场景节点及动画

一 场景节点

在 Engine 的世界中，所有的内容都是由 Engine::Root 创建的几个 SceneManager 中的场景图。场景图（SceneGraph）为何物？就是一个由场景节点组织成的一个树形结构，每一个树形结构由一个根节点开始，根节点可以附加任意数量的孩子结点，循环往复。场景节点上可以挂上 MovableObject，这些 MoveableObject 可以是不可渲染的实体如摄像机（Camera），也可以是可渲染的实体如 Entity，ParticleSystem 等。

渲染整个场景的最简单方法就是沿着根节点开始，获取每个可渲染物，设置好变换矩阵等进行渲染即可。

因此，场景节点是整个场景图的核心部分。场景节点 SceneNode 派生自 Node，下面对他们进行分别介绍。

Node 类单独拿出来的目的是为了让它仅仅负责空间位置相关的操作，记录当前的节点位置，并可以进行任意的变换。记录一个节点的位置信息需要记录两组位置信息：一组是从世界坐标系变换信息，另一个是局部坐标系变换的信息。

世界坐标系变换信息会随着父亲节点的变动而变动。当本节点的位置信息发生移动时如果不通知其孩子结点，则其孩子结点无法知道父亲节点是否发生移动。这里的策略是:绘制循环中，显式调用_update 函数，如果自身发生变动则主动通知孩子结点更改继承而来的位置信息。

自身的变动始终是累加的，这种累加可以依据三种策略：TS_LOCAL. TS_PARENT, TS_WORLD。然后根据移动类型，又有不同策略，如下表-1 所示：

	TS_LOCAL	TS_PARENT	TS_WORLD
移动	局部移动+= 移动量*局部旋转	局部移动+=移动量	局部移动+=（移动量*父节点世界变换之旋转的逆方向）/父亲的全局缩放量
旋转	局部旋转= 局部旋转*旋转量	局部旋转= 旋转量*局部旋转	局部旋转=局部旋转*世界变换之旋转的逆旋转*旋转量*世界变换之旋转变换
缩放	局部缩放*=缩放量		

对其解释：

TS_WORLD 形式的 translate：在世界坐标系中进行移动，就应该按照原始的坐标轴进行移动，但是经过父亲节点的变换，全局坐标系已经进行了变换，如何恢复到原始的坐标系。只能乘以父节点变换的逆变换。Translate 只需要考虑方向，因此乘以的是父节点世界变换的逆旋转变换。

rotate 需要详细描述：

首先，旋转是用四元数来表示的，关于四元数的介绍可以参见相关文档。我的理解是用它表示旋转，可以利用乘法来表示两次连续的旋转，且插值也比较方便。如果仅仅记录三个角度的变换，无法描述上述的两次旋转合在一起的效果；矩阵的话，插值比较困难。

其次，两个四元数 a,b 相乘，a*b 表示 a 先起作用，b 后起作用。这个跟矩阵有用不同。所以，上面的 TS_LOCAL 是局部旋转*旋转量，表示局部变换先起作用，当前的旋转量后起作用；TS_PARENT 同样进行理解即可。

最后，TS_WORLD 形式需要好好理解：可以和 TS_LOCAL 进行对照，将“旋转量”变成“世界变换之旋转的逆旋转*旋转量*父节点世界变换之旋转变换”，前面的“世界变换

之旋转的逆旋转”目的是变回原始的世界坐标系方向，然后再进行当前的旋转变动，最后在将“世界变换之旋转变换”的作用加回去。

每个节点的当前移动信息通过下表所述来生成，这个在渲染节点时用到，可以见 Node::updateFromParentImpl 函数实现。

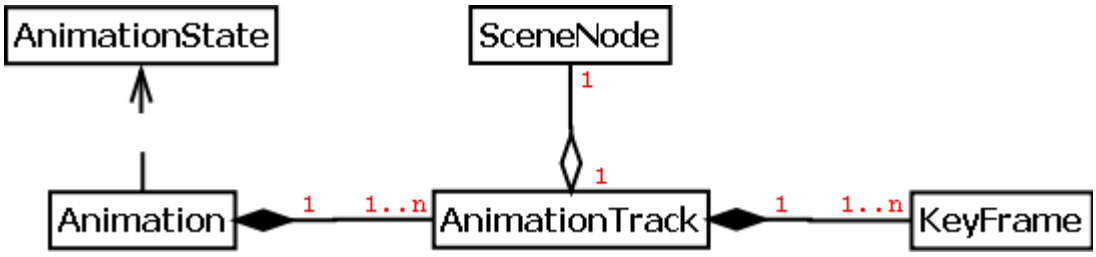
移动量=（父节点世界缩放量*当前节点局部移动量*父节点世界旋转）+ 父节点世界移动量
旋转量=父节点世界旋转量*当前节点局部旋转量
缩放量=父节点世界缩放量*当前节点局部缩放量

