1 MotionBlur

1.1 总体流程

1.1.1 渲染前，保存当前帧的时间

```
// Remember the frame the scene was at so we can restore it later.
originalTime = ManimControl::currentTime();
```

1.1.2 渲染后，恢复最初的时间值

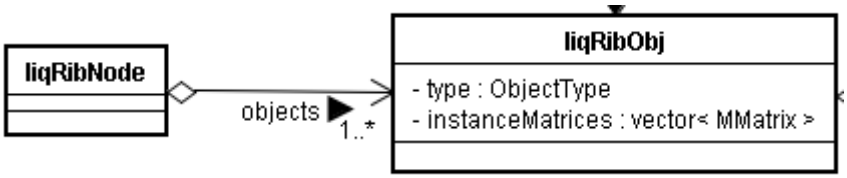
```
void liqRibTranslator::postActions(const MString& originalLayer__)
{
// return to the frame we were at before we ran the animation
MGlobal::viewFrame (originalTime);
...
}
```

1.1.3 每一帧

```
liqRibTranslator::processOneFrame(...)
{
        //  calculate sampling time
        calaculateSamplingTime(scanTime);//计算采样的时间点
        ...
        if( liqglo.doCameraMotion || liqglo__.liqglo_doMotion || liqglo__.liqglo_doDef )
        {
                for ( int msampleOn = 0; msampleOn < liqglo__.liqglo_motionSamples; msampleOn++ ) /*沿时间
        轴采样*/
                        scanScene__( liqglo__.liqglo_sampleTimes[ msampleOn ] , msampleOn );
                } else {
                        liqglo__.liqglo_sampleTimes[ 0 ] = scanTime;
                        liqglo__.liqglo_sampleTimesOffsets[ 0 ] = 0;
                        scanScene__( scanTime, 0 );
                }
        ....
        }
}


MStatus liqRibTranslator::scanScene__(float lframe, int sample )
{
        MTime mt( ( double )lframe, MTime::uiUnit() );
        if( MGlobal::viewFrame(mt) == MS::kSuccess )  //向前推进时间轴，以采样
        {...}
              ...
             scanSceneNodes(...);// 调用htable->insert(...);
}

liqRibTranslator::scanSceneNodes()
{
...
        if(  currentNode.hasFn( MFn::kNurbsSurface )
                || currentNode.hasFn( MFn::kMesh )
                || currentNode.hasFn( MFn::kParticle )
                || currentNode.hasFn( MFn::kLocator )
```

```
                || currentNode.hasFn( MFn::kSubdiv )
                || (currentNode.hasFn( MFn::kPfxHair ) && !currentNode.hasFn( MFn::kPfxGeometry ))
                || currentNode.hasFn( MFn::kPfxToon )
                || currentNode.hasFn( MFn::kImplicitSphere )
                || currentNode.hasFn( MFn::kPluginShape ) ) // include plugin shapes as placeholders
        {
                if( ( sample > 0 ) && isObjectMotionBlur( path ))
                        htable->insert( path, lframe, sample, MRT_Unknown, count++ );
                else
                        htable->insert( path, lframe, 0, MRT_Unknown, count++ );
        }
...
}

int liqRibHT::insert( MDagPath &path, double /*lframe*/, int sample,
                      ObjectType objType,
                      int CountID,
                      MMatrix *matrix,
                      const MString instanceStr,
                      int particleId )
{
  node->set( path, sample, objType, particleId );

}


void liqRibNode::set( const MDagPath &path, int sample, ObjectType objType, int particleId )
{
  // Create a new RIB object for the given path
  MObject obj( path.node() );
  liqRibObjPtr no( new liqRibObj( path, objType ) );
  LIQDEBUGPRINTF( "-> creating rib object for reference\n");
  no->ref();

  if( !objects[ sample ] ) {
    objects[ sample ] = no;//建立liqRibNode::object[i],对path的第i次采样的数据,记录在object[i]里
  } else {
    objects[ sample ]->unref();
    objects[ sample ] = no;//建立liqRibNode::object[i],对path的第i次采样的数据,记录在object[i]里
  }
}
```

