



Матвеев Алексей Сергеевич @HomoLuden

Пользователь

82,5

карма

0,0

рейтинг



Профиль

24

Публикации

1,3k

Комментарии

1,9k

Избранное

27

Подписчики

28 декабря 2010 в 11:57

## Разработка → Ход «Voronoi»

📁 Алгоритмы\*

### Вместо предисловия

Урок русского языка в ~~грузинской~~ нерусской школе.

Учительница:

— Дети, это нельзя понять, это надо запомнить: ОТ ВАС пишется раздельно, а КВАС — вместе.

Анекдот взят [тут](#).

### Введение

На написание статьи вдохновила игра «Wesnoth» — пошаговая стратегия с элементами RPG. В этой игре персонажи перемещаются по карте, состоящей из шестиугольных полигонов. Таким образом, окруженный со всех сторон персонаж может быть атакован шестью вражескими. По этой причине тактическая составляющая в игре очень важна. Возник вопрос: как повлияет на игровой процесс переход от карты с фиксированной геометрией полигонов на карту с произвольной геометрией?

В данном посте решил написать о возможности использования «диаграммы Вороного» в качестве случайной игровой карты. Описаний алгоритмов для построения диаграммы Вороного в сети нашел несколько. Но все, что довелось прочитать мне, либо слишком поверхностны, либо написаны академическим языком. В общем, описания, пригодного для немедленной реализации алгоритма я не нашел. Пришлось начинать реализацию не имея четкого понимания всех тонкостей алгоритма. Нашел в сети исходники на разных языках, но для перевода на нужный мне язык и адаптации под мою задачу все равно не хватало знаний. Поэтому решил изобретать велосипед «с чуть более круглыми колесами и с седушкой помягше».

Возможно кто-то из читателей встречал описание подобной задачи с исходными кодами. Тогда оставьте, пожалуйста, в комментариях ссылку — сравним с тем, что получилось у меня.

### Алгоритм

Итак... серия «на пальцах» продолжается. Вы можете почитать по-русски про диаграмму Вороного в википедии [1]. Суть такова, что для случайного набора точек (центров) на плоскости (я для простоты рассматриваю 2D-случай) эта диаграмма представляет собой совокупность таких полигонов, что все точки внутри полигона находятся ближе к его «центру», чем к «центрам» других полигонов. Соответственно, все точки на границе между двумя полигонами располагаются на одинаковых расстояниях от обоих центров, а вершины полигонов равноудалены сразу от трех или даже более центров.

Самым «простым» способом построения диаграммы является построение «серединных перпендикуляров» между всеми центрами, т.е. перпендикуляров к отрезкам, соединяющим пары центров, размещенных в середине этих отрезков. Затем нужно найти точки пересечения полученных перпендикуляров и произвести отсечения. Однако вычислений при это нужно будет сделать больше всего ( $O(n^2 * \log n)$ ).

Есть алгоритм т.н. «заметаящей прямой» (sweep line, fortune's algorithm [2]). Его вычислительная сложность  $O(n \log n)$ . Он и будет описан ниже.

### Суть алгоритма

В основе этого алгоритма лежит применение вспомогательного объекта — заметаящей прямой (ЗП). Она может быть вертикальной или горизонтальной. Я использую горизонтальную ЗМ, движущуюся от больших значений Y к меньшим.

Суть в том, что при каждом положении ЗП рассматриваются лишь точки выше этой линии и непосредственно на ней. При этом для каждого из «центрОв» строится парабола, точки на которой равноудалены от «центра» и от ЗП. В данном случае «центры» являются **фокусами** соответствующих им парабол, а ЗП — **директриса** этих же парабол.

Уравнение параболы имеет вид:

$$y = ((x - xf)^2 + yf^2 - L^2) / (2 * (yf - L))$$

где **xf**, **yf** — координаты фокуса параболы (центр полигона Вороного);

**L** — положение ЗП (координата Y текущего обрабатываемого события).

Если построить огибающую снизу всех парабол, то получится т.н. «береговая линия» (beach line). Эта кусочная кривая играет

ключевую роль в алгоритме. Точки пересечения «кусков парабол» (назовем их брейкпоинтами) лежат на границах полигонов диаграммы. Когда сходятся в одну точку два брейкпоинта, то одна из «арок» (кусочек одной из парабол) «схлопывается», т.е. две соседние к ней арки соединяются друг с другом. При этом образуется вершина полигона диаграммы.