SceneNode 继承自 Node 类，因此它无需关心空间位置相关的信息。SceneNode 是用来附加 MoveableObject 用的，这些 MoveableObject 有些是可以渲染的，有些则不是。SceneNode 只需要处理好 attach 和 detach 这些 MoveableObject 就好。

二 场景节点动画

场景节点动画是另一个相对来说较为独立的模块，但是需要借助于场景节点才能实现。场景节点动画的基本思想是创建一系列的关键帧动画，用时间来对动画进行控制，绘制动画中的实体（人物，NPC 等）时，先要获取动画的时间，然后根据当前动画时间得到动画周围的两帧动画，并进行插值。插值的结果就是一个变换动作（偏移，旋转，缩放等），最后将该变换的动作实施到场景节点上即可生效。当上述过程在渲染的过程中不断进行并且保持一定的帧率（>20 fps）时，人眼基本上就基本上能够舒适得欣赏动画了。

因此，动画的数学模型可以总结为两个量：时间和关键帧。Engine 中的动画设计也是按照这个原则进行分工的。涉及的类如下图-1 所示：



时间控制部分是由 AnimationState 类来实现的；关键帧部分则是 Animation，AnimationTrack 和 KeyFrame 三个类来完成的。

AnimationState 类的实现由于只管理时间相关的部分，因此它的实现较为简单：记录动画的总长度以及当前时间，动画是否循环播放以及是否使能。当给这个动画增加时间的时候，它就会通过计算得到动画当前的时间位置。

Animation 类是多个 AnimationTrack 类的集合，AnimationTrack 类是当前动画中关于某 SceneNode 的动画信息，包含了多个 KeyFrame，这些 KeyFrame 是该 SceneNode 的轨迹。

如何使得这些动画生效呢？有上面的 AnimationState 可以得到当前动画的时间位置，该时间位置用来给后来的 Animation 来使用，Animation 中的每个 AnimationTrack 利用该时间找到两个 KeyFrame，使得当前动画时间位置在这两个 KeyFrame 的时间值中间。再根据三个时间位置对两个关键帧进行插值，得到节点的偏移后对节点进行偏移即可。

动画模块的一个关键问题在于插值算法的实现，目前我的程序中只提供了线性的插值方式，样条插值的方式还没有进行研究。线性插值的基本思想如下：

首先取得待插值的两个关键帧，设为 keyFrame1 和 keyFrame2，两关键帧的时间分别是 t1 和 t2。当前的时间是 t。于是得到插值结果 keyFrame：

$$keyFrame = keyFrame1 * (1 - \frac{t - t1}{t2 - t1}) + keyFrame2 * \frac{t - t1}{t2 - t1}$$

这种普通的插值对于 IM_LINEAR 方式的平移，旋转有效，对 RIM_LINEAR 方式也有效。当插值方式和旋转插值方式两者分别是 IM_LINEAR 和 RIM_SPHERICAL 时，旋转分量的插值采用的是球形插值的方式，这样可以保证插值的结果在角度之间是平滑的插值的。

四元数的球形插值的基本思想如下：

首先令两个四元数之间的角度是 θ，待插值的两个四元数分别是 q1 和 q2，最后的插值结果是 q，时间依然如上所述，于是有：

$$q = q1 * \sin(\theta * (1 - \frac{t - t1}{t2 - t1})) / \sin\theta + q2 * \sin(\theta * \frac{t - t1}{t2 - t1}) / \sin\theta$$

另外，需要保证一个节点当前只能参与一个节点动画，这个需要研究别人的代码如何实现。为何要作此限制呢？因为节点动画的实现是以动画开始之前的位置为基础的，如果同时执行两个动画，那么第一个动画做完后，第二个动画还是以第一个动画之前的位置为基础，从而导致第一个动画的效果没有呈现出来。最后的效果就是只有一个节点动画的效果会体现出来，因此我们需要这样进行限制。