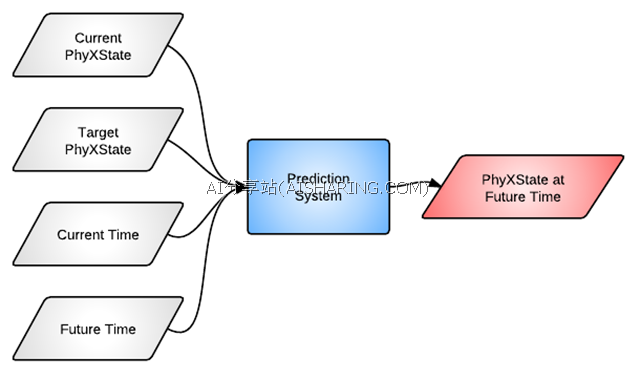
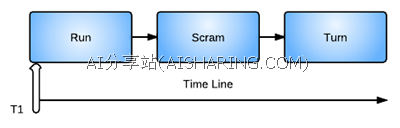
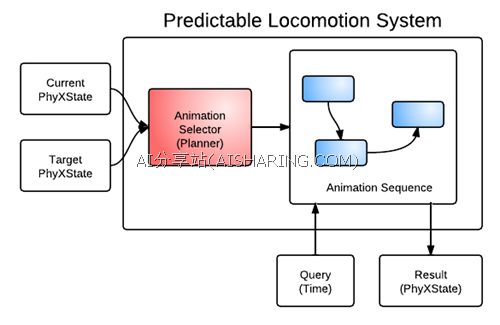
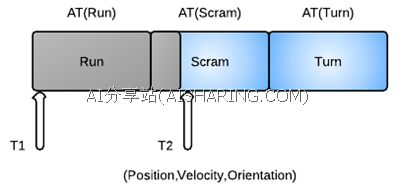
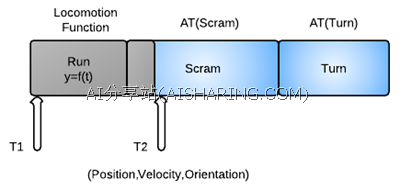
**可预测的运动系统设计**  
  
　　前端时间举家出游了一次，加上国庆期间一直跑东跑西，博客的更新就一直没跟上，距上一篇文章也是好久了呢，有时感觉一个人维护也有点小累，所以如果大家有好的想法，好的分享，也可以投稿给我，我想能有这样一个分享的平台，让志同道合的朋友一起讨论学习，也算是为中国游戏技术的发展贡献点绵薄之力，虽然我是付不起稿费的，但看着文章的浏览数和回复也是挺有成就感的事情，对吧:)。  
  
　　好，言归正传，这次想大家来讨论讨论如何来做一个“可预测的运动系统”。首先来说说什么是游戏里的运动系统。  
  
　　对于游戏里某一个智能体（就是带AI的物体），当AI决策结束，并且告诉行为层要干什么，行为层就会让它“动”起来，动的过程中可能是没有动画的，比如一辆车会前进，会后退，也可能是有动画的，比如一个人会走，会跳。显然一辆车的运动方式和一个人的运动方式是不同，那具体控制它如何运动，就是有专门的“运动系统”来负责了。  
  
　　严格意义上来说，“运动系统”的实现并不在AI的范畴内，而是属于“游戏物理引擎”部分，但在某些时候，作为AI程序员，会对运动系统提出一些特殊的要求，比如今天我们会说到的“可预测性”，这些要求会用来来辅助AI系统的决策。在我现在的项目中，我们对预测的要求相当高，希望在决策的时候能做一个精确的运动预测，但比较遗憾的是，现有的引擎由于这样那样的原因，并不能提供这样一个精准预测，预测结果和运行结果存在较大的差距，所以我在抱怨的同时（因为我用的最多嘛），也在思考如何做这样一个系统，把想到的一些东西也记录在这里。  
  
　　一个比较简单的“可预测的运动系统”就是匀速直线运动，我们可以根据匀速直线运动的公式，来预测将来任意时刻的位置，速度等运动信息。当然，游戏里的运动系统要复杂的多。一般来说，游戏里的运动可以大致分为两种：  
  
·动画配合运动   
·运动配合动画   
  
　　第一种就是运动和动画是分离的，我可以用任何的运动函数来移动物体，动画只是配上去的效果而已，这种方式有动画和没有动画，对于它的运动效果来说是不变的，动画只是一种锦上添花的表现，因为运动方式都是自己定义的，所以我们可以很容易控制它运动效果，但缺点是，可能会产生“滑步”的现象，就是人感觉在太空漫步一样，可以想象我们用跑步的速度配一个走路动画的效果。  
  
　　第二种是物体的运动效果是跟着动画的，如果动画中移动了1米，那我在播放这个动画的时候，这个物体也就移动了1米，这种方式就不存在一个所谓的“运动函数”的概念，因为都是动画数据驱动的，所以运动系统基本上就是从动画中取得当前物体的位置，这样的好处就是避免了“滑步”的问题，但是对于程序员来说，它不可控，完全取决于数据。  
  
　　所以，当我们想要做一个“可预测的运动系统”的时候，就不得不考虑上面的两种情况，因为在不同的游戏中，运动系统的实现是不同的，有的用了第一种，有的用了第二种，有的是在第一和第二中之间切换的。  
  
　　为了更好的表述“可预测的运动系统”，我需要定义一个运动状态的结构来描述当前物体的运动状态：  
  
  
  
　　这个结构里包括位置（3维向量），速度（3维向量）和朝向（2维向量），这是描述物体运动状态的三个基本量，对于通用的“可预测的运动系统”来说，它的输入输出就可以这样来描述：  
  
