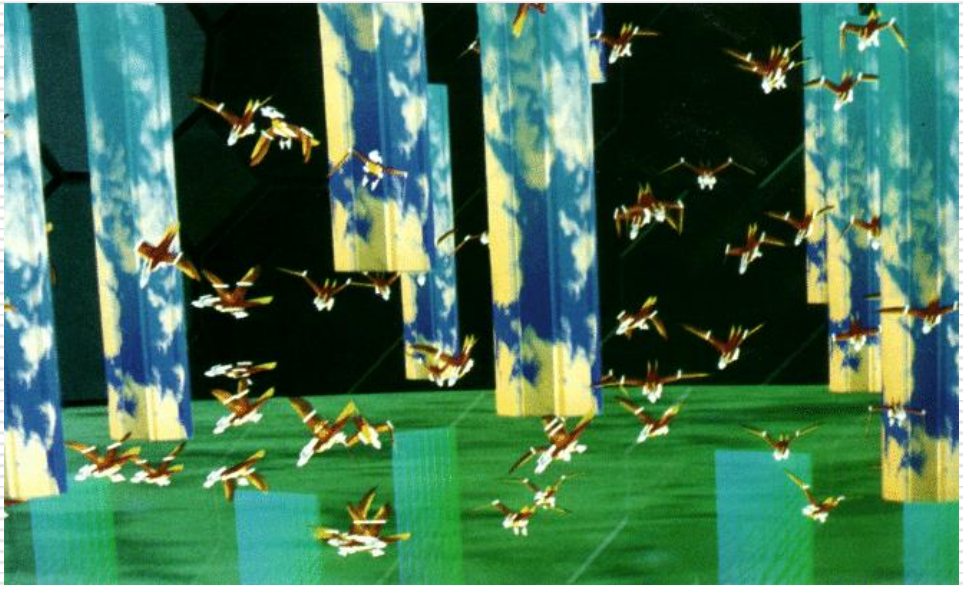
群 组 动 画

(Crowd and Group Animation)

金小刚
Email: jin@cad.zju.edu.cn
浙江大学CAD&CG国家重点实验室
蒙民伟楼512室

Boids模型

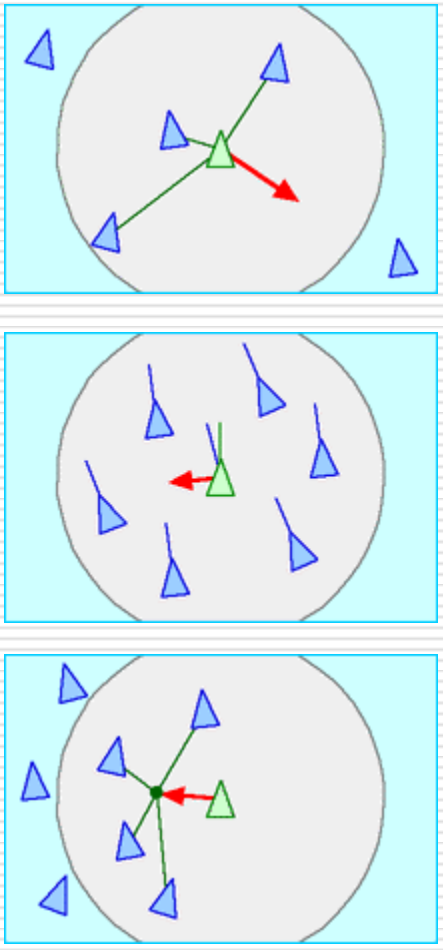


- 参考文献

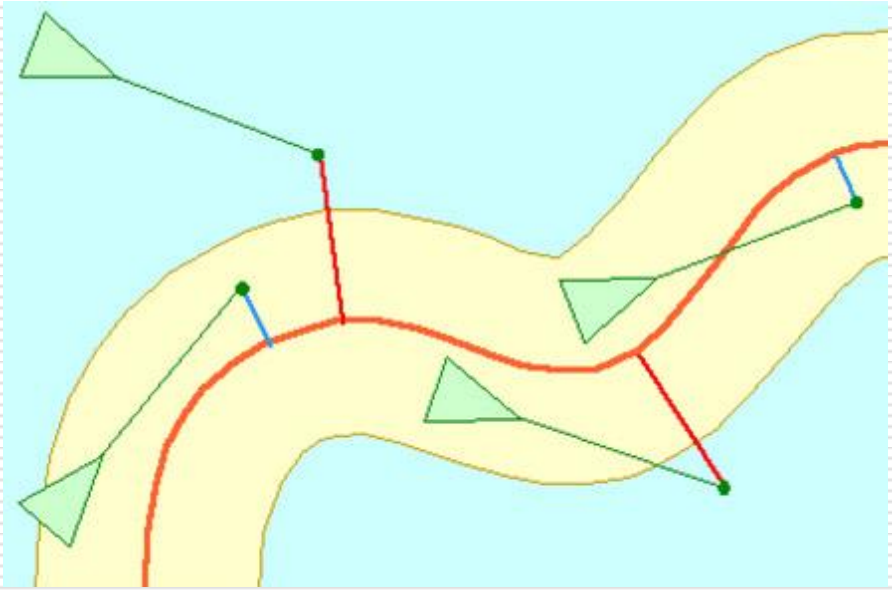
Craig W. Reynolds. 1987. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.* 21, 4 (August 1987), 25-34. (Google citations: 10265次, 2018.12.11)

优先级递减的群体模拟三大原则

- 碰撞避免原则(Collision Avoidance)
  - 避免与相邻的群体成员相碰
- 速度匹配原则(Velocity Matching)
  - 尽量保持与周围邻居群体成员的速度匹配
- 群体合群原则(Flock Centering)
  - 群体成员尽量靠近



Steering Behaviors for Autonomous Characters



- 参考文献

Craig W. Reynolds, Steering Behaviors For Autonomous Characters, *Game Developers Conference 1999*. (Google citations: 1647, 2018.12.11)

很多游戏引擎采用了这些技术，如UNITY 3D，Cocos2d-x等。

大纲

- 介绍
- 简单的车辆模型(Simple Vehicle Model)
- 模型的物理原理
- 导航行为
  - 一个或两个角色的行为
  - 群组行为
- 行为的结合
- 结论