Таким образом, задача сводится к детектированию и обработке двух событий — добавления в список рассматриваемых центров новой точки (site event) и подозрение на наличие вершины (circle event) [3]. И так далее... примерно такое описалово я и находил в сети на разных языках. И на данном, концептуальном, уровне все выглядит не очень сложно. Но как именно (по шагам) осуществляется детектирование? И что именно делать с этими событиями?

Попробую изложить простыми словами.

### Структура данных

Как было сказано выше, в алгоритме ключевыми объектами являются события: событие точки (site event) и событие круга (circle event). Эти события помещаются в список, который сортируется в моем случае по убыванию координаты Y, т.е. события имеющие бОльшие значения координаты Y размещаются выше в очереди и обрабатываются раньше. Таким образом, первое, что нам понадобится — упорядоченный список для хранения событий.

Второе, что нам понадобится — бинарное дерево. В этом дереве будут размещены узлы трех типов: Arc — арка (часть параболы), этот узел должен хранить координаты фокуса параболы и ссылку на событие круга, если таковое имеется; BP — breakpoint, точка пересечения двух парабол; BPOwner — корень поддерева, его детьми являются узлы типа BP. Узлы BP и BPOwner должны хранить ссылку на грань. Нужно **обязательно** хранить грани в отдельном списке, т.к. узлы BP и BPOwner будут удаляться при обработке событий круга.

Соответствие узлам Arc и BP вы легко найдете в публикации [6]. BPOwner прямого наименования не имеет — просто корень поддерева. Я дал ему наименование лишь для удобства программной реализации.

Типичная схема бинарного дерева для диаграммы Вороного (как я ее себе представляю) изображена на рис. 1.

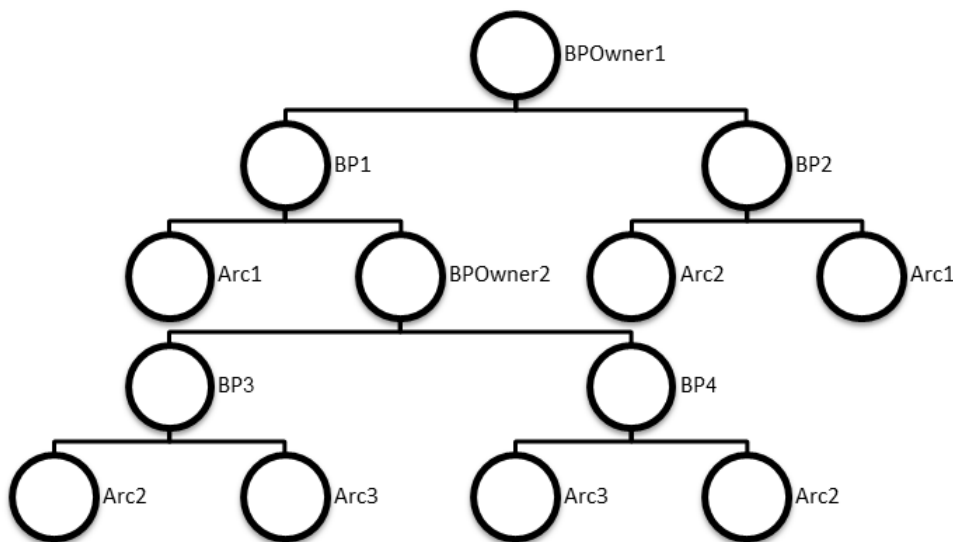


Рис. 1. Бинарное дерево.

На этом рисунке сразу можно выделить потенциально имеющееся событие круга. Если смотреть слева, то Arc1, Arc2, Arc3 образуют тройку фокусов, которые с большой вероятностью не будут лежать на одной прямой. Больше событию круга возникнуть негде, т.к. остаются тройки {Arc3 Arc3 Arc2}, {Arc3 Arc2 Arc2} и {Arc2 Arc2 Arc1}. Такая схема построения дерева пусть и избыточна, но проста в анализе.

Подробнее структуру данных можно изучить в [6]. Возможно мое изложение отличается от представленного в той публикации, но для меня так проще.

### Главный цикл

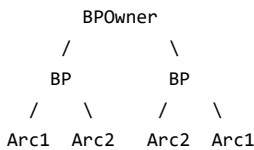
1. Инициализация данных.
2. Пока очередь НЕ ПУСТА:
  1. Вырезать из очереди первое событие (с наибольшим значением координаты Y)
  2. If(event == Site Event)
    - ProcessSite(event)
  3. Else
    - ProcessCircle(event)
3. Финишировать все грани, ссылки на которые имеются в бинарном дереве.