  
  
　　输入是当前状态（Current PhyXState），目标状态（Target PhyXState），当前时间（Current Time）和想要预测的未来时间（Future Time），输出就是在未来时间的状态（PhyXState at Future Time）。特别要指出的是，如果预测的运动状态是基于当前的运动行为，那我们可以不传入“目标状态”，“当前状态”和“当前时间”，仅仅传入“未来时间”即可，因为对于当前运动行为而言，其内部已经保存了“目标状态”，“当前状态”和“当前时间”了。但作为通用描述，还是将这三项列在其中。  
  
　　我们仔细考虑的话，会发现，其实运动系统天生是带有些许预测功能的，当每一帧在更新的时候，就是向运动系统传入了这些参数，然后得出了当前运动物体应该处于的运动状态，在这些参数中唯一值得注意的是“未来时间”，在正常的游戏循环中，“未来时间”是被指定为：  
  
http://upload.gameres.com/201111/sf_3171_383.jpg  
  
　　也许你会觉得问题似乎是解决了，因为这样的话，不是就可以精确预测任意时候的运动状态了吗？  
  
　　但回想一下我上面所说的游戏里用的两种运动方式，那问题就并不是这么简单了，对于第一种运动系统，确实不怎么修改就可以实现精确的预测，因为这种运动系统一般是基于数学函数，或者经验数据表（就是预先算出不同输入下的运动数据值，然后通过查表的方式来返回结果），要做的仅仅是封装一个接口，但对于第二种，或者对于混用的运动系统，我们就需要考虑更多的问题了，……。  
  
　　游戏中的运动系统一般有两种方式，“动画配合运动”以及“运动配合动画”。对于第一种方式，由于是采用运动函数或者经验数据表，所以可以很简单的将“未来时间”带入其中，来预测未来某一时刻的运动结果。但对于第二种情况，因为所有的运动结果都是从动画中取得的，如果不知道动画信息，就无法知道相应的运动结果，所以，简单的将时间带入是不能作出预测的，对于这样的情况，如果我们要做预测的话，就不得不将动画的因素考虑在里面。  
  
　　举个简单的例子，一个人做一个跑动中转身停下的行为，假设他当前时刻T1，处于A1位置，速度是V1，朝向是F1（假设和速度方向一致），他的目标状态是速度是0（没有速度），朝向是F2，位置不指定。这时当我们给出未来时间T2，应该怎样来做这个运动预测呢？  
  
  
  
　　和第一种运动系统不同的是，这里我们需要一个额外的模块，称之为“动画选择模块”。在我们实际要去完成这个运动目标的时候，我们会用“动画选择模块”去选出每一步的动画，比如最直观就是“直线跑”，然后会产生一个“急停”使速度从V1到0，最后是“原地转身”使朝向从F1到F2，当这些动画做完后，我们就可以得到这个人最终的“目标位置”。由于是运动配合动画的，所以对于他的最终位置，我们完全是靠这些动画实际的运行结果而得到的。  
  
  
  
　　当我们要做预测的时候，显然我们也是需要用“动画选择模块”做预先的动画预测的，所以，在设计可预测的运动系统的时候，就需要将“动画选择模块”独立出来。如下图所示：  
  
  
  
　　动画选择模块的输入是“当前状态”和“目标状态”，输出是一个动画的序列，这就有点像以前我们讨论过的“计划器”（Planner），相当于为我们的运动做了一个计划（对于这样的计划器的实现，就不在这次的讨论中了）。有了这样的计划后，当我们将时间T2带入，就可以通过读取动画信息的方式来获得T2时刻的运动信息了，见下图，AT指动画的时间（Animation Duration）  
  
  
  
　　对于给定的“当前状态”和“目标状态”，都会有一组动画序列与之对应，所以如果是基于目前的运动状态所作的预测，那我们可以重用我们已有的动画序列来提高效率。如果是对于假想的运动状态的预测，那我们就需要用“动画选择模块”重新做一个新的动画序列，然后再得出未来时间的运动状态，这也是为什么我们需要独立的“动画选择模块”的原因，但如果频繁做较长步骤的假想预测，可能会产生一些性能上的问题，这是需要注意的地方，当然，这也取决于“动画选择模块”实现的复杂度和动画的丰富程度，比如在上个例子中，如果我们有一个“急停转身”的动画，那我们就可以减少动画序列的个数，也就是减少了计划的[来源：GameRes.com]步长了。  
  
　　另外，在实践中，我们会采用两种运动方式混用的情况，比如对于“直线跑”，我们会用运动函数来实现，而对于“转身”，“急停”这样的行为，我们会采用第二种方式来实现，对于这样的运动系统，也可以用到这样的“计划器”，只是这个计划中的某一步换用函数方式罢了，对于上面例子，我们可以参考下图：  
  
  
  
　　除了预测未来时间的运动状态外，可能我们还会预测到达某一个状态所要用到的时间，当我们有了上面的系统后，这也会非常容易做到，比如上面的例子，如果我们要预测他到“目标状态”需要多少时间，那我们只需要把动画序列中每一个动画的时间求和就可以了，T = T1 + AT(Run) + AT(Scram) + AT(Turn)。  
  
　　可预测的运动系统对于某些游戏可能是一个非常重要的系统，希望上面的讨论对大家有所帮助。  
  
作者：Finney   
Blog：[AI分享站](http://www.aisharing.com)  
Email：[finneytang@gmail.com](mailto:finneytang@gmail.com)

http://bbs.gameres.com/showthread.asp?threadid=175629