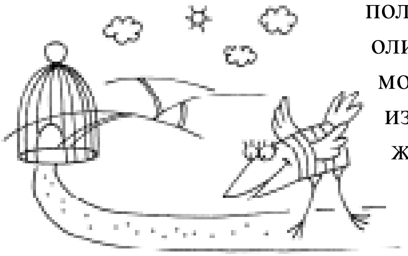
И в шутку и всерьез

КРЕЙГ РЕЙНОЛЬДС О ВЗАИМОДЕЙСТВИИ АВТОНОМНЫХ ПЕРСОНАЖЕЙ

Статья посвящена методам создания сообществ, члены которых действуют автономно в реальном времени, реагируют на взаимодействие с пользователем, а также друг на друга и свое окружение. Их «характеры» основаны на простой умственной модели, позволяющей делать выбор из нескольких конфликтующих целей. Персонажи визуализируются с помощью 3D библиотеки анимированных движений.

ВСТУПЛЕНИЕ

Реальный мир – это беспокойное место, наполненное людьми, животными и средствами передвижения. В противоположность реальному миру, игровые миры пусты и статичны. Обычно один персонаж двигается под управлением игрока, несколько других персонажей двигаются изолированно и еще несколько объектов окружающей среды двигаются в простом цикле. В статье описывается техника пополнения игрового мира большим количеством автономных персонажей, которые ведут себя осмысленно. Если того требует их роль, они могут действовать согласованно, реагируя друг на друга, окружающую среду и пользователя, олицетворяемого одним из персонажей.



СОВМЕСТНАЯ ДЕЯТЕЛЬНОСТЬ

Автономные персонажи некоторых типов существуют в большинстве игр. Любой персонаж, не напрямую контролируемый человеком, должен иметь некоторый уровень автономности, включая врагов, союзников и иногда некоторых нейтральных игроков. Исторически все начиналось с сильно упрощенных автономных персонажей (которые могли только слепо следовать сценарию без возможности реагировать на динамичную окружающую среду), затем появились реагирующие на среду персонажи [Reyn99], а потом и персонажи, которые могут учиться [Gran98], рассуждать [Lair00] и строить планы [Fung99].

Понятия, обсуждаемые в этой статье, восходят к ранним работам по моделированию поведения стай птиц [Reyn87]. Огромную библиографию по автономным персонажам можно найти в [Reyn99].

Непосредственно к теме этой статьи относятся игры, которые содержат большие группы персонажей, взаимодействующих между собой или реагирующих на действия пользователя-человека. Например, множество игр-симуляторов Maxis имеют это свойство. То же можно сказать про SimCity, SimAnt, SimTower, к которым сравнительно недавно добавились The Sims. Игры Lemmings и Populous также имеют право быть отнесенными к этой группе: они содержат большое количество автономных персонажей, которые реагируют на пользователя. Немного другая направленность наблюдается у игр типа Myth и Warcraft: они позволяют группам людей собираться вместе и ставить цель, которую они пытаются достичь, остерегаясь препятствий и друг друга.

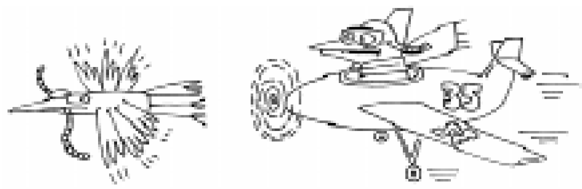
ПОВЕДЕНЧЕСКИЕ МОДЕЛИ

В виртуальном мире персонажи можно разделить на несколько типов: они могут быть статичными, периодичными (например, мерцающее пламя), управляемыми пользователем (как олицетворение пользователя-игрока) или иметь некоторую степень автономности. Автономные персонажи процедурно управляются регулирующей программой, которая соответствует некоторой поведенческой модели. Эта регулирующая программа отображает окружающую среду персонажа в действия персонажа. Среда персонажа состоит из его виртуального мира (внешняя среда) и его памяти или любого другого умственного процесса (внутренняя среда). Степень автономии персонажа может широко варьироваться. С одной стороны, автономная «машина» может просто ехать по улице, останавливаясь, когда зажигается красный свет. (Заметьте, что в рамках этой статьи мы объединяем средство передвижения и его водителя/пилота в единое существо). С другой стороны, персонаж может иметь всю свободу действий, предоставляемую игроку-человеку, так что он может играть роль врага или союзника.

