УДК 519.711.3
ББК 32.965.9

Сергей Олегович Алатарцев,
магистр,
университет им. Отто фон Герике
(Магдебург, Германия), e-mail: alatartsev@gmail.com
Дмитрий Андреевич Макаров,
кандидат технических наук,
Забайкальский государственный университет
(Чита, Россия), e-mail: Rainmaker@chita.ru

О некоторых особенностях алгоритма организации искусственного роя в пространстве

В данной статье рассматриваются вопросы моделирования искусственного роя, дается анализ работы популярного алгоритма SHAPEBUGS, предлагаются модификации алгоритма, направленные на повышение его эффективности и производительности в задачах самовосстановления трехмерных форм.

Ключевые слова: самовосстанавливающиеся формы, искусственный рой, алгоритм SHAPEBUGS.

Sergey Olegovich Alatartsev
Master's Program Student,
Otto von Guericke University
(Magdeburg, Germany), e-mail: alatartsev@gmail.com
Dmitry Andreevich Makarov,
Candidate of Technical Sciences, Trans-Baikal State University
(Chita, Russia), e-mail: Rainmaker@chita.ru

About Some Features of the Algorithm of Artificial Swarm Organization in Space

The paper describes the problems of modeling artificial swarm, analyzes the work of the popular SHAPEBUGS algorithm. It suggests algorithm modifications to increase its efficiency and performance in the issues of self-repairing three-dimensional forms.

Keywords: self-repairing forms, artificial swarm, SHAPEBUGS algorithm.

Модель т. н. «искусственного роя»(swarm) представляет собой один из наиболее интересных примеров самоорганизующихся систем – систем, элементы которых, взаимодействуя друг с другом, создают, изменяют или поддерживают в неизменном состоянии саму систему без внешнего воздействия на неё. В основе данной модели лежат попытки формализовать и искусственно воссоздать некоторые процессы самоорганизации, протекающие в коллективных сообществах в живой природе (яркий пример – муравейник).

Формы, созданные в пространстве с применением концепции искусственного роя, характеризуются высокой устойчивостью к повреждениям, так как неповрежденные элементы формы способны, организуясь, быстро восстанавливать её исходное состояние, что делает применение таких форм крайне актуальным во многих практических задачах. Более того, возможна такая самоорганизация объектов роя, при которой они, взаимодействуя между собой, могут собрать из себя другой объект с качествами, превосходящими сумму качеств его составных частей. Так, например, рой в форме колеса способен быстро преодолевать большие расстояния, нежели каждый участник роя по отдельности.

Вместе с тем для выполнения подобной самоорганизации элементов роя требуется решить множество проблем, связанных как с описанием форм, составленных объектами роя, так и со взаимодействием собственно объектов внутри роя.

В настоящее время исследование и разработка алгоритмов для искусственных роев ведутся научными группами в ведущих университетах мира. Фактически, все теоретические исследования

в той или иной мере сводятся к попыткам организации элементов роя в пространстве. Подходы к организации меняются в зависимости от необходимых функций, возложенных на систему.

Можно выделить три уровня в алгоритмах структурирования роя в соответствии с рис. 1.

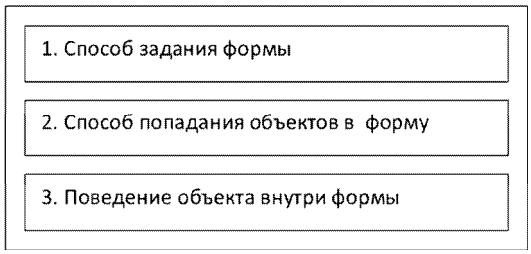

Рис. 1. Уровни в алгоритмах структурирования роя

Каждый уровень не зависит от работы других уровней, поэтому их можно заменять без потери работоспособности всего алгоритма. В данной работе делается попытка подобрать такие подходы для каждого отдельного уровня, чтобы система могла свободно функционировать в трёхмерном пространстве, не требовала избыточно много памяти для хранения формы и обладала высокой скоростью работы.