介绍

- 在游戏和动画中，如何对自主角色进行栩栩如生和即兴的方式进行导航？

- 运动行为的层次结构

Action Selection: strategy, goals, planning

↓

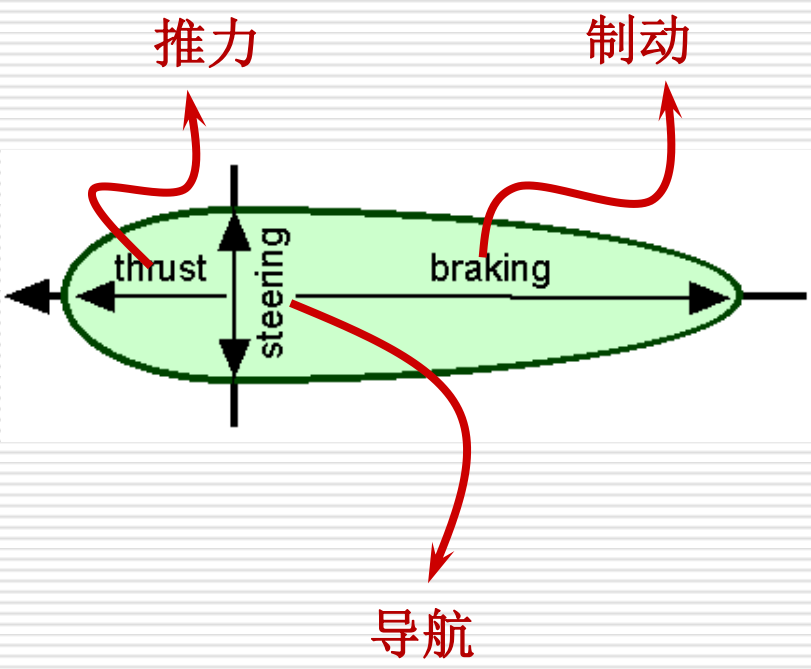
Steering: path determination

↓

Locomotion: animation, articulation

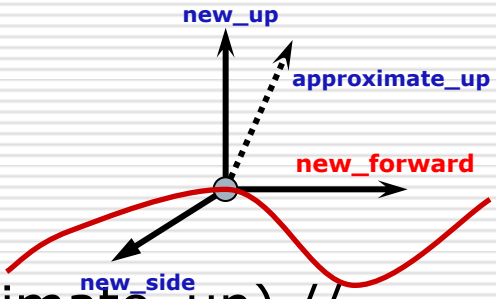
简单的车辆模型(Simple Vehicle Model)

- Simple Vehicle Model
  - mass scalar
  - position vector
  - velocity vector
  - max_force scalar
  - max_speed scalar
  - orientation *N* basis vectors



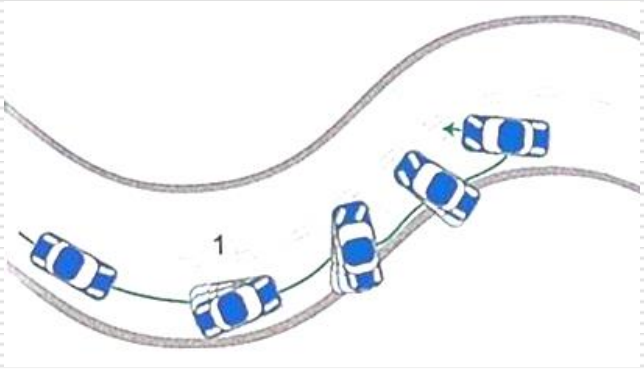
车辆模型的物理原理

- 简单车辆模型的物理原理为前向欧拉积分法：
  - **steering_force** = truncate (steering_direction, max_force)
    acceleration = steering_force / mass
    velocity = truncate (velocity + acceleration, max_speed)
    position = position + velocity
- 构造新的基向量（局部坐标系）：
  - **new_forward** = normalize (velocity)
    approximate_up = normalize (approximate_up) // if needed
    new_side = cross (new_forward, approximate_up)
    new_up = cross (new_side, new_forward)



限制(Constraint)

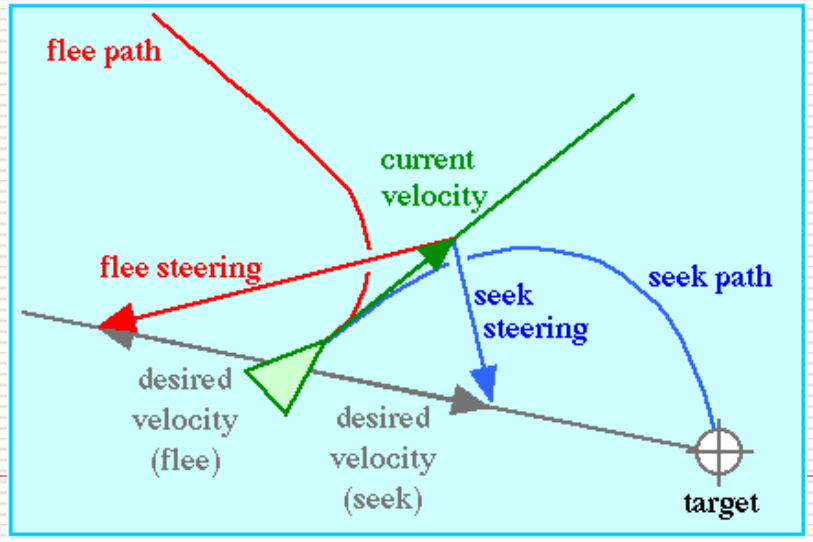
- 这个简单的车辆模型不能模拟如打滑、自旋等效果；



- 而且，该模型允许车辆在速度为零时转弯。

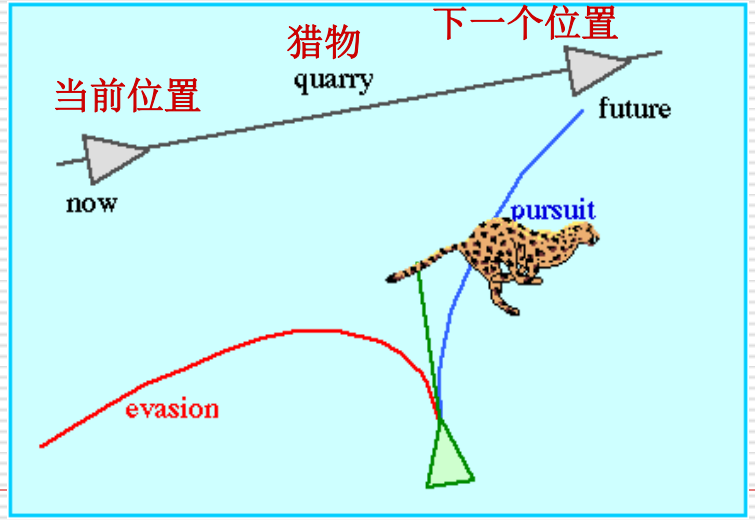
寻找和逃离(Seek and Flee)

- http://www.red3d.com/cwr/steer/SeekFlee.html
- 寻找(或追逐)一个静态的目标
  - **desired_velocity** = normalize (target - position) * max_speed
    steering = desired_velocity – velocity
  - 把追逐反过来，即为逃离



追逐和躲避(Pursuit and Evasion)

- http://www.red3d.com/cwr/steer/PursueEvade.html
- 追逐(Pursuit)与寻找(Seek)非常类似，其区别在于目标为一移动的角色(猎物)。
  - 假设猎物在预测区间T内不会转向。
  - 猎物在将来的位置可通过把它的当前速度乘以T，并把该偏移量与当前位置相加来得到。