Персонажи, описываемые в этой статье, основаны на очень простых физических моделях: материальная точка, обладающая скоростью, и привязка ее к местности. Персонаж двигается посредством применения конечной управляющей силы, которая меняет его скорость, по которой и вычисляется новое положение. На скорость также влияют моделируемые помехи и сила трения. Как результат, позиция, скорость, ориентация и визуальное представление персонажа определяются поведенческой моделью, в первую очередь, посредством контроля силы управления. (Для более детального рассмотрения этого понятия можно посмотреть [Reyn99]).

ГОЛУБИ В ПАРКЕ

Чтобы сделать понятия, описываемые в этой статье более конкретными, была написана демонстрационная программа, названная «Голуби в парке», в которой пользователь/игрок взаимодействует в реальном времени с большой группой простых автономных персонажей. Действие происходит на сымитированном лугу в виртуальном парке. Стая похожих на голубей птиц на земле ищут еду. Пользователь водит игрушечную радиоуправляемую машинку. Движущаяся машинка беспокоит птиц. Если машинка приближается медленно, то голуби отходят с ее пути. Если машинка подъезжает слишком близко или двигается слишком быстро по направлению к ним, птицы паникуют и



взлетают. Паника заразительна и, таким образом, спокойные птицы тоже взлетят, если взлетит достаточная часть их соседей. Когда голуби летят, они формируют стаю, согласно модели согласованного группового движения *boids* (*птушек*) [Reyn87]. Когда голуби находятся на земле, они формируют нежестко организованную 2D-стаю, по существу стадо. Виртуальный мир «Голубей в парке» включает в себя птиц, машинку, холмистую местность (см. рисунок 1), невидимую сферическую преграду, которая окружает птиц (чтобы удержать их от блужданий), и дополнительные декорации, такие как небо и некоторые удаленные деревья.

Вдобавок к беспокойству голубей машинкой пользователь также может нажать кнопку, которая симулирует громкий звук, похожий на гудок. И проезжающая рядом машинка, и гудок пугают птиц и служат причиной для того, чтобы они взлетели. К тому же, необходимость сдвинуться с места заразительна, так что каждая птица чувствительна к количеству находящихся рядом птиц, которые только что взлетели (их не

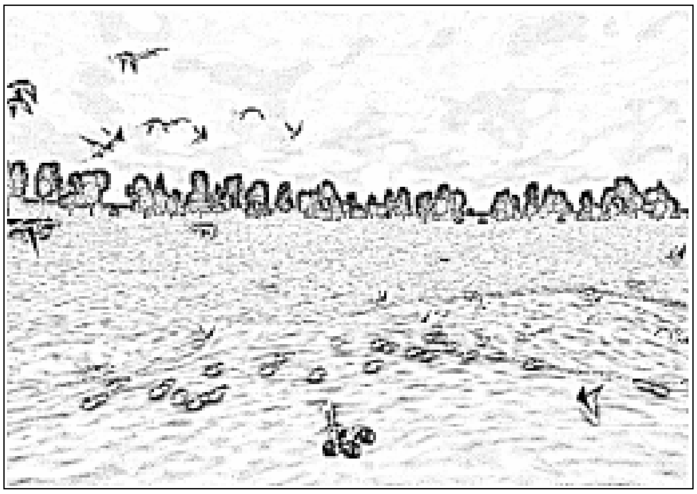


Рисунок 1. Типичный пейзаж программы «Голубей в парке».

беспокоят птицы, просто пролетающие рядом), и если это количество превышает определенную пороговую величину, они взлетают тоже. В результате, если пользователь напугает одну птицу в плотной толпе, большое количество птиц взлетит, откликаясь на волну сигнала тревоги, распространяющуюся в толпе.