Цель настоящей работы состоит в том, чтобы предложить решение, расширяющее возможности одного из существующих алгоритмов организации искусственного роя и осуществить программную реализацию этого решения.

Рассмотрим один из наиболее популярных алгоритмов организации искусственного роя – SHAPEBUGS [4; 5]. При работе этого алгоритма поведение объектов, составляющих некую заданную форму, напоминает поведение молекул газа в закрытом контейнере. Так, при сближении объектов на расстояние, определяемое т.н. зоной отталкивания, происходит их отталкивание друг от друга вплоть до достижения нейтральной зоны, как показано на рис. 2.

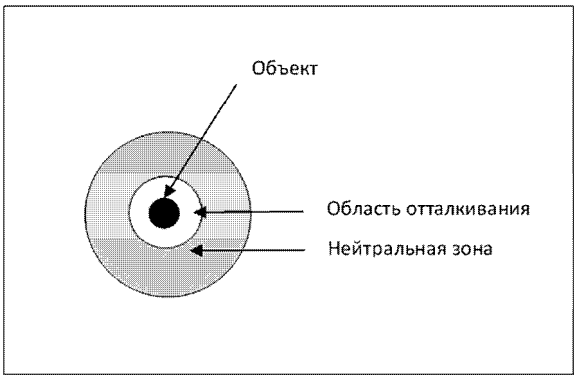

Рис. 2. Области влияния объекта согласно алгоритму SHAPEBUGS

Объект по-разному реагирует на различную концентрацию соседних объектов и пытается выйти из зоны «с высоким давлением»в зону «с низким давлением»подобно молекулам газа в реальной физической среде. Благодаря этому при последовательном выполнении алгоритма «давление»во всех точках контейнера (форме) становится одинаковым, а распределение объектов в форме – равномерным. За счёт этого свойства алгоритма SHAPEBUGS обеспечивается, в частности, восстановление исходной формы в случае изменения числа объектов, которые её составляют.

Задание формы в SHAPEBUGS выполняется в виде трафарета – однобитового растрового *bitmap*-изображения, в котором 0 соответствует пустому пространству, а 1 соответствует форме. Например, чтобы задать двумерную форму дельфина можно белыми точками (нулями) обозначить пустую область, а чёрными (единицами) задать саму форму, как показано на рис. 3.

Несмотря на простоту задания трафарета, данный подход в некоторых ситуациях весьма неудобен и предполагает значительные накладные расходы. В частности, при задании форм, соответствующих сущностям реального мира, необходимо определять формы в трёхмерном пространстве.

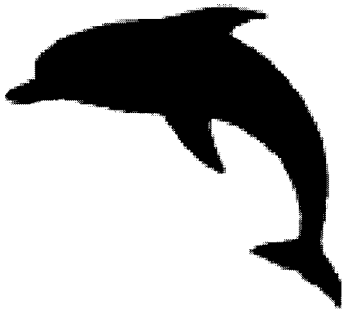

Рис. 3. Пример трафарета

В этом случае задание трафаретом будет достаточно трудоёмким, так как количество трафаретов увеличится с одного до некоторого N, где под N подразумевается минимальное количество заполненных ячеек для третьего измерения.

Некоторым развитием метода задания формы трафаретом является метод использования градиентных карт [6]. Градиентная карта представляет собой массив элементов, каждая ячейка которого хранит целое число и соответствует пикселю на трафарете, задающем форму. Для расчёта элементов градиентной карты используется т.н. Манхэттенское расстояние или расстояние городских кварталов, которое является средним разностей координат заданной точки и точки начала сегмента или формы. Как показано в [3], в большинстве случаев эта мера расстояния приводит к таким же результатам, что и при вычислении обычного евклидового расстояния, позволяя тем самым упростить вычисление расстояний между объектами в системе. Для указания принадлежности элемента градиентной карты к задаваемой форме Манхэттенское расстояние принимается положительным, а для указания того, что точка не входит в задаваемую форму, соответствующий элемент градиентной карты принимается отрицательным.