Действительно просто. Вся магия в обработчиках событий.

### Обработка «события точки»

Что такое событие точки для нашего бинарного дерева? Когда ЗП попадает на очередную точку, то в дерево добавляется новая

парабола. Нужно заметить, что, во-первых, по алгоритму ЗП перемещается от события к событию (в демо-анимациях к алгоритму эта прямая движется плавно), а во-вторых, изначально парабола представляет собой вертикально направленный луч. В месте пересечения этого луча с одной из парабол образуется сразу два «брейкпоинта» (BP). Таким образом, для каждого события точки в дереве находится парабола с которой произойдет пересечение соответствующего луча. Сделать это легко методом спуска, начиная с корня всего дерева и сравнивая координаты X этого события и узлов BPOwner и BP. Спуск происходит до тех пор пока не встретится арка. Если арка содержит ссылку на существующее в очереди событие круга, то нужно удалить это событие из очереди и удалить ссылки на это событие из данной арки и ее соседей, арки справа и слева. Далее вместо арки (см. Arc1 ниже) создается поддерево вида:



Теперь, когда структура дерева поменялась, следует пробежать по дереву слева направо в поисках событий круга. Для этого нужно брать последовательно тройки арок и проверять координаты фокусов соответствующих им парабол на коллинеарность. Если три фокуса не лежат на одной прямой, то можно построить общую для них окружность. Если нижняя точка этой окружности лежит ниже ЗП, то можно добавить в очередь событие круга. Нижняя точка окружности — это место, где событие произойдет, а центр — это точка, где будет находиться вершина полигона Вороного.

Подводя итог скажу, что события точек только добавляют узлы в дерево и события круга в очередь событий. Больше ничего при их возникновении не делается. При наступлении события точки даже не производится вычисление координат BP — точек пересечения парабол. «Главная магия» делается в обработчике событий круга.

### Обработка «события круга»

Каждое событие круга должно содержать ссылку на арку, которая будет удалена («схлопнется», см. выше) при наступлении этого события. Ранее было написано, что для детектирования события круга рассматриваются тройки арок. Так вот, в событии круга должна храниться ссылка на среднюю арку, т.е. на ту, у которой значение координаты X находится между соответствующими значениями двух других арок. В этих соседних арках также должна храниться ссылка на то же событие круга, что и у средней. Хотя особой необходимости я в этом не вижу. Мне, как «на-палечнику», проще записать в среднюю арку ссылки на две соседние арки — левую и правую. Благодаря этому не придется прогуливаться по дереву в поисках соседней ветки с нужной аркой (обязательно одна из оставшихся двух арок будет в соседней ветке, т.к. дерево бинарное).

Обработка события круга будет заключаться в обновлении координат брейкпоинтов, непосредственно ограничивающих удаляемую арку. Далее производится удаление арки и двух, связанных с ней брейкпоинтов. После этого нужно перестроить дерево. В результате будет добавлен новый узел BP, а получившееся поддерево переместится на уровень выше...

Текстом такие процедуры воспринимаются достаточно трудно. Во всяком случае для меня, как человека без серьезных знаний в теории графов и бинарных деревьев, это было нетривиальной задачей, поэтому привожу еще и иллюстрацию на рис. 2.