Будучи испугана в первый раз, птица пролетает короткое расстояние и приземляется опять, так что она может вернуться к еде, как показано на диаграмме простых состояний на рисунке 2. Но каждая птица имеет величину раздражения, которая увеличивается каждый раз, когда птица напугана, и далее со временем ослабляется. Если птица уже обеспокоена, когда ее пугают, она улетит дальше, перед тем как сесть на землю. В результате, в первый раз, когда птица напугана, она улетит на несколько футов (1 фут = 30,48 см) от своей начальной позиции, если ее напугать снова до того, как пройдет достаточно времени, чтобы она успокоилась, она улетит дальше, стараясь избавиться от назойливого пользователя. Длительность полета контролируется двумя параметрами: один устанавливает приблизительное количество времени, которое птица проводит, поднимаясь в воздух после взлета, другой – приблизительное количество времени, перед тем как птица начнет спускаться на посадку. Оба параметра определяются степенью обеспокоенности птицы. Индивидуальный полет происходит по исключительно простой арке вверх и потом вниз, но имеет тенденцию следовать тому, что делают ближайшие соседи по стае.

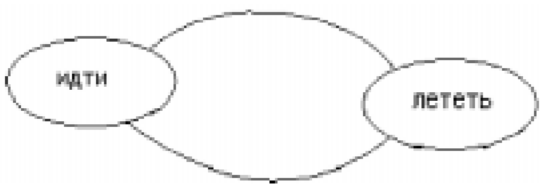


Рисунок 2. Диаграмма поведенческих состояний для голубей.

Движение птицы определяется набором нескольких моделей поведения [Reyn99]. Первоначально птицы собираются в стаю (согласно правилам BOYDs, которые включают тактики *разделения, выравнивания и сплоченности*), избегая препятствий (используя тактику, получившую название *сдерживание* в [Reyn99]), таких как физические особенности местности и окружающий их (но невидимый зрителям) огораживающий купол. Они также реагируют на олицетворение пользователя (машинку), как описано в следующем разделе.

Анимация птиц создается из небольшого множества заранее анимированных циклов и учитывает несколько параметров поведенческой модели. Эти поведенческие параметры таковы: позиция и ориентация персонажа, сигналы идти/лететь и (извлекаемая из ориентации) степень изменения уклона планирования (угла между плоскостью земли и траекторией движения). Заранее анимированные циклы: *Идти, Лететь, Скользить* и *Изгиб Крыльев*. Когда птицы находятся на земле, они обычно представляются анимацией *Идти,* а анимация *Изгиб Крыльев* образует переход от *Лететь* к *Идти.* В полете контролер анимации выбирает между тактикой *Планировать* и несколькими вариантами тактики *Лететь,* основываясь на угле полета. Подъем использует анимацию *Лететь* на высокой скорости, полет на определенном уровне использует анимацию *Лететь* в более медленном темпе, спуск использует анимацию *Планировать*. Переходы между анимационными клипами занесены в список, так что они появляются, когда текущий цикл достигает точки, где он переходит в следующий выбранный цикл.

На рисунке 3 показана структура данных описывающая «голубей».

РЕАКЦИЯ НА ПОЛЬЗОВАТЕЛЯ

Автономные персонажи, какие использованы в «Голубях в парке», реагируют на пользователя двумя разными способами: реакция может быть моментальной или продолжительной. Пользователь может инициировать перемещение птицы из ходьбы в полет, либо сигналом «гудок», либо скоростью и расположением машинки (олицетворения пользователя). Переключение состояний и взлет – пример изолированной реакции на пользователя.

К тому же, голуби хотят сохранить расстояние от них до машинки пользователя, особенно когда она быстро двигается. Когда голуби чувствуют, что машинка подбирается слишком близко, они двигаются в направлении, которое удалит их от машинки, пытаясь сохранить направление текущего движения. Эта тенденция сочетать избегающее движение с другим поведением персонажа – пример продолжительной реакции на пользователя.