1.2 Transform Motion Blur

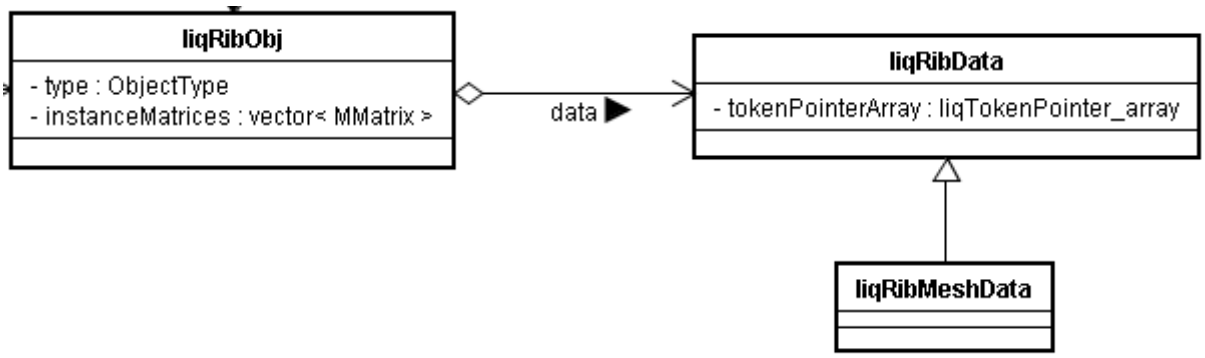



```
liqRibObj::liqRibObj( const MDagPath &path, ObjectType objType )
{
 MFnDagNode nodeFn( obj );

 // Store the matrices for all instances of this node at this time
 // so that they can be used to determine if this node's transformation
 // is animated.  This information is used for doing motion blur.
 MDagPathArray instanceArray;
 nodeFn.getAllPaths( instanceArray );
 unsigned last( instanceArray.length() );
```

```
  instanceMatrices.resize( last );
  for( unsigned i( 0 ); i < last; i++ )
    instanceMatrices[ i ] = instanceArray[ i ].inclusiveMatrix();
}
```

如何取transform motion的数据,见elvishray::Renderer::exportOneObject()

1.3 Deform Motion Blur




```
liqRibMeshData::liqRibMeshData( MObject mesh )
{
pointsPointerPair.set( "P", rPoint, numPoints );
normalsPointerPair.set( "N", rNormal,...);
pFaceVertexPointerPair.set( "st", rFloat, numFaceVertices, 2 );
pFaceVertexSPointer.set( "u", rFloat, numFaceVertices );
pFaceVertexTPointer.set( "v", rFloat, numFaceVertices );

  // Add tokens to array and clean up after
  tokenPointerArray.push_back( pointsPointerPair );
  tokenPointerArray.push_back( normalsPointerPair );
  tokenPointerArray.insert( tokenPointerArray.end(), UVSetsArray.begin(), UVSetsArray.end() );
  tokenPointerArray.push_back( pFaceVertexSPointer );
  tokenPointerArray.push_back( pFaceVertexTPointer );


  addAdditionalSurfaceParameters( mesh );
}


void liqRibData::addAdditionalSurfaceParameters( MObject node )
{
  // find the attributes
  MStringArray floatAttributesFound  = findAttributesByPrefix( "rmanF", nodeFn );
  MStringArray pointAttributesFound  = findAttributesByPrefix( "rmanP", nodeFn );
  MStringArray vectorAttributesFound = findAttributesByPrefix( "rmanV", nodeFn );
  MStringArray normalAttributesFound = findAttributesByPrefix( "rmanN", nodeFn );
  MStringArray colorAttributesFound  = findAttributesByPrefix( "rmanC", nodeFn );
  MStringArray stringAttributesFound = findAttributesByPrefix( "rmanS", nodeFn );

□ 于上面的每一个□ 量,先填入局部□ 量liqTokenPointer tokenPointerPair;然后
tokenPointerArray.push_back( tokenPointerPair );

}
```

如何取deform motion的数据,见elvishray::Renderer::exportOneGeometry_Mesh()