具有自身平衡系统的虚拟生物在三维空间内捕食与逃逸的关键技术研究

班晓娟 陈 希 宁淑荣
(北京科技大学信息工程学院 北京 100083)

摘 要 以虚拟鱼为例,提出了虚拟生物在纯三维环境中捕食与逃逸的关键技术点的算法,即在保持自身平衡的前提下,自由游弋探查周围食物和在锁定敌人或食物后进行行为预测及调整自身方向等过程中的核心算法,并以程序运行结果证实了此一系列算法的有效性。

关键词 虚拟生物,人工智能,自身平衡,捕食,逃逸
中图法分类号 TP311 **文献标识码** A

Research on Hunting and Escape of Virtual Life with Self-balancing System in 3 Dimensional Space

BAN Xiao-juan CHEN Xi NING Shu-rong
(College of Information Engineering, University of Science and Technology Beijing, Beijing 100083, China)

Abstract Taking virtual fish as an example, the paper presented an algorithm with critical technical points on virtual lives' food hunting behavior and escaping in three-dimensional environment. The core algorithm were researched which can search automatically, detect food, predict self-behavior and adjust searching direction after an enemy or food is locked down with the precondition of maintaining self-balance. Results of simulation were presented to prove the effectiveness of the series of algorithms.

Keywords Virtual life, Artificial intelligence, Self-balance, Hunting, Escaping

随着虚拟现实技术及人工智能的不断发展,3D技术已不仅运用于动画和简单3D游戏,而是更多地加入了人工智能的成分,如智能动画、虚拟智能生物的演示、游戏中高AI的游戏角色等。Reynolds的鸟群动画"Boid"[1]和涂晓媛的人工鱼动画"xiaoyuan's fish"[2]是此类技术的典型代表。

对于一个虚拟的智能生物来说,"移动"、"捕食"、"避敌"可以说是比较基本的几种行为,故在三维空间内完成这几种动作模块设计也就成为设计智能虚拟生物过程中一项必不可少的工作。而在有关"xiaoyuan's fish"的学术论文中并没有深入提及这方面内容的具体细节。随后出现的一系列关于虚拟生物的学术论文,如Kingsley Stephens的Modelling Fish Behaviour[3]、权晓林的基于人工生命方法的虚拟鱼行为模型[4],其中着重介绍的是虚拟生物整个行为模型的构建,对其具体的实现方法并没有深入剖析。而许多3D游戏或其他虚拟现实作品中的智能生物虽然采用的是三维模型,但其移动都是在二维平面上的;有些将三维空间中的y轴离散化,即将活动空间定义在几个平面上,角色通过在不同的y轴坐标平面上运动来实现三维空间中的运动。

本文主要从技术实现的角度,以虚拟鱼为例,研究其在纯三维空间内的捕食、逃逸过程中维持自身平衡、自由游弋、探察目标物体、调整自身方向及预测等技术要点算法,并在此过程中对比指出了一些理论化方法的不足之处。本文首先介绍捕食、逃逸模块的整体逻辑设计;然后介绍虚拟鱼完成所有行为的一个重要前提,即"维持自身平衡"的方法;接着介绍虚拟鱼自由游动、对食物与敌人的探查、方向调整等行为算法;随后介绍预测机制作为调整算法的改进;最后对前面一系列算法给出验证截图及总结。

1 虚拟鱼捕食与逃逸功能模块总体设计

本文以虚拟鱼作为研究对象,使用MAYA8建立鱼类模型,然后进行鱼类表皮的蒙贴,最后加入鱼类的骨骼,做出游动效果,如图1所示。

图1 骨骼鱼动画

建立模型的时候,将鱼头方向设为z轴正向,将鱼的脊背方向和左侧设为y轴和x轴。然后使用以C++为基础语言的3D引擎进行模型的调用。

图2为虚拟鱼捕食的逻辑图,主要算法集中在5个环节上:

1)维持自身平衡;

2)三维空间内的自由游动;