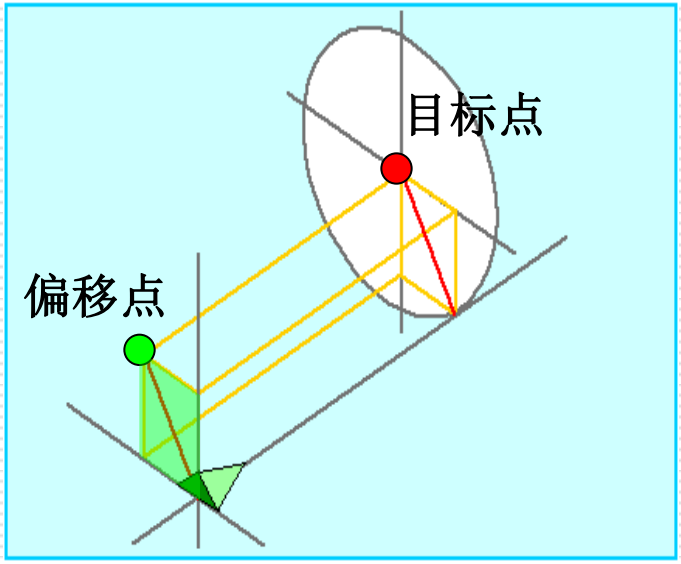


追逐和躲避(Pursuit and Evasion)

注：也有把Pursuit and Evasion说成Seek and Flee的

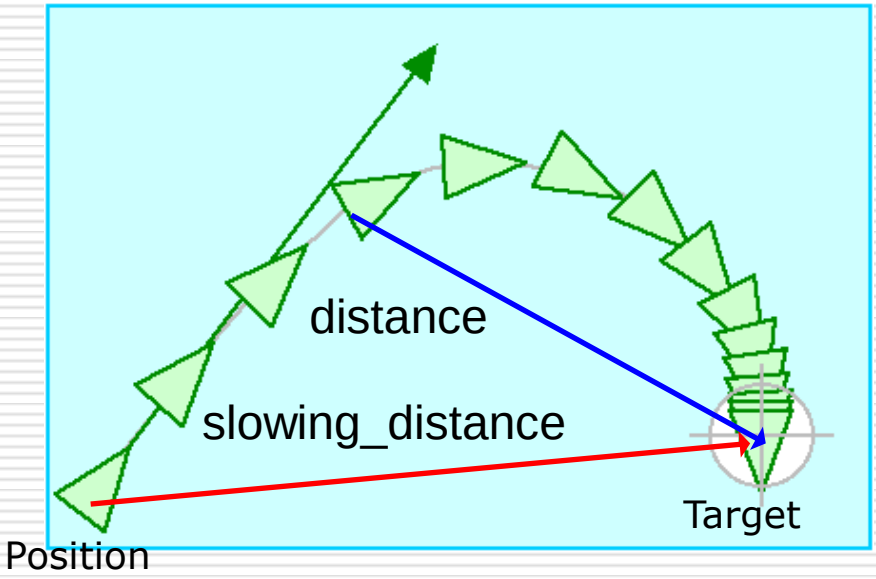
偏移的追逐(Offset Pursuit)

- 沿着一条接近目标的路径追逐，但是不直接进入移动的目标（例子，战斗机进行空中扫射）；
- 对预测的目标点进行局部化(在目标角色的局部坐标系)，把局部目标投影到角色的侧向(Side-Up)平面，对偏移向量进行归一化，乘上比例因子R(偏移值)，并把它加到 局部目标点, 然后把该值进行全局化(变换回世界坐标系)。
- 采用寻找(Seek)行为来接近偏移点。



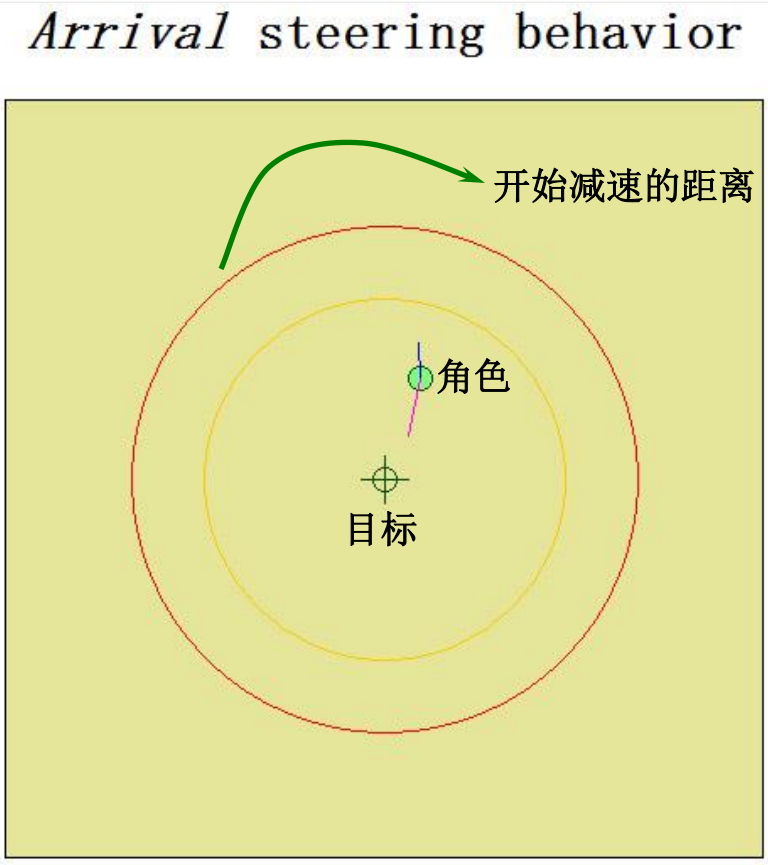
偏移的追逐(Offset Pursuit)

到达(Arrival)



- 当角色远离目标时，到达行为与寻找行为几乎是一样的；
- 但是，寻找行为是以全速穿过目标。而对于到达行为，角色将放慢速度，直到停止，并与目标位置重合。
- 开始减速的距离为到达行为的一个参数；

到达(Arrival)



到达(Arrival)

- http://www.red3d.com/cwr/steer/Arrival.html
- **target_offset** = target – position **(这是一个矢量)**

  distance = length (target_offset) **(这是一个标量)**

  ramped_speed = max_speed * (distance / slowing_distance)

  clipped_speed = minimum (ramped_speed, max_speed)

  desired_velocity = clipped_speed * **target_offset** / distance

  steering = desired_velocity - velocity

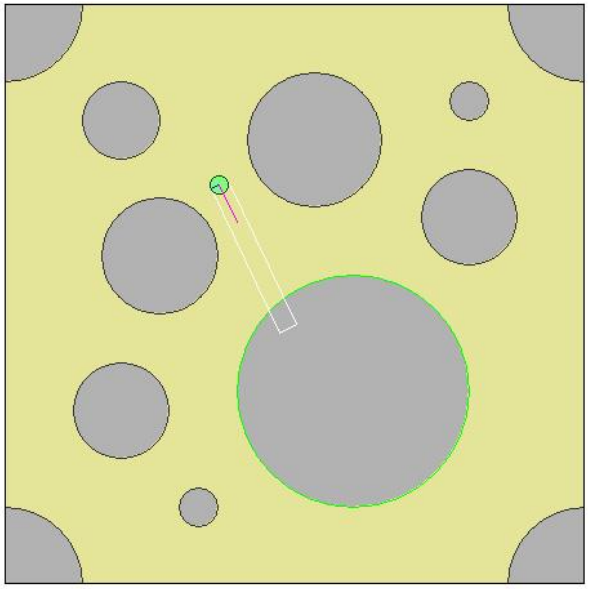
slowing_distance: 表示开始减速的距离

DEMO

障碍避免(Obstacle Avoidance)

