceHiddenCuboid: 消除了隐藏面的长方体。由于对于长方体自消隐有上述简易方法，故做了个消除隐藏面的长方体类（其实与其说是长方体，不如说只是满足某些拓扑结构的点列，因为其中没有做任何点的位置的检查）。消隐所需的视方向数据由draw函数的worldInfo提供。

Shadow: 由于作业4的需要，写了个Shadow，作为某一个Shape3D的投影（未必是2D投影，但此处仅仅是个2D投影）。这是对于Shape3D的一个比较有趣的应用。但由于有点偏离正轨，Shadow也有不少问题。Shadow的构造函数需要一个Shape3D作为影子的原像，一个投影矩阵，和一个视方向。这个视方向决定了被投影物体影子的消隐。这就暗示这个Shadow要么是具有绝对位置的（通过添加到world时指定），要么每次都需要外界修改这个视方向。这里由于xoy, yoz, zox平面都具有绝对位置，故采用前者。Shadow的原理很简单：draw的时候用一个自己做一个Painter3D的Decorator传和修改了视方向的WorldInfo给原像物件让它draw，这个decorator在画primitive时把primitive变换一下（根据投影矩阵）再传给原先那个painter3D，于是就有了投影的效果。但这个东西要transform就会比较麻烦，不太好定义，结果我的实现中不支持这个操作。

有了上面这些东西，最终MyController内的任务就是向world里add shape，然后做在适当的时候，做适当的变换。

当然，构造任意轴旋转矩阵还是一个值得一提的工作。对于同一个旋转轴不同的旋转角度，仅仅中间的绕y轴旋转矩阵不同，而两旁的矩阵不必重复构造，故作业3和4均作了缓存处理。

对于作业3，演示的时候最终没有采用长方体，由于感觉到对称物体旋转有点“看不清”，不过这个旋转的感觉也不是太好 :p 。

对于作业4，演示时的旋转轴是平行于y周过长方体中心的轴和平行于xoz平面主对角线的过长方体中心的轴。由这两根轴，已经可以演示出各种旋转后投影的效果，故没有采用3轴模式（这样还要进一步设计鼠标操作）。

最后，为了演示效果更佳，做了对拖动后放开鼠标时的速度的保留的自动旋转。但由此产生了画面闪烁的不足。