到稿日期:2008-10-21 返修日期:2009-01-22 本文受国家自然科学基金项目(11250018)资助。
班晓娟(1970-),副教授,主要研究方向为人工智能、人工生命、计算机动画等;陈 希(1983-),研究生,主要研究方向为人工智能虚拟现实,E-mail:matrix19830721@163.com;宁淑荣(1975-),讲师,主要研究方向为人工智能、人工生命。

3)探查食物或敌人；

4)调整方向；

5)预测。

碰撞检测的技术比较成熟，在一般的 3D 引擎中已有比较成型的算法，本文不做过多讨论。

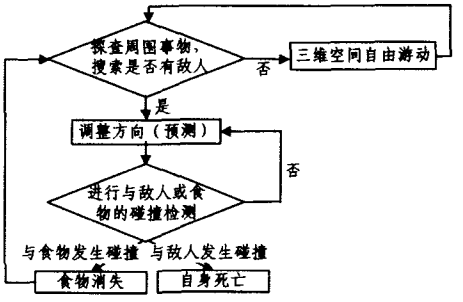

图 2　虚拟鱼捕食的逻辑图

2　维持虚拟智能体自身的平衡

有些物体在三维空间内移动的时候是需要维持自身平衡的。比如一条鱼在水中游动捕食的时候，其运动和一个质点或椎形物体的运动是有区别的，如图 3 所示。这主要是由于鱼具有自身的一套 xyz 坐标系统，鱼在移动的时候不仅仅要维持自身的 z 轴朝向目标方向，同时要使自身的 z 轴与 y 轴所组成的平面与世界坐标系 x 轴与 z 轴组成平面垂直，而且自身 y 轴的方向与世界坐标系 y 轴的夹角在±90°之间，即

$$(ZOY_{fish} \perp ZOX_{world}) \cap (\alpha(\overrightarrow{OZ}_{fish}, \overrightarrow{OZ}_{world}) < 90°) \quad (1)$$

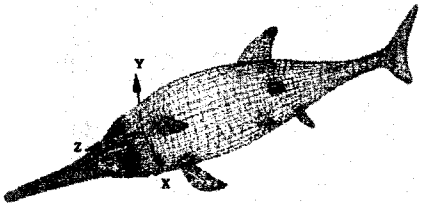

图 3　鱼类的建模图

3　三维空间内的自由游动

虚拟鱼在三维空间内的自由游动可分为两部分：一部分是绕 Y 轴的转动，另一部分是绕 X 轴的转动。通俗地讲，一部分控制上下摆头，另一部分控制左右摆头。然后只需要将两个随机的角度传递给这两部分，虚拟鱼就可以在空间内自由游动了。

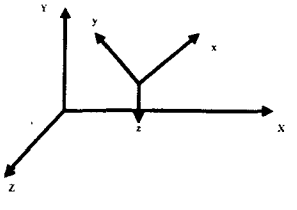

图 4　世界坐标系与局部坐标系

这看似是个理论上很可行的方法，但实际上存在问题，因为这其中关系到坐标系的选择和虚拟鱼平衡的保持问题。众所周知，在三维世界中有两种坐标系(如图 4 所示)：一种是不随物体移动而改变的世界坐标系(XYZ)，另一种是随物体移动而改变的局部坐标系(xyz)。