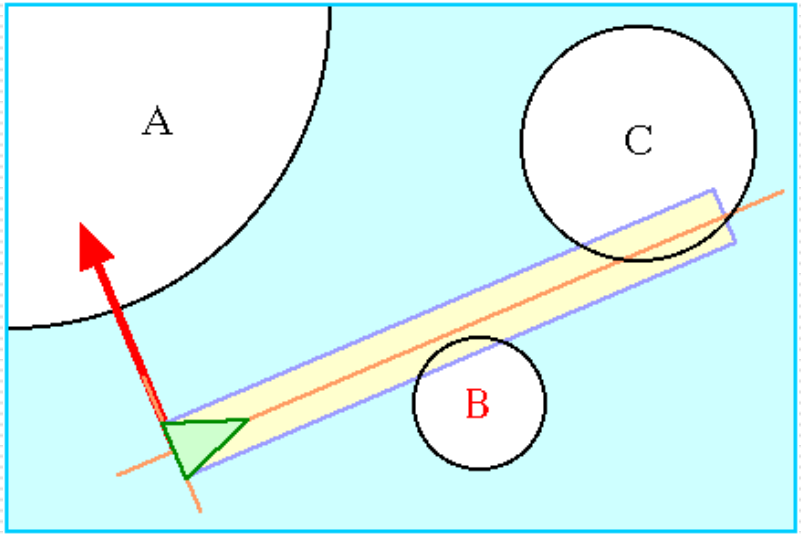
- http://www.red3d.com/cwr/steer/Obstacle.html
- 该行为的目标是：在该角色前面，保证有一个假想圆柱的自由空间。

Obstacle Avoidance steering behavior



障碍避免(Obstacle Avoidance)

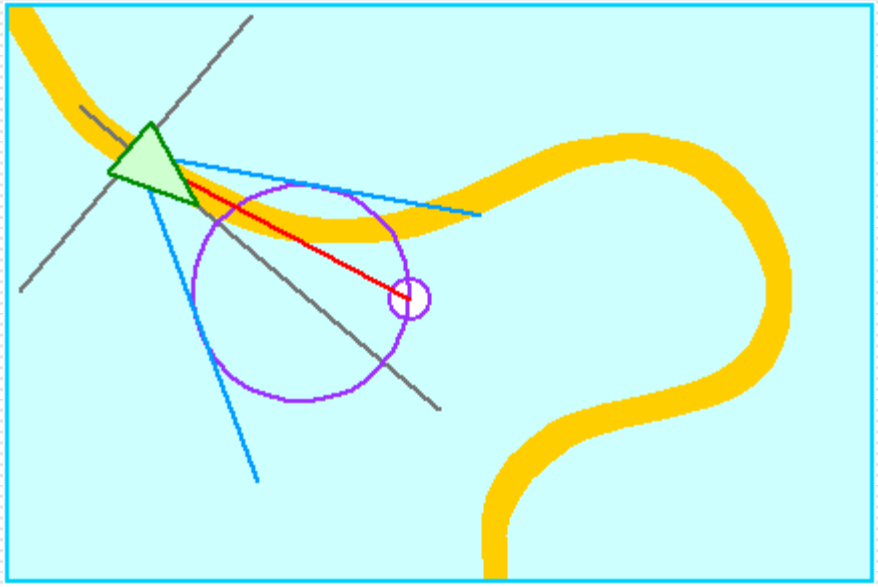
- 障碍避免返回的值：
  - 返回避免最有威胁障碍物的导航值；
  - 如果没有碰撞是迫在眉睫的，返回一个特殊值（空值，或零向量），表示在这一时刻不需要纠正量。



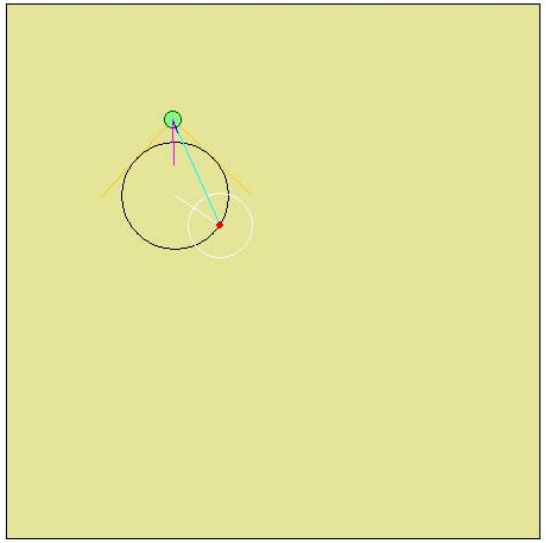
障碍避免(Obstacle Avoidance)

徘徊(Wander)

- http://www.red3d.com/cwr/steer/Wander.html
- 导航方向表示为红色向量
- 图中的大圆对导航进行约束
- 小圆对导航的随机偏移量进行约束



徘徊(Wander)

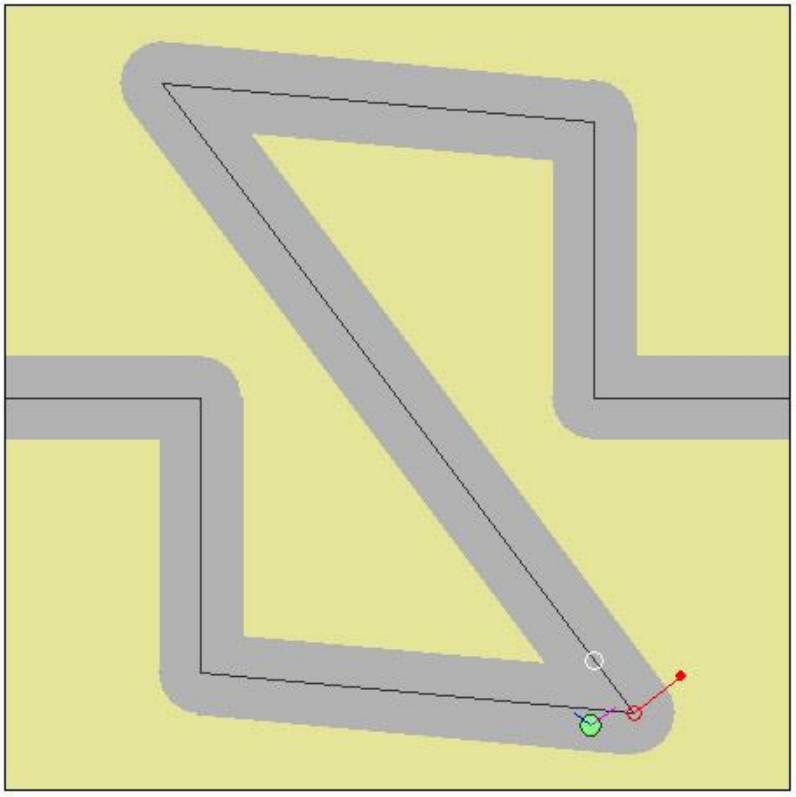


徘徊(Wander)