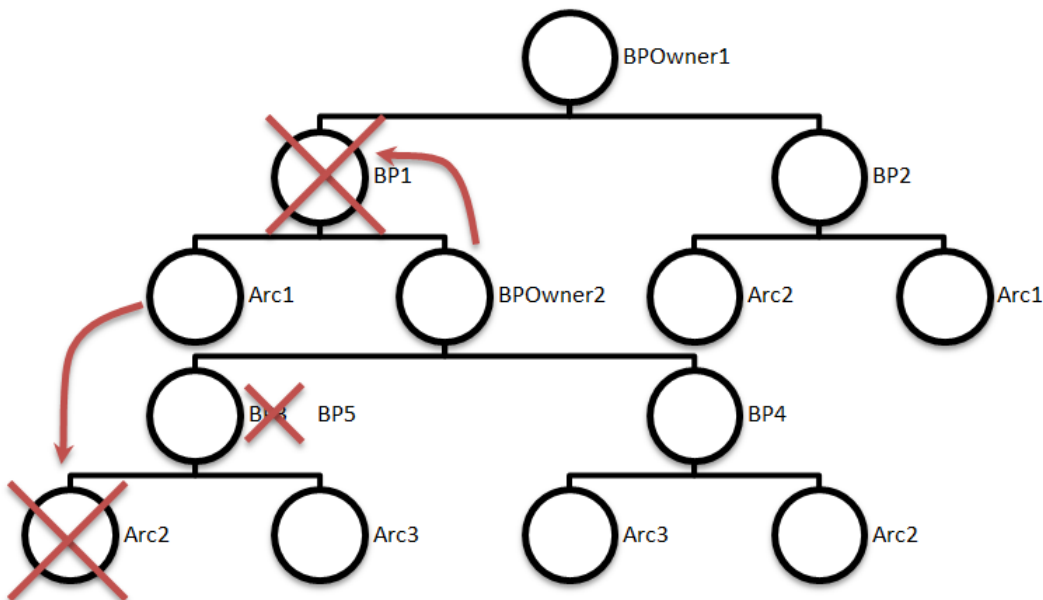


Рис. 2 Обработка события круга

### Результат

Здесь я приведу характерный вид диаграммы, т.е. вид игровой карты с произвольной полигональной геометрией.

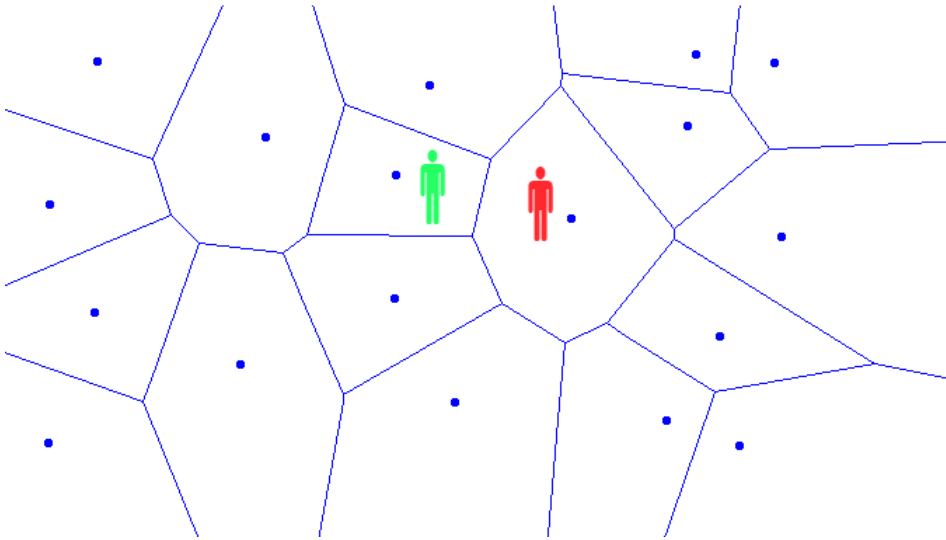


Рис.3 Игровая карта

На этой карте в «центрАх» (напомню — это исходный набор случайно распределенных по площади точек) могут находиться персонажи. Через грани этих полигонов на персонажа может напасть персонаж противника. На представленном выше рисунке «зеленый человечик» обозначает... нет не алкоголизм. Полигон, в котором размещен силуэт зеленого цвета — безопасная локация с точки зрения количества возможных атак за ход противника (4 максимум). А вот персонажу красного цвета не повезло. На него может быть совершено максимум 8 нападений со стороны недружественного населения.

## Тактика

Каждому полигону на карте можно присвоить «стоимость прохождения». Можно поступить иначе — назначить «стоимость пересечения» границ полигонов или использовать комбинацию того и другого (стоимость пересечения границы + половина стоимости прохождения). Также, в зависимости от площади полигона (или количества охватываемых точек координатной сетки карты), можно назначить различные бонусы (периодичность рождения артефактов, происхождения каких-либо событий и пр.), очки защиты, ресурсы постройки укреплений и т.д. Это уже предоставляет большие тактические возможности. Плюс к этому следует правильно выбирать полигон, в котором можно остановиться. Как написано в начале, большой полигон может иметь большее количество границ, а значит и большее число нападений за один ход противника. Однако большой полигон — это большие бонусы (при условии назначения бонусов в зависимости от площади, а не от количества границ). Значит особо ценными будут полигоны с максимальной площадью при минимальном количестве границ. Вот такая задача «МаксиМин», которую нетрудно автоматизировать. Труднее автоматизировать успешное использование получаемых бонусов для ИИ.