Хранение таблицы и постоянное обращение к памяти является весьма затратным с точки зрения потребляемых ресурсов, не говоря уже о вычислении самой градиентной карты. Например, для трёхмерного случая куба со стороной, равной 1000, максимальное число в ячейке составит $|1-1000|+|1-1000|+|1-1000|=2997$. Для его описания потребуется 12 бит ($2^{12}=4096$), а для описания всей трёхмерной таблицы понадобится $1000\times1000\times1000\times12$ бит, что составляет порядка 1,4 гигабайт и требует соответствующего объёма памяти в вычислительной системе. Таким образом, задание трёхмерной формы посредством градиентной карты в общем случае представляется весьма затруднительным, так как постоянная работа и поиск в объеме памяти, измеряемом гигабайтами, является достаточно медленным и нецелесообразным процессом. В этой связи применение метода градиентных карт на практике ограничивается заданием двумерных форм, либо трёхмерных форм очень малых размеров.

В силу указанных обстоятельств, представляется целесообразным выполнить определённую модификацию алгоритма SHAPEBUGS в целях повышения эффективности и расширения области его применения. Данная модификация, соответственно, должна быть направлена на устранение тех недостатков алгоритма, которые в наибольшей степени отражаются на производительности и эффективности. С этой целью предлагается модернизировать работу алгоритма SHAPEBUGS в следующих ключевых направлениях:

– изменить способ задания формы;
– изменить способ попадания объектов внутрь формы;
– изменить алгоритм выбора направления движения объекта внутри формы;
– предложить способ определения состояния сбалансированности формы после её изменений.

Для изменения способа задания формы откажемся от трафаретов и опишем форму математически, используя систему уравнений. В качестве примера рассмотрим трехмерную чашу, запаянную сверху и показанную на рис. 4.

$$\begin{cases} z = x^2 + y^2 \\ z \leq 5 \end{cases} \tag{1}$$

Очевидно, что такой подход имеет значительные преимущества по сравнению с трафаретным

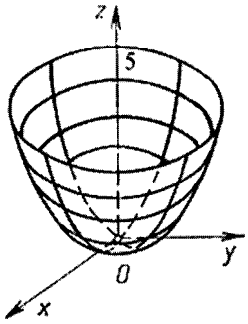

Рис. 4. Параболоид

подходом и методом градиентных карт. Главной его особенностью является минимальное количество или же и вовсе отсутствие избыточной информации, необходимой для описания формы. Так, например, для сжатия или расширения формы, приведённой на рис. 4, достаточно лишь изменить соответствующие коэффициенты в уравнении (1), в то время как при использовании градиентной карты для задания той же формы, потребовалась бы полная её перестройка, которая к тому же, вероятно, потребовала бы и увеличения размера карты. Иначе говоря, задание формы посредством уравнения или системы уравнений радикальным образом снижает требования к памяти и одновременно серьезнейшим образом повышает производительность алгоритма SHAPEBUGS.

Следует отметить при этом, что наравне с преимуществами такого подхода возникает и серьезная проблема, связанная с тем, что реальные сущности, имеющие сложную форму, трудно описать математически. Для решения этой проблемы необходима декомпозиция сложной формы на совокупность более простых форм, которые проще описать. Очевидно, что в этом случае возрастает и трудоёмкость реализации данного подхода, однако, преимущества, которые он даёт, все же диктуют необходимость продолжения исследований в данном направлении.

Другим направлением модификации алгоритма SHAPEBUGS является изменение задания способа попадания объектов внутрь формы. В существующей реализации объекты роя перемещаются случайным образом и со временем попадают в форму благодаря замкнутости пространства. Тем самым повышается время конвергенции системы, то есть приведения её в согласованное состояние, например, восстановление формы после удаления из роя части объектов или добавления новых объектов.