- 一种随机导航，但又有一定的秩序：某一帧的导航方向与下一帧的导航方向有关。这将产生比生成一随机导航方向更有趣的运动。
- 徘徊行为将保持一个“徘徊方向”这一状态，这里表示为一个红点。调整的徘徊方向约束到一个黑圆上。 黑圆的半径为1，其中心处于沿角色向前轴(*forward* axis)sqrt(2)单位长度的位置。
- 徘徊模型的参数：
  - **Strength**：从0(没有转向) 到 1.
  - **Rate**：控制 “徘徊方向“变化的程度，从0（没变化）到1 (快速的飘忽不定的变化），这里表示为白圆的半径 (上图显示的为 0.6，表示最大的随机偏移量)
- 导航方向表示为青色的矢量。车辆的速度用品红矢量来表示，导航力用蓝色矢量来表示。

路径跟随(Path following)

Path Following steering behavior



路径跟随(Path following)

- 绿圆显示了路径跟随行为。其任务是在一个给定的方向进行路径遍历（左边进入，右边退出），同时保持其中心位于灰色区域。

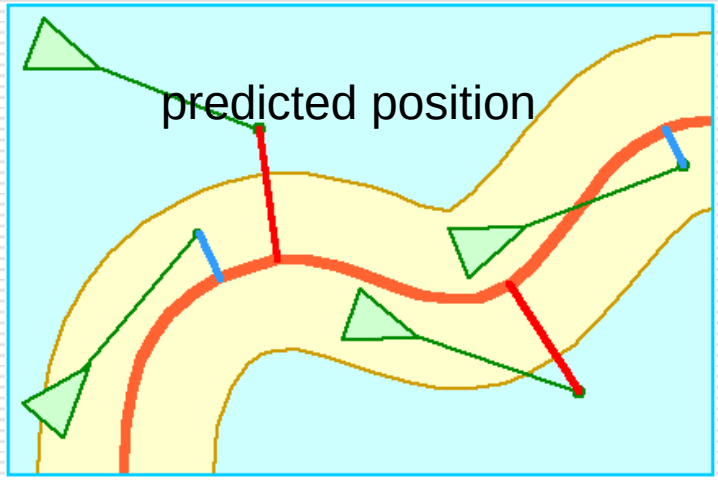
- 在该例子中，路径定义为一条折线和一个半径。该行为与抑制(containment)和沿墙运动(wall following) 相关，其区别在于，在路径跟随中，路径意味着行驶方向；

- 在路径跟随行为中，只有当车辆将离开灰色区域时，才执行校正导航。

路径跟随(Path following)

- http://www.red3d.com/cwr/steer/PathFollow.html
- 沿着一条路径移动角色，并同时保持在脊柱线的指定半径内。
- 投影距离：小于路径半径时，不需要进行导航校正。
- 否则，把预测的位置投影到路径上，把该点作为目标点，并进行寻找(Seeking)行为。

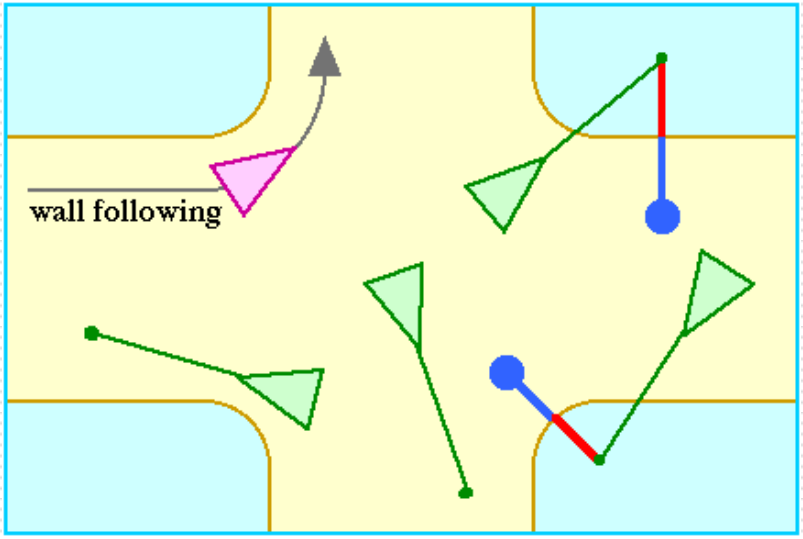
投影距离(Projection distance)：
从预测位置到最近路径点的距离

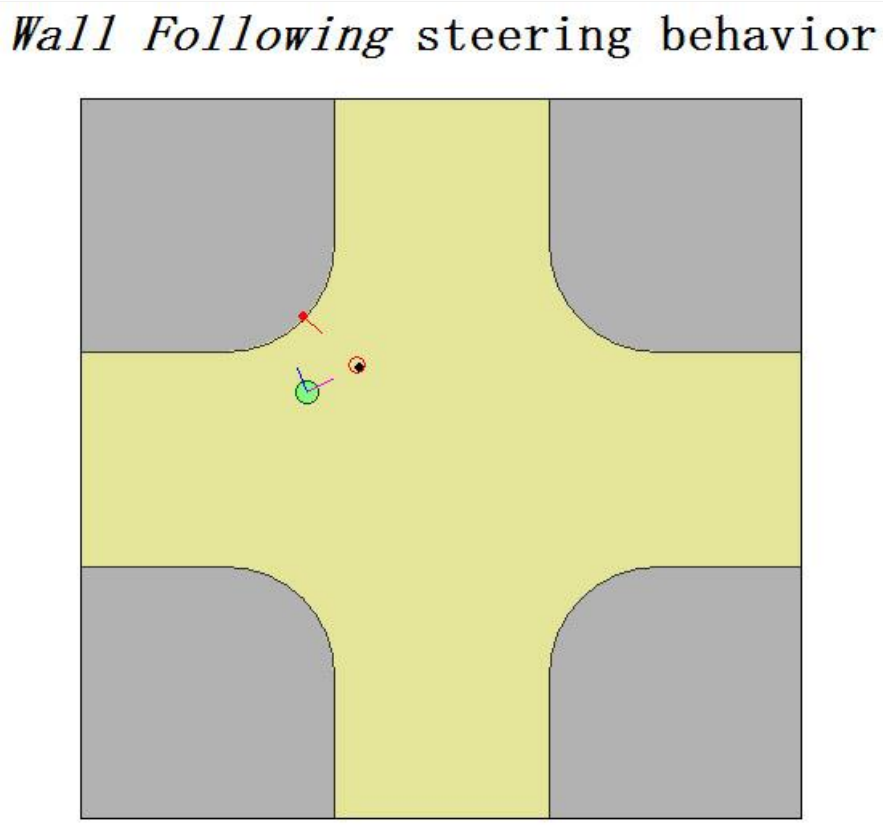


DEMO

沿墙运动(Wall Following)