如果在转动的时候只选择一种坐标系，是根本无法完成预期的自由转动的效果的，虚拟鱼会在游动中失去平衡(如图 5 所示)，即不能满足前面提到的平衡条件。

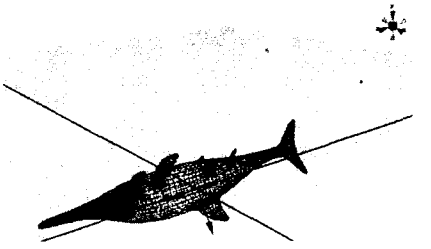

图 5　虚拟鱼失去平衡

如果综合使用两个坐标系，就可以较好地完成鱼的三维空间内的游动动作。使用世界坐标系的 Y 轴作为虚拟鱼左右摆动的轴心，而使用以虚拟鱼自身为参照物的局部坐标系的 z 轴为上下摆头的轴心。这样，只要在初始化虚拟鱼的时候将虚拟鱼的 y 轴方向与世界坐标系的 Y 轴方向一致，即 $\overrightarrow{OY} // \overrightarrow{oy}$，就可以基本保证虚拟鱼的平衡。此外还应注意的一点是上下和左右摆动角度的范围。左右摆动角度 ϕ_{OY}(绕世界坐标系 Y 轴的旋转角)基本上是没有限制的，即 $-180 \leqslant \phi_{OY} \leqslant 180$；而上下摆动角度(绕局部坐标系 z 轴的旋转角)就有一定范围的限制。一般随机出现的角度 ϕ_{oz} 要维持在±90°，使 $\alpha(\overrightarrow{OZ}_{fish}, \overrightarrow{OZ}_{world}) < 90°$，以保证虚拟不会出现虚拟鱼“翻个”游动的情况。

4　对食物或敌人的探察

模仿动物感知系统的局限性十分重要[5]。虚拟智能生物(如虚拟鱼)一般会设置一个感知范围。很多情况下会将感知范围设置为一个正圆形，每一次探测都会计算一下虚拟场景内的敌人或食物相对于自己的距离，即

$$dis_{target} = \sqrt{(X_{tar} - X_{fish})^2 + (Y_{tar} - Y_{fish})^2 + (Z_{tar} - Z_{fish})^2} \quad (2)$$

其中，下标为 tar 的坐标均为目标体坐标，下标为 $fish$ 的坐标为虚拟鱼坐标。

然后将 dis_{target} 与鱼自身的感知距离 dis 进行比较，决定事物是否进入了鱼的感知范围。

这无疑是一种比较理想的情况，因为对于很多智能生物体来说其感知范围并不是一个圆。大多数情况下，虚拟生物正对的方向，也就是其 z 轴正向的感知范围会大一些，其他方向上的感知范围会小一些，特别是在背后(z 轴负向)。这就会为计算其感知范围带来不小的困难。

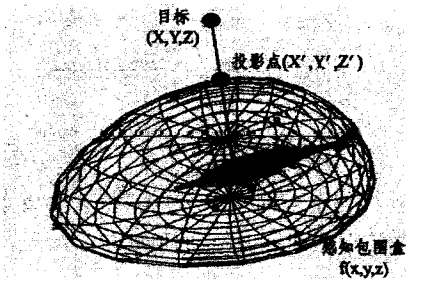

图 6　虚拟鱼感知包围盒

在计算感知范围的时候，使用以虚拟鱼为参照物的局部

坐标系会给计算带来许多方便，如图 6 所示。现在以虚拟鱼为中心点，已知目标物体相对于虚拟鱼的坐标点(X,Y,Z)及虚拟鱼的感知范围函数$f(x,y,z)=0$，只要求出目标在感知包围盒上的投影，然后比较投影距离与目标距离的大小，就可以知道目标是否在感知范围内。

列出方程组：

$$\begin{cases} f(X',Y',Z')=0, & (X',Y',Z')\text{在包围盒上} \\ \dfrac{X}{X'}=\dfrac{Y}{Y'}, & \text{目标点、投影点、中心点在同一直线上} \\ \dfrac{X}{X'}=\dfrac{Z}{Z'}, & \\ \dfrac{X}{X'}>0,\dfrac{Y}{Y'}>0,\dfrac{Z}{Z'}>0, & \text{目标点与投影点的坐标必然同号} \end{cases} \tag{3}$$

通过解这个方程组得出唯一坐标点，再通过距离的比较就可以得出目标点是否在虚拟鱼的感知范围之内。

5 方向的调整

在锁定目标后，虚拟鱼要做出相应的调整方向的动作，即进行头部的转动，与此同时还要保持自身的平衡。下面将以虚拟鱼的捕食为例，说明虚拟鱼方向调整方法。

现在已知虚拟鱼的位置$(X_{fish},Y_{fish},Z_{fish})$、$z$ 轴方向$(x_{fish},y_{fish},z_{fish})$和食物的坐标点$(X_{food},Y_{food},Z_{food})$，做一下简单的坐标变换，将世界坐标系的原点移到鱼中心坐标原点上，这时虚拟鱼的坐标中心将会变为$(0,0,0)$，z 轴方向不变，而食物坐标会变为$(X'_{food},Y'_{food},Z'_{food})=(X_{food}-X_{fish},Y_{food}-Y_{fish},Z_{food}-Z_{fish})$，如图 7 所示。