Предлагаемая модификация поведения объектов основана на понятии точек входа в форму. Точка входа в форму – это внутренняя точка формы, причем такая, что при движении по направлению к ней из любой точки пространства гарантируется попадание внутрь формы даже без достижения самой этой точки.

Предположим, что объекты изначально создаются вне границ формы, поэтому первостепенная задача – попасть внутрь. Для этого каждый объект обладает информацией о точках входов в форму. Пусть координаты объекта в пространстве выражаются вектором (x, y, z), а координаты точки входа – вектором (x_p, y_p, z_p) . Объект выбирает ближайшую точку входа (или ту, которая считается оптимальной по другим критериям) и устремляется к ней по формуле (2).

$$\begin{cases} x = x_p + b(x - x_p) \\ y = y_p + b(y - y_p) \\ z = z_p + b(z - z_p), \end{cases} \tag{2}$$

где b – константа, отвечающая за «величину шага». При этом, очевидно, b должно быть меньше 1, иначе объект будет не приближаться к точке входа, а отдаляться от неё.

Вопрос определения оптимальной точки входа в форму представляет отдельный интерес для исследования. Самой простой стратегией является выбор ближайшей точки, так как в этом случае объект будет находиться минимальное время за пределами формы и быстрее приступит к функционированию. При этом, однако, возможно, что в этой части формы существует большая концентрация других объектов, что потребует дальнейшего перераспределения объектов роя уже внутри самой формы. В этом случае, очевидно, было бы целесообразнее выбрать другую точку, которая бы позволила системе быстрее прийти к сбалансированному состоянию. Однако объект не способен

прогнозировать оптимальность точки входа и текущую загруженность объектами возле неё. Поэтому предлагается решение, позволяющее создавать объекты в разных сторонах от формы таким образом, что, применяя стратегию вхождения в ближайшую точку входа, объекты будут заведомо равномерно распределяться внутри. В простейшем случае возможен также вариант равномерного распределения точек входа внутри формы.

Ещё одной важной модификацией алгоритма SHAPEBUGS является переопределение поведения объектов роя при их попадании внутрь формы, так как необходимо обеспечить такое поведение и тактику, при которой объекты постараются распределиться равномерно.

Особое значение тактика распределения объектов приобретает в ситуации удаления или добавления части объектов роя (частичное разрушение формы или её наращивание). В общем случае поведение объектов определяется двумя принципами: «Отойди от ближайшего»и «Подойди к самому дальнему». Применение принципа «Отойди от ближайшего»соответствует ситуации, когда при изменении числа объектов в рое требуется восстановить исходную форму. Применение принципа «Подойди к самому дальнему»соответствует ситуации, когда при изменении части объектов в рое (особенно при удалении объектов) требуется их более компактная группировка, и объекты должны быть расположены как можно ближе друг к другу. Таким образом, применение той или иной тактики распределения объектов в форме определяется исходными условиями задачи.

Для иллюстрации применения этих принципов представим упрощённую ситуацию, когда в системе задействованы только 3 объекта. Будем работать с ограниченным с двух сторон одномерным пространством, т.е. объекты могут передвигаться только влево или вправо так, что значение их координаты будет ограничено на некотором отрезке $[a, b]$. В виду того, что каждый объект функционирует независимо от других, через некоторый промежуток времени (такт) производится выборка значений координаты объектов, что позволяет отслеживать изменения в системе, вызванные взаимодействием объектов между собой.

На рис. 5 показан пример применения принципа «Подойди к самому дальнему» при распределении объектов в форме. Очевидно, что при удалении одного из объектов два оставшихся будут стремиться друг к другу и максимально сблизятся. При добавлении же нового объекта возникнет ситуация, соответствующая такту 1 на рис. 5, за исключением того, что в системе будут присутствовать не три, а четыре объекта.

