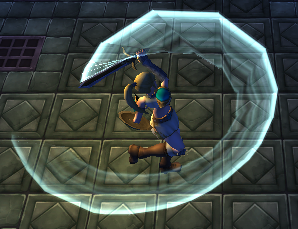
Sweeping Trail武器拖尾效果

(2018.8.1)

在一些格斗、动作游戏里面，例如Devil May Cry、战神、真三国无双，可能会有一些近战武器的拖尾效果，如下图







在网上随便参考了一下一些blog，感觉比较typical的实现就是生成一串连续的Quad，quad sequence的轨迹是沿着物体的运动轨迹的。（其实也可以只记下剑的多帧位置点，然后生成alpha blending是additive的粒子。但暂时(2018.7.19)还是考虑生成quad吧，考虑到近战武器的一般都是一长条的，所以我们设计的时候应该有下面的要点：

1. 记录下武器在多帧时候的**线段**
2. 这一堆线段感觉可以是一个链表/vector（都行）
3. **每两条线段最少可以生成两个三角形（不一定是quad），其实也可以考虑在四个点里面插插值生成更多顶点让拖尾的geometry更加平滑**。考虑到这个，其实三角形就可以在Geometry Shader里面生成了（给cbuffer传入一定长的数组，emmm shader端就不能动态改变线段数量了噢，这应该是要），而不是在CPU端update vertex/index buffer了吧。
4. **究竟是每个“quad“的uv就覆盖[0,0]-[1,1]，还是整个quad sequence的u坐标从头到尾渐变到0-1呢？(2018.7.19)**感觉是u坐标[0,1]覆盖整条拖尾比较舒服，这样子就可以customize整个拖尾的贴图了。

Idea：

1. 实际上整个trail sequence上的uv都是每帧动态变化的。而且头和尾的line segment都是要每帧移动的。
2. **头部的生成条件**：Head segment需要实时移动，然后可以隔一定时间（e.g 30ms）来“固定”下来，新增一个Quad。
3. Tail Segment也需要实时移动，注意最后一个quad的u坐标可能会有一部分小于零，所以比较好的方法是Tail Segment也实时移动以保证整个sequence的u都属于[0,1]
4. 这个weapon trail的每一个quad的生命周期其实都有点像粒子系统，然后就每一个quad上的顶点的u都在不断减少，所以u坐标其实就相当于是trail vertices的剩余生命了。理论上增加到1为止就算是生命周期结束了。（其实都行，有点像拉式描述，就是只关注某一个点上的值的变化，而不是欧式描述，就是关注某一个特定实体的移动）（这个场的定义域是不断变化的，头不断前进，尾不断缩回）
5. 每个新增的线段的顶点的u坐标干脆都设为0吧，等每帧更新他们加到1的时候，尾巴讲道理就要缩到这个位置了，那么他就可以退出历史的舞台了。
6. **尾部的销毁条件**：当尾部的quad的四个顶点的u都等于0的时候，quad\_count可以减一。
7. 一种极端情况：所有的quad从不褪去，即消退速度为0，消退时间无穷大，那么这种情况的最大顶点数要斟酌一下。万一active的quad数量大于vertex buffer的最大容量，怪罪策划了。
8. Vertex buffer的usage讲道理可以不需要Dynamic了，buffer的cpu xx flag应该为CPU\_ACCESS\_WRITE就好了，内存那里也维护一份vertices，然后用updateSubresource()来单向更新（其实讲道理Map/Unmap性能应该也不会差不多）都行吧，看心情
9. 要不就别用index buffer了，因为尾部缩过来之后，要删掉尾部quad，所有index都失效了，都要重新计算。如果只用vb的话，三角形其实可以是无序的，一个draw call无脑画就好了吧
10. 如果这个拖尾有一个free header,一个free tail，那得起码有3条line segment才可以开始正常工作吧？于是平时每帧更新的时候只改变free header&tail 一共四个顶点。（咦好像不是喔，每一帧都要更新所有顶点的uv，所以其实每一帧的更新都是在所难免的咯？）
11. 原本在想要不要一直保持不管多少个quad，头的u就是0，尾就是1。但是这样子的容易在拖尾没到拉到最大长度的时候，整个贴图会被强行拉长。而且这样子u坐标的计算其实也挺麻烦的。如果我们引入一小个概念，就是像粒子系统一样的**“生命周期”**，那么原来有个奇怪的概念“TailQuadCollapseTime”就可以去掉了，因为如果每个线段都有一定的生命周期，那么**尾部quad的坍缩时间就等于线段的冷却时间**。

未来可能的优化和特性：

1. 边缘vertices轨迹的平滑和插值（√，2018.7.31）
2. 基于线段扫掠的基础上推广，做一个2D图形的扫掠（还挺硬核的）

(2018.7.31)现在已经实现了网格生成/坍缩和贴图；顶点切线估计和用Cubic Hermite做平滑的样条插值。