В обоих случаях эти реакции управляются динамически развивающимся соседством машинки, которое зависит от ее положения и скорости (скорость здесь рассматривается как вектор). Соседство определено в терминах радиуса и скорости движения машинки. Ее окрестность – полукруг сзади машины и полуэллипс спереди машины, одинаково ориентированный с передом машины, чья длина пропорциональна текущей скорости машинки. Рисунок 4 показывает форму окрестности, когда машинка стоит, едет медленно и едет быстро.

```
typedef struct pigeon
{
boidObject boid;          /* основной объект boid */
int onGround;             /* идти или лететь */
float annoyance;          /* насколько обеспокоен голубь? */
int framesSinceTakeOff;   /* действительное время полета */
                          /* на такое расстояние*/
int framesBeforeDescent;  /* контролирует примерную длительность полета */
int ascentFrames;         /*контролирует примерную длительность подъема */
} pigeon;
typedef struct boidObject
{
struct boidObject* next;/* для компоновки стаи*/
vecObject position;       /* центральная позиция*/
vecObject side;           /* базисный вектор (x), направленный в сторону*/
vecObject up;             /* базисный вектор (y), направленный вверх*/
vecObject forward;        /* базисный вектор (z), направленный вперед*/
vecObject velocity;       /* вектор текущей скорости*/
vecObject acceleration;   /* вектор текущего ускорения*/
float mass;               /* размеры материальной точки*/
float animationPhase;     /* где замкнутая система анимации*/
lqClientObject lqco;      /* местный запрос proxy*/
int skipCounter;          /* обдумывание пропуска фазы */
} boidObject;
```

Рисунок 3. Структура «основного класса» Boid.

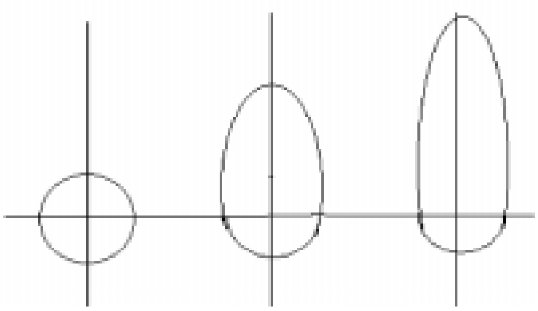
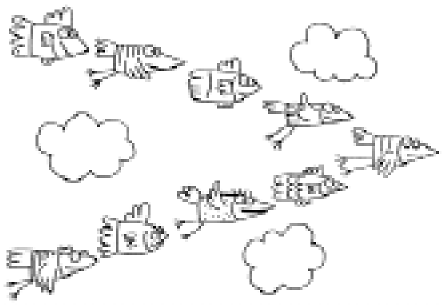


Рисунок 4. Изменение окрестности машинки в зависимости от ее скорости.

Окрестность, олицетворяющая пользователя, которая инициирует желание голубей отойти, больше (и меньше удлиняется с увеличением скорости), чем окрестность, которая инициирует их желание взлететь. Когда голуби находятся в пределах зоны, из которой они стремятся уйти, их поведение – смесь движений, которые уводят их от потенциальной неприятности, затрагивая другие их поведенческие линии (сбор в стаю и уклонение от препятствий).



ПРЕДСТАВЛЕНИЕ В РЕАЛЬНОМ ВРЕМЕНИ

Для моделирования сюжета со многими действующими лицами, такого как «Голуби в парке», с желаемой нормой передачи в 60 кадров в секунду, важно сохранять полную вычислительную стоимость низкой. Имеется три основных источника вычислительной сложности, каждый из которых должен быть сведен к минимуму:

– быстрый вывод на экран изменяющейся обстановки,

– моделирование «мышления» персонажей,

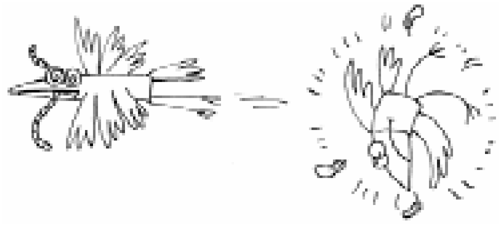
– учет всех возможных взаимодействий между персонажами.

Решение первой задачи достигается за счет относительно скромных требований к сложности изображаемой местности в терминах подсчета числа многоугольников, из которых составляется картинка, и использования простых теневых моделей.

Вторая по значимости вычислительная стоимость у времени, за которое каждый автономный персонаж-птица успевает *подумать*. Как описано выше, «мозг» автономного персонажа – программа, которая должна сработать, чтобы определить, какое действие совершить, основываясь на анализе текущей окружающей среды. Хотя эти вычисления относительно просты, они должны быть проделаны для каждого персонажа, и, таким образом, стоимость умножается на количество персонажей. Про процессы такой сложности говорят, что они имеют сложность $O(n)$. (Чтобы иметь представление об асимптотической сложности, взгляните на [Fren99]).