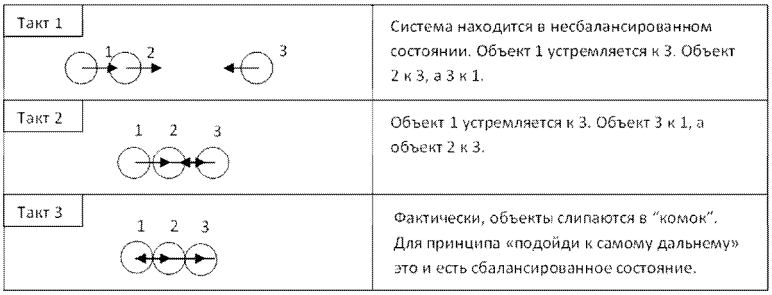

Рис. 5. Применение принципа «Подойди к самому дальнему»

На рис. 6 показан пример применения принципа «Отойди от ближайшего» при распределении объектов в форме. Как видно из рис. 6, объекты роя стремятся занять всё пространство, отведённое под форму, прич'м с максимально возможным равномерным распределением. Для дополнительной оптимизации распределения объектов предлагается исключить нейтральную область, что заставит объекты всегда находиться в состоянии отталкивания вплоть до приведения системы к полностью сбалансированному состоянию. За счёт этого обеспечивается более быстрая и равномерная наполняемость формы объектами по сравнению со стандартной реализацией SHAPEBUGS.

В стандартной реализации SHAPEBUGS новая позиция объекта вычисляется из суммы позиций объектов, учитывая расстояние до каждого из них. Иными словами, в произвольный момент времени необходимо знать расстояние до каждого объекта и его позицию, чтобы система функционировала корректно, т.е. объекты роя должны на каждом такте обмениваться сообщениями о своем текущем расположении. В предлагаемой же модификации, которая основана на исследованиях, приведённых в [7], используется информация лишь о ближайшем объекте.

Так как нахождение координат ближайшего объекта осуществляется значительно легче, чем поддержание в актуальном состоянии информации о позициях объектов на каждом такте, то данный подход требует значительно меньших ресурсов и ускоряет конвергенцию системы.

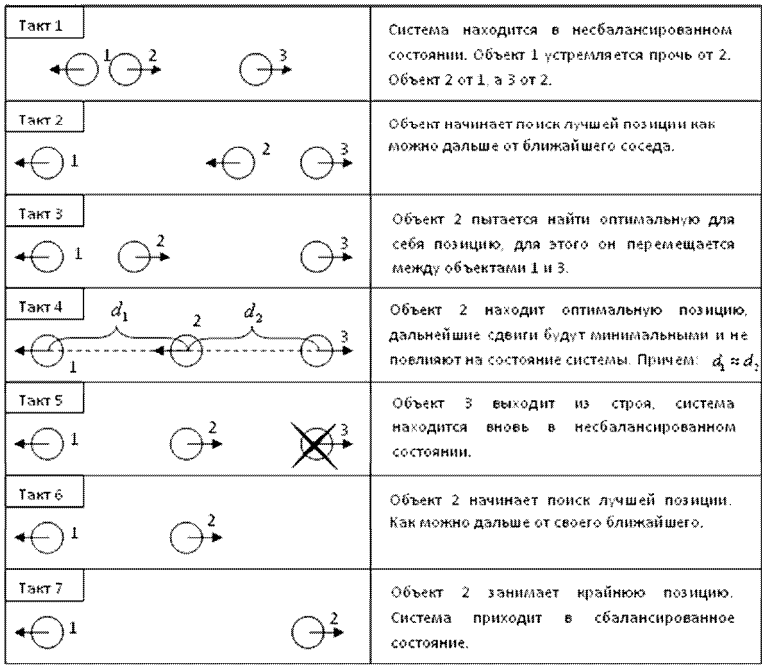

Рис. 6. Применение принципа «Отойди от ближайшего»

Важной задачей является также определение момента, когда система становится сбалансированной. Для простых форм эта задача не представляет особой сложности, так как достаточно определить внутренний объём, создаваемый совокупностью объектов роя, и сравнить его с объёмом фигуры (площадью для двумерного случая). Однако, в ситуациях, когда формы являются сложными и при этом не всегда выпуклыми, эта задача сразу становится весьма нетривиальной.