Поиск пути можно осуществлять с использованием  $A^*$  («TODO» для отдельного поста). Можно ввести систему прокладки путей — сначала проводится персонаж вручную, делается запись пути, а потом можно использовать созданные «waypoint'ы» для автоматизации. Но это будет скорее всего слишком неудобно. На геймплее может сказаться негативно.

Теперь об укреплениях. У каждого полигона должен быть некий набор ресурсов для создания укреплений, от которых будет зависеть тип фортификации и скорость возведения сооружений. Прямо «Цивилизация» получается. На укрепление в выбранной позиции персонаж тратит очки действий. Их можно разделить на две группы — защитные и атакующие. При этом защитные ОД тратятся на постройку укреплений и оборонительную активность во время атаки противника (поднять щит, увернуться от стрелы, заблокировать атаку и пр.), а атакующие могут быть потрачены как в стадии атаки, так и при защите (контратака, обезоруживание). Тут можно упомянуть ZPG «MyBrute», в которой аватарчик может при защите перехватить инициативы и атаковать в ответ, не получив удара от противника.

Каждый персонаж может иметь набор укреплений, постройке которых он обучен. Каждый тип укреплений дает разную степень защищенности и/или бонусы. Помимо бонусов от укреплений, сам полигон может давать случайные бонусы. Пребывая долгое время в данном месте, персонаж увеличивает вероятность найти что-то ценное или «открыть» некое скрытое свойство данного полигона.

Можно и дальше продолжать такие рассуждения. Пространства для маневров достаточно, но тогда мы отклонимся от темы статьи.

## Выводы

Смена геометрии игровой карты неминуемо приведет к серьезным изменениям (думаю в лучшую сторону) в тактике игры. Появится дополнительный тактический уровень, возможность при правильном подходе в переломный момент битвы перевесить чашу весов в свою сторону. К тому же, такой подход позволит сделать игровую карту динамической — видоизменять ее топологию (локально или глобально). Тут нельзя не вспомнить не реализованную до конца (на мой взгляд) задумку изменяемых локаций в играх «S.T.A.L.K.E.R.». В 2D это реализовать не очень сложно. Но не забываем о том, что диаграмма Вороного строится и для 3D. Спасибо за внимание! Ваши комментарии...

## Список литературы

1. [Диаграмма Вороного](#)
2. [Voronoi Diagrams and a Day at the Beach](#)
3. [Planar Voronoi Diagrams via Fortune's Algorithm](#)

- 4. Polygonal Map Generation
- 5. Voronoi diagrams
- 6. Fortune Sweep Algorithm

игры, алгоритмы, Voronoi, случайная, карта, генерация

↑ +50 ↓


👁 16,8k ⭐ 86

🐦

📧

📌

📧



Матвеев Алексей Сергеевич @HomoLuden

82,5 0,0

Пользователь

Похожие публикации

- +29

Генерация псевдослучайных чисел

👁 53,3k

⭐ 125

💬 36
- +157

Сравнение алгоритмов поиска маршрутов в StarCraft и StarCraft 2

👁 10,3k

⭐ 66

💬 110
- +59

Q4wine — установка Windows-игры в Wine в картинках

👁 16,6k

⭐ 78

💬 51

Реклама помогает поддерживать и развивать наши сервисы

Подробнее

Спецпроект

Самое читаемое				Разработка
Сейчас	Сутки	Неделя	Месяц	
+13	Одна простенькая задачка. Быстро, красиво или чисто?			
👁 9,9k	⭐ 60	💬 27		
+20	Парадокс Rimworld: захватывающая сюжетом «песочница»			
👁 2,4k	⭐ 20	💬 8		
+11	Объясняем бабушке: Как зашифроваться за час			
👁 4,5k	⭐ 64	💬 22		
+26	Log in или Log on? Front-end или Frontend? Продолжаем разбираться			
👁 2,5k	⭐ 27	💬 16		
+9	Как я добавил 30000 человек в первый круг контактов, а эту соцсеть заблокируют в РФ			
👁 2,4k	⭐ 13	💬 6		

Комментарии (21)

X

miXe14

28 декабря 2010 в 12:38


#

+2

↑

↓

Слова «вилка» и «тарелка» пишутся без мягкого знака, а слова «сол» и «фасол» — с мягким.