Поскольку частота обновления экрана 60 Hz довольно быстра, по сравнению с движением персонажа, простой путь снижения стоимости поведенческой модели – проводить поведенческие расчеты на более низком уровне. В программе «Голуби в Парке» достаточно, чтобы голуби «думали» с частотой 10 Hz, то есть одну шестую времени анимации. Таким образом, управляющая программа для данных персонажей работает только на каждой шестом шагу моделирования. Между этими «мыслительными» моментами персонаж будет продолжать совершать одно и то же действие.

В действительности управление меняется постоянно из-за того, что лежащая в основе физическая модель пересчитывает ускорение по отношению к управляющим силам. Также часть «мыслительного процесса» персонажа, связанная с уклонением от препятствий, может осуществляться по более высокой цене (примерно 60 Hz). Эксперименты показали, что предсказание возможного столкновения редко вредит ровному и эффективному уклонению от препятствий. Это разделение позволяет совмещать хороший уровень уклонения от препятствий с довольно низкими требования-



ми к сложности «мышления». (Здравый смысл подсказывает, что для «птичьих мозгов» достаточна довольно скромная вычислительная сила). Наконец, заметим, что каждая птица в основном думает на 10 Hz, фазы этих циклов случайны, то есть примерно одна шестая всей популяции думает в каждом кадре.

Третий и потенциально самый серьезный источник вычислительных проблем – учет взаимодействий между персонажами. Поведенческую модель группы (или любой другой ограниченной в пространстве системы с большим количеством персонажей) можно представить как систему взаимодействующих частиц. Такие системы имеют основное свойство: каждая частица должна взаимодействовать со всеми остальными частицами. В результате система взаимодействия частиц имеет асимптотическую сложность $O(n^2)$. То есть, к примеру, удвоение числа персонажей учетверяет количество времени, затрачиваемого на разрешение вопросов их взаимодействия. Существенно то, что не имеет значения, как быстро каждый вопрос одного взаимодействия может быть решен, с возрастанием популяции стоимость обработки всех взаимодействий, в конечном счете, доминирует над всеми остальными вычислительными стоимостями.

В нашей модели все важные взаимодействия происходят между соседними индивидуумами. Часть базового алгоритма вычисления взаимодействий со сложностью $O(n^2)$ служит только для того, чтобы найти нескольких соседей каждого индивидуума. Эффективный путь к ускорению решения этого вопроса – снабдить персонажей чем-то вроде пространственной структуры данных (spatial database). Программистская цель – удержать персонаж «заранее классифицированным» на основании его положения в пространстве, так чтобы быстро найти, кто находится в соседстве, не проверяя всю популяцию.

В вычислительной геометрии созданы различные типы пространственных структур данных, которые можно использовать для ускорения решения вопросов взаимоположения (например, в [Samet89]). В программе «Голуби в Парке» использована простая техника разделения пространства на ячейки.

В пространственном делении на ячейки часть пространства в форме куба разделена на множество более мелких кубиков. В начале персонажи распределены по ячейкам на основании их начальных позиций. Каждый раз, когда они двигаются, проверяется, перешли ли они в новую ячейку, и, если так, обновляется их принадлежность к ячейке. Большой куб подобран так, что он окружает интересующую нас область (луг в парке). Система из 1000 ячеек $10 \times 10 \times 10$ кажется довольно хорошим началом для экспериментов.

В обсуждаемой программе вопрос близости решается с помощью определения сферы, окружающей данный персонаж. Программа идентифицирует все ячейки, которые хотя бы частично покрываются сферой (рисунок 6). Если объект принадлежит сфере, то его взаимодействие с объектом в центре сферы учитывается. Ячейки сами по себе состоят из единственного указателя на двунаправленный список находящихся в них объектов, который содержит

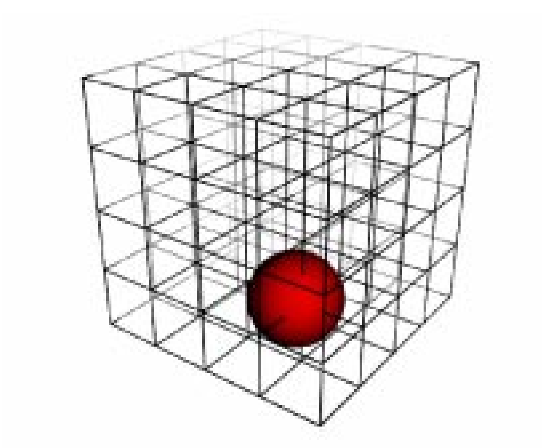


Рисунок 5. Сфера внутри пространства, разбитого на ячейки.