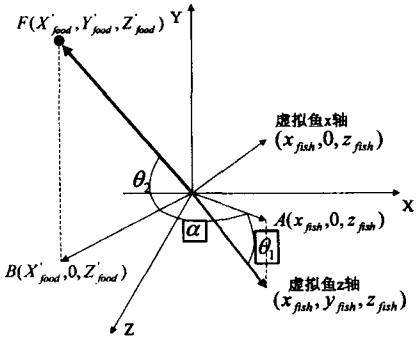

图 7　虚拟鱼方向调整示意图

虚拟鱼转角时需要(α,β)两个角度，α 为绕世界坐标系 Y 轴的转角，β 为围绕虚拟鱼自身 x 轴的转角。虚拟鱼 z 轴和食物在 XOZ 平面上会有两个投影点，分别为 $A(x_{fish},0,z_{fish})$，$B(X'_{food},0,Z'_{food})$，α 就是$\overrightarrow{OA}$与$\overrightarrow{OB}$的夹角，且

$$\cos\alpha=\frac{X'_{food}*x_{fish}+Z'_{food}*z_{fish}}{\sqrt{X'^{2}_{food}+Z'^{2}_{food}}*\sqrt{x^{2}_{fish}+z^{2}_{fish}}} \tag{4}$$

$$\alpha=\arccos\left(\frac{X'_{food}*x_{fish}+Z'_{food}*z_{fish}}{\sqrt{X'^{2}_{food}+Z'^{2}_{food}}*\sqrt{x^{2}_{fish}+z^{2}_{fish}}}\right) \tag{5}$$

不要认为有了 α 角就可以知道虚拟鱼需要绕世界坐标系的 Y 轴怎样旋转了。$\arccos(x)$的值域为$[0,\pi]$，这个值是没有方向的，而虚拟鱼的水平旋转角度为$[-\pi,\pi]$。换句话说，只有一个角度是不够的，因为虚拟鱼不知道是向左转还是向右转。这时就需要用虚拟鱼的 x 轴进行辅助判断，求出虚拟鱼 x 轴与$\overrightarrow{OB}$的夹角 ϕ：

$$\cos\phi=\frac{X_{food}'*x_{xfish}+Z_{food}'*z_{xfish}}{\sqrt{X'^{2}_{food}+Z'^{2}_{food}}*\sqrt{x^{2}_{xfish}+z^{2}_{xfish}}} \tag{6}$$

当 $\cos\phi\geqslant 0$ 时，食物在虚拟鱼的左侧，以 α 作为水平转角；

当 $\cos\phi\leqslant 0$ 时，食物在虚拟鱼的右侧，以 $-\alpha$ 作为水平转角。

下一步要计算虚拟鱼绕自身 x 轴的转角 β。β 可以分解为两部分进行计算，一部分是虚拟鱼的 z 轴与其在 xoz 平面投影的夹角 θ_1，另一部分是食物点向量$\overrightarrow{OF}$与其在 xoz 平面上的投影向量$\overrightarrow{OB}$的夹角 θ_2。

$$\cos\theta_1=\frac{|\overrightarrow{Oz}|}{|\overrightarrow{OA}|} \tag{7}$$

$$\cos\theta_1=\frac{\sqrt{x^{2}_{fish}+z^{2}_{fish}}}{\sqrt{x^{2}_{fish}+y^{2}_{fish}+z^{2}_{fish}}} \tag{8}$$

$$\theta_1=\arccos\left(\frac{\sqrt{x^{2}_{fish}+z^{2}_{fish}}}{\sqrt{x^{2}_{fish}+y^{2}_{fish}+z^{2}_{fish}}}\right) \tag{9}$$

$$\cos\theta_2=\frac{|\overrightarrow{OF}|}{|\overrightarrow{OB}|} \tag{10}$$

$$\cos\theta_2=\frac{\sqrt{X'^{2}_{food}+Z'^{2}_{food}}}{\sqrt{X'^{2}_{food}+Y'^{2}_{food}+Z'^{2}_{food}}} \tag{11}$$

$$\theta_2=\arccos\left(\frac{\sqrt{X'^{2}_{food}+Z'^{2}_{food}}}{\sqrt{X'^{2}_{food}+Y'^{2}_{food}+Z'^{2}_{food}}}\right) \tag{12}$$