Задача определения сбалансированности системы тесно связана с задачей математического описания сложных форм. Как указывалось ранее, с этой целью применяется декомпозиция сложной формы на совокупность более простых форм в зависимости от исходной задачи. В качестве примера приведем оценку сбалансированности системы для случая двумерной фигуры на плоскости, показанной на рис. 7.

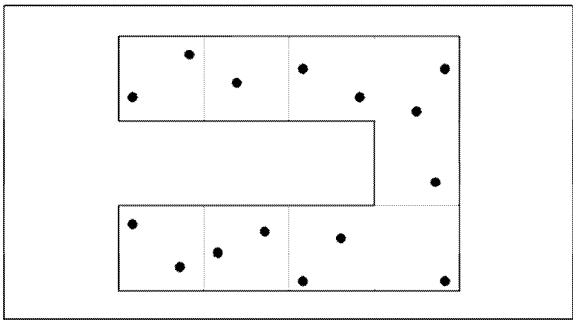

Рис. 7. Исходная двумерная форм

Общая формула для определения сбалансированности системы в данном случае имеет вид:

$$S_f - S_o \leq \varepsilon, \tag{3}$$

где S_f – площадь формы, S_o – площадь, задаваемая совокупностью объектов роя, ε – величина

ошибки. Легко заметить, что, варьируя величину ошибки, можно задать точность сбалансированности системы.

Приведённый пример иллюстрирует сложность адекватного определения площади, занимаемой объектами роя. Так, применение алгоритма Джарвиса [2] (алгоритм завёртывания подарка) создаёт несколько парадоксальную ситуацию, показанную на рис. 8. Как видно из рис. 8, все объекты находятся внутри формы, но площадь, которую они занимают, якобы больше площади самой формы.

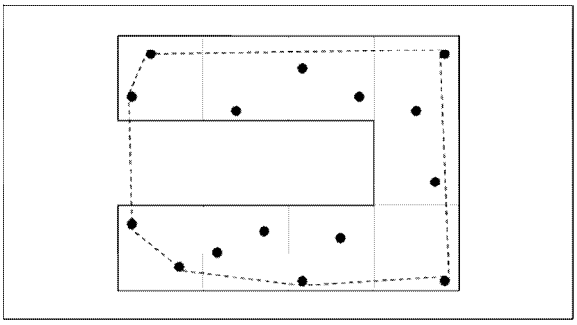

Рис. 8. Применение алгоритма Джарвиса к исходной форме

Очевидно, что в данном случае условие (3) выполнится, но система так и не придёт в сбалансированное состояние. Отсюда вытекает необходимость построения такого множества объектов роя на плоскости, которое по своим очертаниям максимально соответствовало бы форме, как показано на рис. 9. В этом случае площадь была бы определена корректно, и оценка сбалансированности системы была бы адекватной.

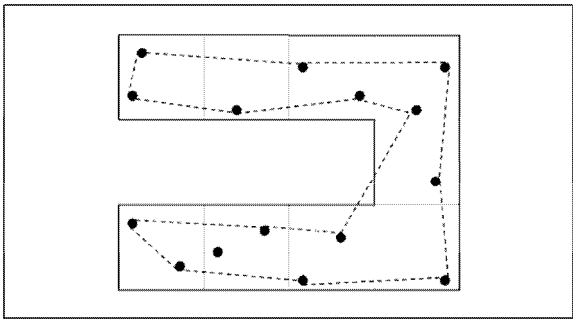

Рис. 9. Оптимизация площади, занимаемой объектами роя

Вообще говоря, построение площади, занимаемой объектами роя, которая максимально соответствовала бы исходной форме, является задачей, требующей дополнительных исследований, поэтому в качестве упрощённого решения может быть предложено построение выпуклой оболочки для каждой компоненты сложной формы.