HomoLuden

28 декабря 2010 в 12:45

#

h

↑

0

↑

↓

Я хотел именно эту интерпретацию идеи опубликовать, но поиск выдал другой вариант. Суть та же — некоторые вещи просто один раз заимплементировать и забыть.



Lopar 28 декабря 2010 в 14:30 # h i

+4 ↑ ↓

Но самый сложный русский слово — это слово ГРУСТ. Это не тот ГРУСТ, который в кошёлка собирают — это тот ГРУСТ, который ТОСКА. Это не тот ТОСКА который заборы делают — это тот ТОСКА который ГРУСТ.

(орфография анекдота сохранена)



wishope 28 декабря 2010 в 13:52 #

+6 ↑ ↓

Не так давно тоже занимался построением диаграммы Вороного на плоскости.

Стоит заметить, что диаграмма Вороного двойственна триангуляции Делоне, для которой построения которой существует более просто, чем sweeping line алгоритм, при этом работающий также за  $O(N \log N)$ .



Hizof 28 декабря 2010 в 17:54 # h i

0 ↑ ↓

Можете привести названия алгоритмов? И может быть встречали описание алгоритма sweeping line для 3D пространства? Попытки обобщить для n-мерного случая не увенчались успехом. Я для поиска Делоне в пространстве использую выпуклую оболочку.

НЛО прилетело и опубликовало эту надпись здесь



HomoLuden 30 декабря 2010 в 13:00 # h i

0 ↑ ↓

Вот нашел описание [триангуляции Делоне](#) на русском. Там заявлена сложность  $O(n^2)$ . После этого потребуется еще потратить время на переход к диаграмме Вороного.

Вопрос к читателям, хорошо знающих теорию оценки вычислительной сложности. Одна и та же оценка ( $O(n \log n)$ ), но для разных алгоритмов может давать разное время работы алгоритмов? Количество операций приблизительно одинаково, но операции разные... Будет ли различие значительным?



GraD\_Kh 3 января 2011 в 00:53 # h i

0 ↑ ↓

Если распределение точек не случайное, а такое что соседние точки расположены по близким индексам, то написать алгоритм, справляющийся за  $O(n \log n)$  — несложно, сам такой реализовывал. В противном случае нужно применять динамическое кеширование, для итеративных алгоритмов, например.



bagyr 28 декабря 2010 в 14:15 #

0 ↑ ↓

Чаще на случайных точках строят связный граф, а на вершинах-ребрах уже что-то делают. Все остальное тогда нужно для красоты, не больше. Пример с вестнотом тоже не от туда, на описанной механике базируясь серии Total War и Dominions.

Открытые реализации диаграмм Вороного есть, из того, что видел сам, в XScreensaver, MATLAB и Mathematica.



Hizof 28 декабря 2010 в 17:58 # h i

0 ↑ ↓

Ещё про алгоритмы построения двойственной диаграммы можно почитать в «Триангуляция Делоне и ее применение» Сковцов А.В.



Leezarius 28 декабря 2010 в 14:35 #

+23 ↑ ↓

Горе от ума.

Суть любой ролевой системы — скелета ролевой игры это умное упрощение. Главная задача построить игровую модель максимально простую и при этом максимально четко описывающую игровой мир и взаимодействия.

В молодости из максимализма мы меняли простые правила ролевой системы на физические модели потому что так правильнее, как нам казалось. Вместо того чтобы одним броском кубика выяснить попал ты в противника или нет мы строили траектории снарядов, вектора рикошета и другие очень нужные, как мы думали, вещи.

Что в итоге? В итоге ничего хорошего. Игра интереснее не становится зато становится на порядок сложнее и монструознее. Хорошие ролевые системы гениальны в своей простоте, это начинаешь ценить после реализации каких-то RPG проектов. Усложняя систему вы не делаете ее лучше вы всего лишь делаете ее сложнее.

К чему я это все?

К тому что шестиугольники в РПГ используются не из-за возможных 6 противников а по одной простой причине — **расстояния между центрами любых соседних гексагонов одинаковы**. Это единственное преимущество гексагональной карты перед обычной прямоугольной сеткой, где расстояние диагональных клеток отличается от расстояния между центрами горизонтальных и вертикальных клеток. К тому же расстояния между диагональными клетками не целое число и это многое усложняет.

То есть единственное обоснование существования гексагональной карты это простота расчета пути персонажей перемещающихся по гексам. Все.