- http://www.red3d.com/cwr/steer/Wall.html





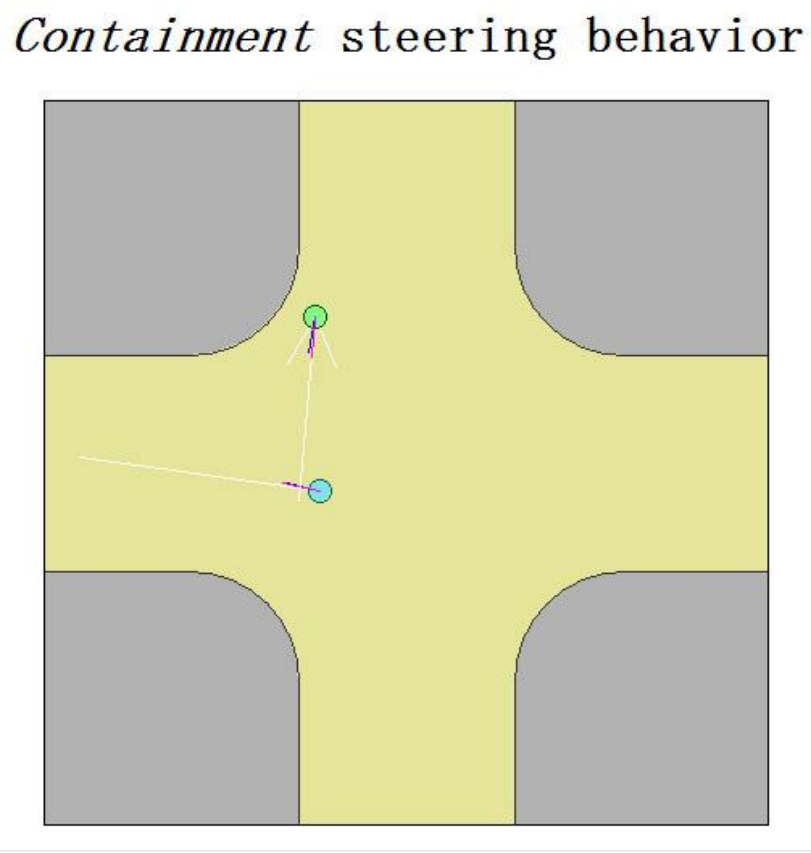
沿墙运动(Wall Following)

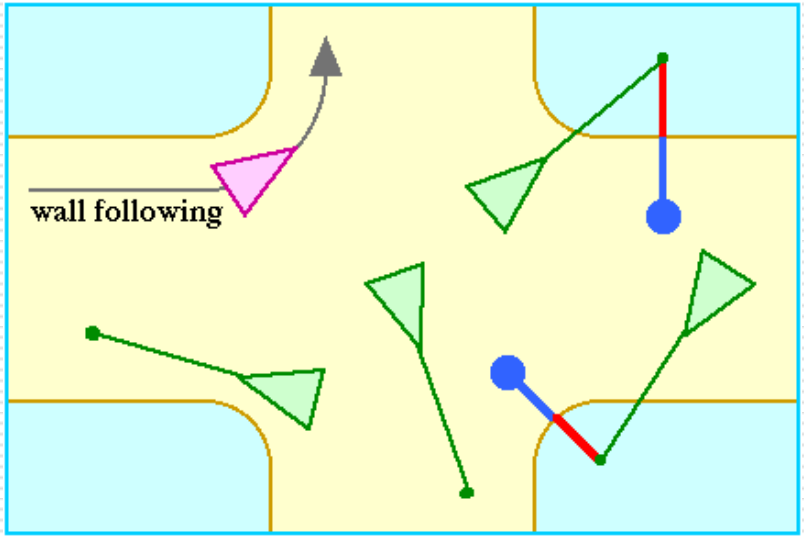
- 绿色车辆使用沿墙运动行为以使得它沿标记为灰色的区域平行移动并保持一定的偏移量。汽车在移动时，需要保持与墙给定的距离。车辆的速度由品红色矢量表示，导航力由一蓝色矢量表示。
- 沿墙运动行为的实现：首先根据当前的速度预测一定时间后车辆的位置，把该位置投影到墙上离它最近的点(红点)，把该点沿墙的法向(红线)移动给定的偏移量，从而生成一个目标点(红色的圆)。最后采用寻找(Seek)行为来接近目标点。
- 车辆与墙之间的通信通过一个通用的表面协议(generic surface protocol)来实现：车辆经过墙表面时，经过询问可得到墙上的最近点和该点的法向。因此，导航行为并不需要知道墙壁表面的形状信息。

抑制 (Containment, 包含在一个容器中)

抑制 (Containment)

- http://www.red3d.com/cwr/steer/Containment.html

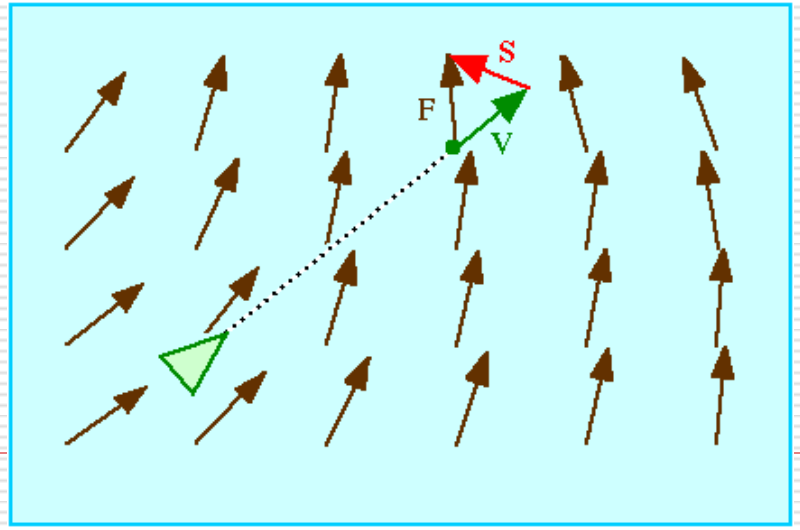




- 车辆使用抑制(或广义的障碍避免)行为使得它远离标记为灰色的区域。这种行为与前述的障碍避免密切相关（障碍物通过包围球或圆柱表示）。然而，这种行为可以处理任意形状，并允许车辆导航接近障碍物的表面。
- 车辆通过探测点测试前面的空间(用白线表示), 当没有接触障碍物时，车辆将采用徘徊行为，以避免使得它的运动与“通道”对齐。当探测点接触到障碍物时，把该点投影到障碍物的表面，并计算这点的法向，沿法向偏移一定量的点作为Seek目标进行导航。
- 车辆与障碍物之间的通信通过一个通用的表面协议(generic surface protocol)来实现：车辆询问探测点是否位于障碍物内部，如果是的话，得到障碍物上的最近点和该点的法向。因此，导航行为并不需要知道墙壁表面的形状信息。

流场跟随行为(Flow Field Following)

- http://www.red3d.com/cwr/steer/FlowFollow.html
- 假设运动区域已经有一个速度流场
- 估算角色将来的位置并计算在这点的流场
- 得到的速度(F)即为我们期望的速度，导航方向(S)为期望速度和当前速度的差



DEMO

流场的设置

- Paint方法(massive 软件支持这个功能）

- 基于径向基函数的散乱点插值方法

Xiaogang Jin, Jiayi Xu, Charlie C. L. Wang, Shengsheng Huang, Jun Zhang: Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields. *IEEE Computer Graphics and Applications* 28(6): 37-46 (2008)

Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields

- 二维散乱数据插值问题可以描述为：

  给定k个平面约束点$\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$和对应的值$\{h_1, h_2, \ldots, h_k\}$, 求一个光滑曲面$f(\mathbf{x})$,使得$f(\mathbf{c}_i)= h_i, 1< i<k$.