得出 θ_1 与 θ_2 后，这两个角同样不具有方向感，因此简单地将这两个角相加或相减都是盲目的。正确的做法是首先通过 z_{fish} 与 Z'^{2}_{food} 的关系来确定 $\beta=|\theta_1+\theta_2|$ 还是 $\beta=|\theta_1-\theta_2|$。

当$\frac{z_{fish}}{Z'^{2}_{food}}\geqslant 0$ 时，可认为食物和鱼头均在 xoz 平面的同侧，$\beta=|\theta_1-\theta_2|$；当$\frac{z_{fish}}{Z'^{2}_{food}}\leqslant 0$ 时，可认为食物和鱼头均在 xoz 平面的异侧，$\beta=|\theta_1+\theta_2|$。

得出 β 后，借助虚拟鱼 y 轴的向量坐标$(x_{yfish},y_{yfish},z_{yfish})$求出虚拟鱼 x 轴与$\overrightarrow{OF}$的夹角 γ。

$$\cos\gamma=\frac{X'_{food}*x_{yfish}+Y'_{food}*y_{yfish}+Z'_{food}*z_{yfish}}{\sqrt{X'^{2}_{food}+Y'^{2}_{food}+Z'^{2}_{food}}*\sqrt{x^{2}_{yfish}+y^{2}_{yfish}+z^{2}_{yfish}}} \tag{13}$$

当 $\cos\gamma\geqslant 0$ 时，则食物在虚拟鱼的上侧，以 γ 作为垂直转角；

当 $\cos\gamma\leqslant 0$ 时，则食物在虚拟鱼的下侧，以 $-\gamma$ 作为垂直转角。

经过以上多步计算处理，水平和垂直方向的两个转角即可得出，然后调用 3D 引擎旋转模型即可得到虚拟鱼在发现食物后，调整方向，游向食物的效果。

虚拟鱼避敌的处理，实际上是捕食处理的一种改进。如图 8 所示，虚拟鱼在感知到敌人后的逃逸方向，就是虚拟鱼与敌人所构成向量的反方向。

将危险物置于虚拟鱼与其连线上，以虚拟鱼为中心点，在另一侧生成一对称点，作为一个假食物点。虚拟鱼的逃跑方向即是虚拟鱼游向假食物点的方向，这样就可以将逃逸转换成捕食来处理。

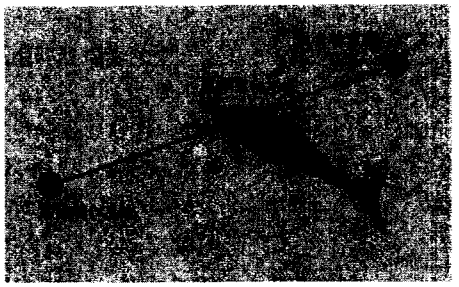

图 8　假食物点生成图

假食物点坐标的计算公式为

$$\begin{cases}\dfrac{X+X'}{2}=x\\ \dfrac{Y+Y'}{2}=y\\ \dfrac{Z+Z'}{2}=z\end{cases}\quad(14)\Rightarrow\begin{cases}X'=2x-X\\ Y'=2y-Y\\ Z'=2z-Z\end{cases}\qquad(15)$$

6　预测机制

无论是食物还是敌人都不可能全是静止的物体。预测机制是一个智能虚拟生物智能性的集中体现。好的预测机制可以使虚拟鱼更快地捕捉到食物，而且可以有效躲避敌人。Kingsley Stephens 在其 Modelling Fish Behaviour 论文中给出了一种预测方法，如图 9 所示。