Ваша модель рушит весь смысл гексов, по вашим полигонам невозможно перемещаться равномерно, следовательно придется переходить к перемещению по отдельной сетке координат, тогда ваша сетка полигонов будет иметь некоторую стратегически-декоративную роль, но эта роль не будет описывать никаких реальных моделей. В итоге система усложняется вводится ряд условностей которые невозможно объяснить и ради чего все это происходит понять сложно.

P.S. Очень радует подход с которым вы относитесь к играм, почитайте про PnP системы, там очень много интересного по данной специфике, отсюда растут корни современных RPG.



HomoLuden 28 декабря 2010 в 15:20 # h i

0 ↑ ↓



| простота расчета пути персонажей перемещающихся по гексам.

Видел описание двух алгоритмов просчета гексагональной системы координат (с 2мя осями и 3мя). Лично мне не показались эти алгоритмы уж больно простыми. В моем случае сетка подразумевается прямоугольной. Просто не во всех точках этой сетки может быть помещен персонаж. Зато нет необходимости преобразовывать прямоугольные координаты в гексогональные.

по вашим полигонам невозможно перемещаться равномерно, следовательно придется переходить к перемещению по отдельной сетке координат, тогда ваша сетка полигонов будет иметь некоторую стратегически-декоративную роль, но эта роль не будет описывать никаких реальных моделей.

Что значит равномерно? Алгоритмы поиска пути для полигональной карты описаны. Есть список возможных позиций с их координатами, есть текущая позиция, есть целевая. Требуется найти путь. В чем конкретный недостаток такого подхода? Я и писал, что меняется тактика ведения боя.

Почему игровой алгоритм должен описывать реальную модель («реальная модель»  $\Leftrightarrow$  оксюморон — «горячий лед»)?

 Leezarius 28 декабря 2010 в 17:21 

+4  

Видел описание двух алгоритмов просчета гексагональной системы координат (с 2мя осями и 3мя). Лично мне не показались эти алгоритмы уж больно простыми. В моем случае сетка подразумевается прямоугольной. Просто не во всех точках этой сетки может быть помещен персонаж. Зато нет необходимости преобразовывать прямоугольные координаты в гексогональные.

Мы с вами о разной простоте говорим. Клетка в RPG это мера расстояния.

«Герой продвинулся на 3 клетки вперед» — типичный отыгрыш передвижения. Пока в прямоугольной сетке герой перемещается исключая диагональные переходы, все просто, ходим по клеткам, считаем их, как кончились шаги — встали. Красота.

Когда мы добавляем переход по диагоналям, красота пропадает. Шаги начинают дробиться, расчет усложняется. То что вы называете алгоритмами просчета к RPG никакого отношения не имеет, это всего лишь алгоритмы визуализации.

Ваша беда в том что вы пока не разделяете ролевую систему и вид игры. Это примерно как в MVC, RPG — контроллер, а алгоритм просчета гексагональной системы это вид, его вы можете усложнять или упрощать, это никак не повлияет на игровую составляющую. Например вы можете использовать какой-нибудь простой и очень примерный алгоритм пересчета координат и ваши герои будут позиционироваться внутри гексов неровно или можете ввести суперточный алгоритм и при этом добавить еще эффектов перемещения, на игру это никак не повлияет, лишь будет смотреться чуть топорнее или чуть лучше.



Что значит равномерно? Алгоритмы поиска пути для полигональной карты описаны. Есть список возможных позиций с их координатами, есть текущая позиция, есть целевая. Требуется найти путь. В чем конкретный недостаток такого подхода?

Равномерно значит если герой прошел 5 клеток длина его пути равна пяти клеткам. Вы возможно не понимаете, но на гексагональной карте герои перемещаются исключительно по клеткам, плавное передвижение от клетки к клетке на игру никак не влияет, это всего лишь украшение. Вы взяли гексагональную карту которая является неотъемлемым механизмом игры и отвечает за перемещение, убрали основное преимущество этого метода да и собственно прямое назначение и создали свою надстройку, в силу своего понимания процесса игры. Но при этом я вот лично не понимаю почему герой должен находится в центре вашей клетки и как он теперь должен двигаться? На него можно напасть пока он не дошел до центра или надо ждать пока дойдет? А вдруг он передумает? Может на меня напасть противник если он не в центре соседней клетки... и много других вопросов в результате усложнения концепции.