- 求解如下能量函数最小化问题的变分解：

$$E = \int_{\Omega} f^2{}_{xx}(\mathbf{x}) + 2 f^2{}_{xy}(\mathbf{x}) + f^2{}_{yy}(\mathbf{x})$$

- 采用RBF(Radial Basis Function)函数，解可表达为：

$$f(\mathbf{x}) = \sum_{j=1}^{n} d_j \phi(\mathbf{x} - \mathbf{c}_j) + p(\mathbf{x})$$

其中d_j为系数， $p(\mathbf{x})$为一个多项式， $\phi(\mathbf{x}) = |\mathbf{x}|^2 \log(|\mathbf{x}|)$

Interactive Control of Large-Crowd Navigation in Virtual Environments Using Vector Fields

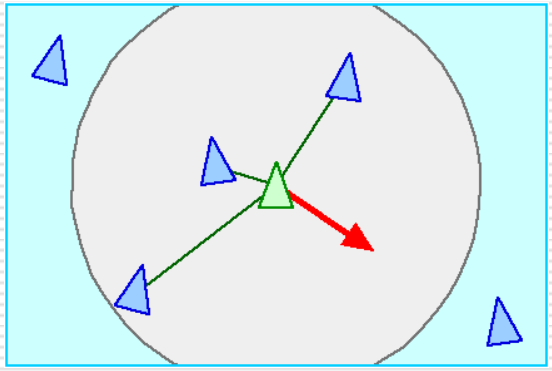
- 转为为求解一个对称且半正定的线性方程组：

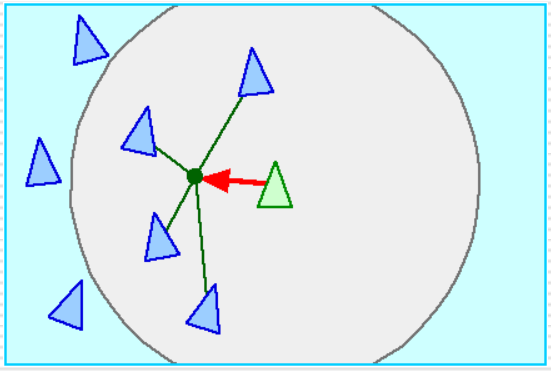
$$\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \cdots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \cdots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \cdots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \cdots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \cdots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \\ p_0 \\ p_1 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

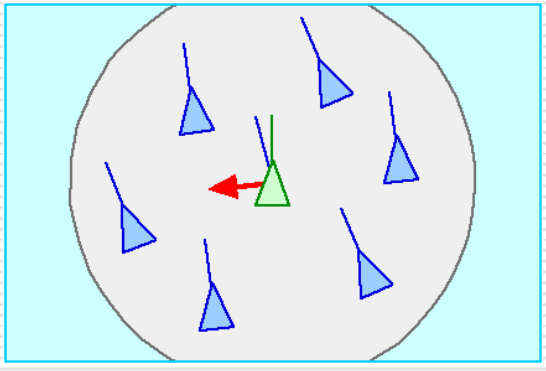
Demo

群组的行为(Group Behavior)

- 分离(Separation)、聚集(cohesion)和对齐(alignment)与群体的行为有关；
- 导航行为决定一个角色如何对周围的角色做出反应；





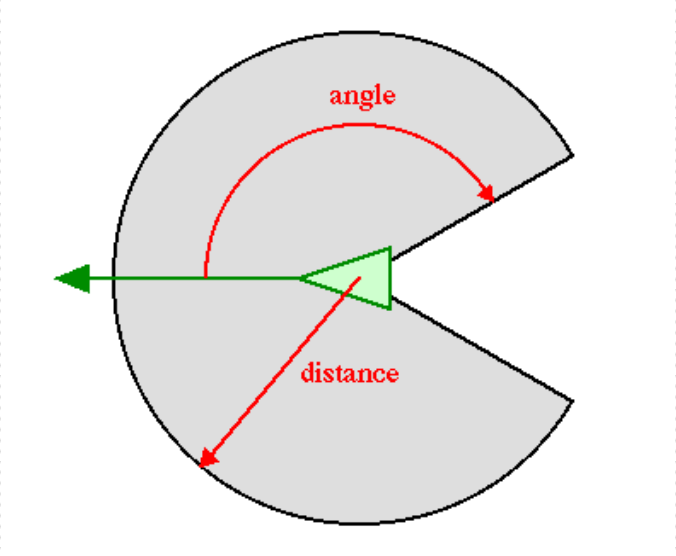


Repulsive force is computed by subtracting the positions of our character and the nearby character

平均速度为我们期望的速度

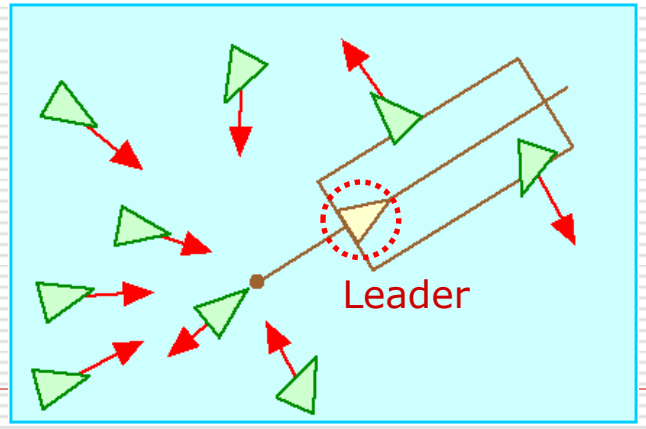
邻居(Neighborhood)

- 角色的局部邻居
  - 邻居群组成员由一个距离 (从角色的中心开始度量)和一角度(从角色的运动方向开始度量)来刻划，该角度定义了角色的感知视野；



跟随领导(Leader Following)

- http://www.red3d.com/cwr/steer/LeaderFollow.html
- 如果一个跟随成员发现自己处于领导前面的一个矩形区域，它会横向远离领导者的路径；
- 否则，到达(arrival)目标为领导后面的一个偏移点；
- 跟随成员采用分离行为来避免相互拥挤；



跟随领导(Leader Following)

行为的组合(Combining Behaviors)

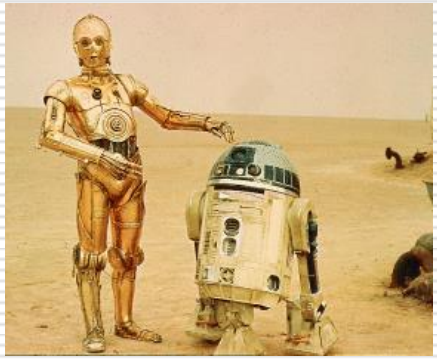
- 组合行为可以采用如下两种方式:
  - 切换(Switch)
  - 混合(Blending)