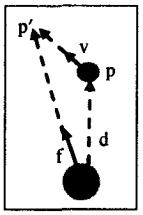

图 9　Kingsley Stephens 预测方法示意图

如图 9 所示，Kingsley Stephens 通过已知食物的移动速度、与食物的距离及自身的速度计算出一个相遇点，然后让虚拟鱼直接游向相遇点。这种预判方法是一种很理想化的方法，真正实现起来会存在一个问题，就是食物的变速情况。无论是食物速度方向的改变还是大小的变化，都会使前面的大量预测计算作废，同时增加系统的开销。比较切合实际的做法就是仅预测下一步目标的动向。

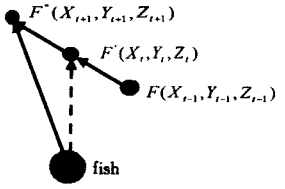

图 10　预测点生成示意图

如图 10 所示，蓝色点为虚拟鱼的位置。当虚拟鱼在最初感知到目标时，就会记录下目标点的位置 $F(X_{t-1},Y_{t-1},Z_{t-1})$，然后调整方向游动。当虚拟鱼再次侦测到目标位置时，它会将原来的目标位置 $F(X_{t-1},Y_{t-1},Z_{t-1})$ 储存起来，再记录当前目标位置 $F'(X_t,Y_t,Z_t)$，然后进行预测点计算。在 $F'(X_t,Y_t,Z_t)$ 与 $F(X_{t-1},Y_{t-1},Z_{t-1})$ 所确定的直线上，以 $F'(X_t,Y_t,Z_t)$ 为中心点，作 $F(X_{t-1},Y_{t-1},Z_{t-1})$ 的对称点 $F'(X_{t+1},Y_{t+1},Z_{t+1})$，$F''(X_{t+1},Y_{t+1},Z_{t+1})$ 就是预测点。然后调整虚拟鱼的方向游动即可。

预测点坐标计算公式为

$$\begin{cases}\dfrac{X_{t+1}+X_{t-1}}{2}=X_t\\ \dfrac{Y_{t+1}+Y_{t-1}}{2}=Y_t\\ \dfrac{Z_{t+1}+Z_{t-1}}{2}=Z_t\end{cases}\quad(16)\Rightarrow\begin{cases}X_{t+1}=2X_t-X_{t-1}\\ Y_{t+1}=2Y_t-Y_{t-1}\\ Z_{t+1}=2Z_t-Z_{t-1}\end{cases}\qquad(17)$$

求出预测点后，就可以将点的坐标交给调整方向算法处理，完成带有预测机制的方向调整。

7　程序截图

图 11 和图 12 为在程序中虚拟鱼应用了上面所述的捕食和逃逸算法后的行为。如图 11 所示，小鱼与鲨鱼在水中自由游动时，鲨鱼发现食物，随后小鱼发现敌人。图 12 为小鱼和鲨鱼运行带有预测机制的方向调整算法进行转向。

图 11　鲨鱼发现食物，小鱼发现敌人

图 12　鲨鱼和小鱼调整各自的游动方向

结束语　本文提出的一系列算法均是以虚拟鱼为研究对象，从实际编程实现过程中设计、修改、总结提炼而来。指出了目前文献中一些比较理论化的知识的不足之处，并较为成功地实现了虚拟鱼在水下的逃逸与捕食的行为，为研究和实现虚拟生物更高级的智能行为打下了基础。当然，在此算法中还存在着一些不足之处，如没有考虑虚拟鱼自身偏角等细节问题，有待在今后的研究中不断完善。

参 考 文 献

[1]　Reynolds C W. Flocks, Herds, and Schools: A Distributed Behavioral Model[J]. Computer Graphics, 1987, 21(4): 25-34

[2]　Tu Xiaoyuan. Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior [D]. Toronto: University of Toronto, 1996

[3]　Stephens K , Pham B , Wardhani A . Modelling Fish Behaviour [R]. USA: The Association for Computing Machinery, Inc, 2003

[4]　权晓林，钟绍春，王文永，等. 基于人工生命方法的虚拟鱼行为模型[J]. 计算机仿真学报，2006：45-51

[5]　班小娟，宁淑荣，艾冬梅. 人工鱼[M]. 北京：科学出版社，2007