При этом подходе невыпуклая форма разбивается на множество простых выпуклых форм, к каждой из которых по отдельности затем можно применять алгоритм Джарвиса или другой подходящий алгоритм, например, алгоритм Грэхема [1]. Декомпозиция формы на атомарные выпуклые компоненты показана на рис. 10.

Недостатки данного подхода выражаются, в первую очередь, в том, что теряется неучтённая площадь, которая фактически имеется на стыке атомарных компонент, вследствие чего ε из формулы (3) требуется делать достаточно большим. Но это влечёт за собой потерю точности определения состояния сбалансированности системы. Тем не менее, достоинством этого подхода является то, что он легко реализуем на практике, с учётом того, что форма строится из выпуклых компонент ещё на этапе ее проектирования. Таким образом, он может быть с успехом применён в ситуациях, где допустимо некоторое отклонение роя в сбалансированном состоянии от первоначальной формы.

Приведённый пример ясно демонстрирует проблемы, возникающие при декомпозиции сложной исходной формы на совокупность более простых форм, как на этапе её описания, так и при

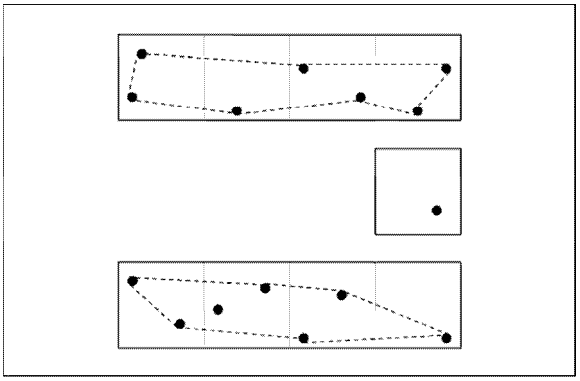

Рис. 10. Декомпозиция сложной формы

оценке сбалансированности состояния роя при работе алгоритма SHAPEBUGS. Вместе с тем, по нашему мнению, именно этот подход способен привести к значительному повышению эффективности работы алгоритма и радикальному снижению используемых вычислительных ресурсов, что компенсирует трудоёмкость подобного подхода и делает его предметом пристального внимания для дальнейшего изучения.

В целом же предложенные в рамках проведённых исследований модификации алгоритма SHAPEBUGS способны в будущем послужить основой для разработки специализированного программного обеспечения, находящего свое применение в техническом дизайне, медицинских исследованиях, проектировании инженерных систем и других сферах деятельности.

Список литературы

1. Алгоритм Грэхема (wikipedia)

2. Алгоритм Джарвиса (wikipedia)

3. Веб-сайт: http://habrahabr.ru/blogs/data_mining/67078/ (дата обращения: 12.01.2012)

4. Веб-сайт проекта SHAPEBUGS: http://www.shapebugs.org (дата обращения: 12.01.2012).

5. Cheng J., Winston Cheng, Radhika Nagpal. Robust and self-repairing formation control for swarms of mobile agents. Proceedings of the American Association for Artificial Intelligence (AAAI), 2005. 6 p.

6. Rubenstein M., Wei-Min Shen. Scalable Self-Assembly and Self-Repair In A Collective Of Robots, 2009. 6 p.

7. Unsal C., J. Bay. Spatial Self-organization in Large Populations of Mobile Robots. International Symposium on Intelligent Control, 1994. 14 p.

Статья поступила в редакцию 30.03.2012 г.