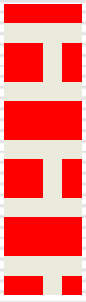
- 最直接的方法可简单地计算每个组的导航行为并把他们加权求和。

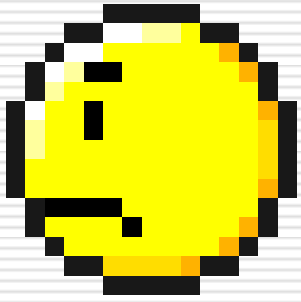
DEMO

方法缺陷

- 运动可能会不太自然
  - 运动像一个完美的机器人
- 碰撞避免后
  - 一遍又一遍沿着障碍物转
- 碰撞反应
  - 碰撞避免有可能失败







解决方法

- 运动可能会不太自然
  - 加入噪声(分形噪声或Perlin噪声)和湍流(turbulence)
- 碰撞避免后
  - 保持碰撞避免力一段时间
- 碰撞反应
  - 让角色停止并等待

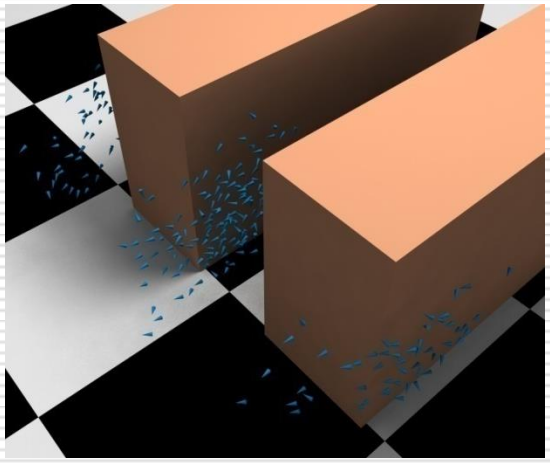
内在噪声感知的昆虫群体模拟

- 昆虫群是自然界中的常见动物群体，其特点是个体在运动过程中除了会遵循群体的运动方向，同时也会产生固有的随机运动现象
- 生物学上将其称为“固有噪声”(Inherent noise)。现有的群组模拟方法并不能很好地模拟这一现象
- 我们提出了一种内在噪声感知的用于模拟虫群行为的运动合成方法

Xinjie Wang, **Xiaogang Jin**, Zhigang Deng and Linling Zhou, “Inherent Noise-Aware Insect Swarm Simulation, ” *Computer Graphics Forum*, 2014, 33(6): 51-62.

内在噪声感知的昆虫群体模拟

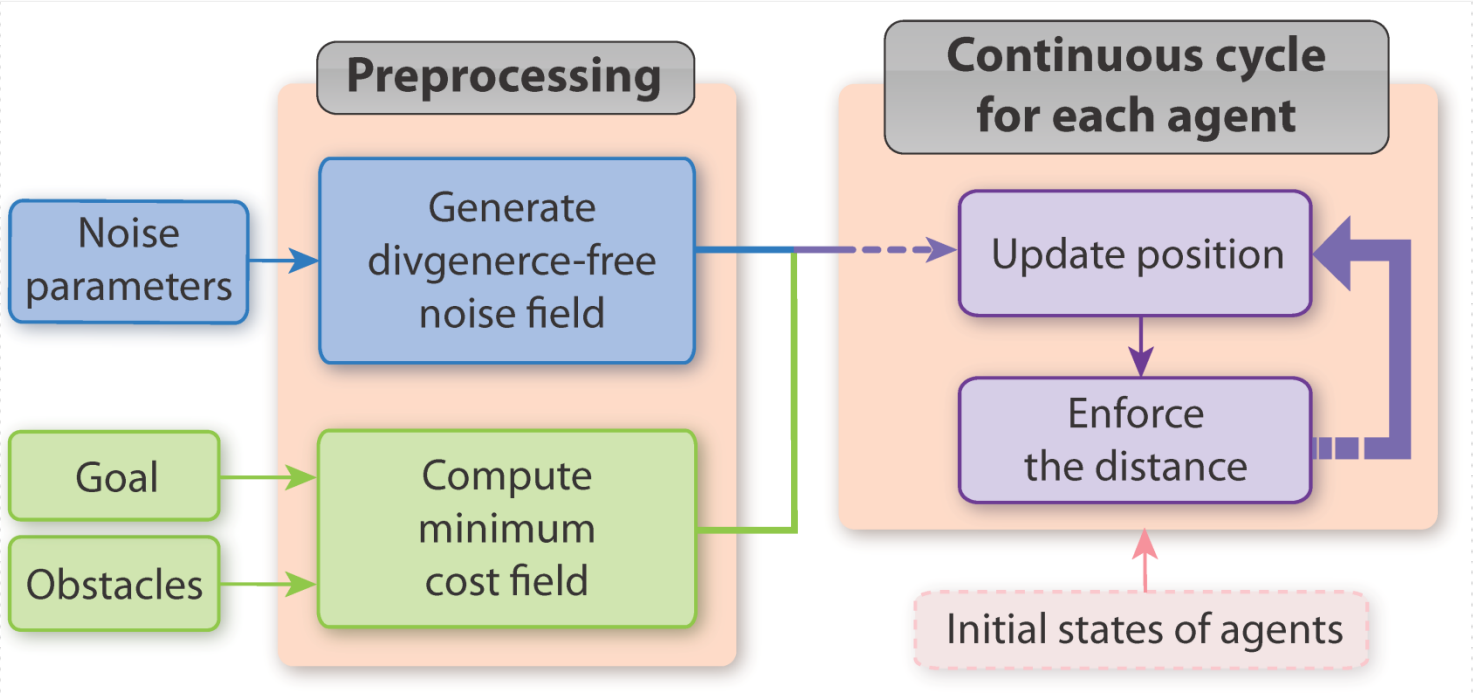




- 使用无散噪声场来获得具有无碰撞性质的昆虫“固有噪声”运动
- 设虫群有一个飞行目标，本方法将计算出最优飞行轨迹，并能够将“固有噪声”运动与虫群的飞行运动进行合成，同时避免与障碍物的碰撞
- 可以用来模拟常见的虫群行为，如趋光行为、攻击行为和迁徙行为
- 该方法生成的虫群效果在视觉上与真实虫群十分接近

算法流程

- 算法框架图：本文算法分为预处理和循环模拟两部分



Inherent Noise-Aware Insect Swarm Simulation

Demo

OpenSteer

Steering Behaviors for Autonomous Characters

- OpenSteer is a C++ library to help construct steering behaviors for autonomous characters in games and animation. In addition to the library, OpenSteer provides an OpenGL-based application called OpenSteerDemo which displays predefined demonstrations of steering behaviors. The user can quickly prototype, visualize, annotate and debug new steering behaviors by writing a plug-in for OpenSteerDemo.

- OpenSteer provides a toolkit of steering behaviors, defined in terms of an abstract mobile agent called a "vehicle." Sample code is provided, including a simple vehicle implementation and examples of combining simple steering behaviors to produce more complex behavior. OpenSteer's classes have been designed to flexibly integrate with existing game engines by either layering or inheritance.