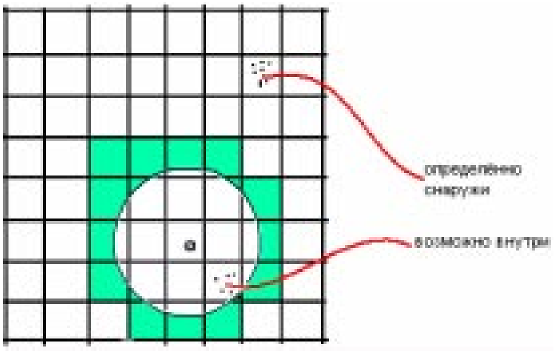


Рисунок 6. Объекты в теневой области могут быть внутри сферы, объекты снаружи теневой области не могут быть внутри сферы.

координаты (x; y; z) и указатель на персонаж (в нашем случае, персонаж – голубь).

Из-за того, что персонажи поддерживают минимальное разделяющее расстояние и таким образом максимальную сплоченность, число персонажей на пространстве данного радиуса ограничено. Обозначим границу k. Таким образом, используя пространственное деление, сложность учета взаимодействий системы частиц (как модели стаи) уменьшается с $O(n^2)$ на $O(nk)$, которая, если мы рассмотрим k как константу, асимптотически эквивалентна $O(n)$. В одном эксперименте с помощью разбиения пространства на ячейки, использующем летящую стаю из 1000 птиц, решение проблемы взаимодействий происходило в 16 раз быстрее, чем в простой реализации, имеющей сложность $O(n^2)$.

Литература

[Fung99] *John Funge*, *Xiaoyuan Tu* and *Demetri Terzopoulos* (1999) «Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters» in Computer Graphics, volume 33 Annual Conference Series (Proceedings of SIGGRAPH 99) pages 29–38.
http://www.dgp.toronto.edu/~funge/sig99/fttsig.pdf

[Gran98] *Steve Grand* and *Dave Cliff* (1998) «Creatures: Entertainment software agents with artificial life» in Autonomous Agents and Multi-Agent Systems, 1(2).

[Lair00] *John E. Laird* (2000) «It Knows What You're Going To Do: Adding Anticipation to a Quakebot» to appear in the AAAI 2000 Spring Symposium Series: Artificial Intelligence and Interactive Entertainment, March 2000.

[Popo94] *Zoran Popovic* (1994) «The Rapid Simulation of Proximal-Interaction Particle Systems» unpublished manuscript. http://sulfuric.graphics.cs.cmu.edu/~zoran/actin/pop.ps

[Reyn87] *Craig W. Reynolds* (1987) «Flocks, Herds, and Schools: A Distributed Behavioral Model» in Computer Graphics 21(4) (SIGGRAPH 87 Conference Proceedings) pages 25–34.
http://www.red.com/cwr/boids.html

[Reyn99] *Craig Reynolds* (1999) «Steering Behaviors For Autonomous Characters» in Conference Proceedings of the 1999 Game Developers Conference, pages 763–782.
http://www.red.com/cwr/steer/

[Nagy99] CDS Documentation, unpublished working paper (CDS.pdf).

[Samet89] *Hanan Samet* (1989) Applications of Spatial Data Structures, Addison-Wesley.

[Fren99] *Michael Frenklach* and *Kal Sastry* (1999), «Chapter 20: Big-O Notation» in course notes to E77N: http://www.me.berkeley.edu/~e77/lecnotes/ch20/ch20.htm

© Наши авторы, 2006.
Our authors, 2006.

Перевод статьи Крейга Рейнольдса (статья приводится в сокращенном изложении) выполнен студеткой III курса математико-механического факультета СПбГУ (специальность «Прикладная информатика в социологии») Екатериной Варфоломеевой.