Почему игровой алгоритм должен описывать реальную модель («реальная модель»  $\Leftrightarrow$  оксюморон — «горячий лед»)?

Потому что классическая РПГ именно этим и занимается, что естественно вовсе не означает что не может быть других оригинальных подходов. «Реальная модель» в отличии от вашей «искусственной модели» интуитивно понятна — герой ходит по клеткам которые одновременно может занимать только один субъект, если рядом с героем кто-то стоит, с ним можно взаимодействовать и т.д. Ваша модель искусственна, вы должны объяснить игроку почему он перемещается внутри полигона, что это за полигон, почему он такой, почему противник может напасть только с одной стороны полигона, почему в одном полигоне помещается только один герой и тд.

Вы можете перевести вашу игру на другой уровень и сделать вашу модель «реальной», то есть интуитивно понятной упрощенной моделью реальной жизни. Например ваши полигоны могут быть королевствами, тогда становится понятно почему герой перемещается внутри полигона, понятно почему один противник нападает с одной стороны, но это уже немного другой жанр.

 HomoLuden 28 декабря 2010 в 17:55 

0  

Но при этом я вот лично не понимаю почему герой должен находится в центре вашей клетки и как он теперь должен двигаться? На него можно напасть пока он не дошел до центра или надо ждать пока дойдет? А вдруг он передумает? Может на меня напасть противник если он не в центре соседней клетки...

С точки зрения позиционирования внутри полигона мало что меняется. В гексагоне персонаж находится в центре. Здесь под «центрАми» полигонов понимается те случайные точки, что изображены на диаграмме. Рисунок 3 неудачен в плане понимания. Позиционирование также дискретное. Хватает очков действий для перемещения в выбранный полигон — персонаж туда попадает. Не хватает — остается в соседнем. Режим пошаговый. Сначала один игрок ходит потом другой. Поэтому такое разбиение карты не только декоративное. Это как раз и получается структура на графах.

У текущего полигона есть соседи. Заданы некие связи между ними.

Равномерно значит если герой прошел 5 клеток длина его пути равна пяти клеткам. Вы возможно не понимаете, но на гексагональной карте герои перемещаются исключительно по клеткам, плавное передвижение от клетки к клетке на игру никак не влияет, это всего лишь украшение.

А про плавное перемещение Вы заговорили. Вот я и не понял к чему оно в пошаговой игре. Итак понятно, что все анимации к логическому быкграунду имеют опосредованное отношение.

Про расстояние... «Центры» распределяются по прямоугольной сетке. То, что по диагонали расстояние визуально отличается от двух переходов «по катетам», при выбранной схеме генерации топологии будет иметь меньшее значение, т.к. нет четко выраженных форм



полигонов (квадратов, гексов и/или т.д.). И почему на расстояние завязываться. Ту же функцию играет стоимость пересечения каждого конкретного полигона.

НЛО прилетело и опубликовало эту надпись здесь

**HomoLuden**

28 декабря 2010 в 18:25 # h ↑

0 ↑ ↓

Согласен... Но я и не делаю упор на РПГ. Я вел разговор о влиянии на тактическую часть.

**diomas**

28 декабря 2010 в 23:12 #

0 ↑ ↓

Здесь очень интересно написано про генерацию игровой карты с помощью разбиений Вороного: [www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/](http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/)

**HomoLuden**

29 декабря 2010 в 08:29 # h ↑

0 ↑ ↓

Было изначально в списке литературы

**diomas**

29 декабря 2010 в 13:26 # h ↑

+1 ↑ ↓

ой. хреновый я читатель...

**PavelSandovin**

12 января 2011 в 12:22 #

0 ↑ ↓

Изображенная на иллюстрации карта асимметрична. Возникает вопрос: каким образом при организации игры на такой карте обеспечить начальный паритет игроков?

**HomoLuden**

12 января 2011 в 16:12 # h ↑

0 ↑ ↓

Чтобы избежать изначального окружения одного игрока другими, можно использовать шаблон размещения, предопределив заранее некоторое количество точек. Можно привязать базы к углам, можно к серединам сторон карты и т.д.

Только зарегистрированные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

## Интересные публикации



Новые-старые форматы: HD-винил и DIY-пластинки 2



Делаем стартап просто и технологично. Маячки Eddystone 0



Производительность межпроцессного обмена сообщениями в node.js 0



СМАРТ ТВ – будущее телевидения 8



Выпуск Rust 1.13 4