Исследование методов разбиения пространства с целью ускорения работы алгоритмов обнаружения столкновений движущихся объектов

М.Ю. Барышникова, Е.В. Брянская, В.А. Иванов

Московский государственный технический университет

им. Н.Э. Баумана, Москва, Россия

Аннотация

Работа посвящена анализу подходов к повышению временной эффективности алгоритмов вычисления столкновений движущихся объектов. Проанализированы алгоритмы, основанные на пространственном разбиении. Для верификации полученных данных разработано программное обеспечение, с помощью которого исследована зависимость времени работы алгоритмов от параметров сцены.

Постановка задачи

В настоящее время задача моделирования движения тел приобретает все большую актуальность. Данные модели используются, например, при разработке тренажеров или при создании компьютерных игр, сценарий которых основан на взаимодействии тел. Одной из проблем, которую приходится решать при создании систем подобного типа, является обнаружение столкновений физических объектов (такие столкновения часто принято называть коллизиями). Сложность построения таких визуальных моделей заключается в необходимости добиться плавности движения и реалистичности поведения довольно большого количества объектов при их соударении друг с другом, а также исключить эффект «проваливания» одного объекта в другой. В качестве дополнительного ограничения выступает требование минимизации времени на обновление сцены.

Важную роль в решении данной задачи играет выбор алгоритма вычисления столкновений, наиболее оптимального с точки зрения быстродействия.

В силу своей актуальности, проблемы обнаружения столкновений движущихся объектов активно исследовались в работах различных авторов. Так, например, в книге Кристера Эриксона «Real-Time Collision Detection» [5] приводится большое количество различных алгоритмов, по-разному решающих поставленную задачу. Результаты аналогичных исследований нашли отражение в статье «I-COLLIDE: an Interactive and Exact Collision Detection System for Large Scale environments» [6], написанной группой учёных из Университета Северной Каролины, в которой они подробно описывают разработанный ими алгоритм, основанный на использовании списка активных объектов. Однако, указанные работы не акцентируются на повышении временной эффективности предложенных алгоритмов. В них б**о**льший упор делается на универсальность по отношению к разнообразию форм взаимодействующих объектов.

Целью данного исследования было проведение анализа существующих подходов к решению задачи обнаружения столкновений физических объектов с точки зрения возможного улучшения временных характеристик данных алгоритмов.

Решение задачи вычисления столкновений объектов целесообразно начать с рассмотрения алгоритма полного перебора, который обеспечивает наибольшую корректность результата. Однако существенным ограничением его использования является сложность порядка $O(n^2)$ [1], что сказывается на временных характеристиках. Поэтому в первой части данной работы была поставлена задача проанализировать имеющиеся модификации алгоритма полного перебора и по возможности предложить собственные решения, направленные на уменьшение временных затрат. В ходе анализа должны были быть выявлены ограничения на условия применения различных алгоритмов и выработаны рекомендации по переходу с одного алгоритма на другой в зависимости от параметров сцены.

В качестве одного из возможных подходов было предложено использовать структуры деревьев для иерархического разбиения

пространства на подобласти [2]. Это позволит исключить из рассмотрения такие пары объектов, которые явно не могут сталкиваться в текущий момент времени.

Для проведения анализа временных характеристик различных алгоритмов была написана программа, моделирующая движение шаров, одинаковых по массе и размеру, в двухмерном пространстве. Шары могут сталкиваться с границами области и между собой. При этом используется модель абсолютно упругого удара и для упрощения не рассматривается вращение шаров вокруг собственной оси. Трение также не учитывается для сохранения динамичности системы в процессе исследования.

Ниже приведены краткие характеристики алгоритмов, рассмотренных в ходе анализа.

Алгоритм полного перебора

Как было сказано выше, использование алгоритма полного перебора является самым простым решением поставленной задачи. Принцип алгоритма заключается в том, что каждый объект проверяется на наличие его столкновения со всеми остальными.

В моделируемой сцене в качестве объектов выступают шары одинакового размера, поэтому, чтобы определить наличие коллизии между любыми двумя из них, нужно сравнить расстояние между их центрами с двойным радиусом. В случае, если расстояние меньше или равно сумме радиусов, считается, что столкновение есть.

Преимуществом данного метода является:

- простота реализации;
- отсутствие необходимости использования дополнительных структур данных;
- универсальность.

Однако, как было сказано выше, существенным недостатком алгоритмом полного перебора является высокая сложность, которая при

увеличении количества объектов на сцене будет провоцировать быстрый рост временных затрат.

При этом для небольшого количества объектов данный алгоритм показывает хорошие результаты, поэтому он используется как основа для построения алгоритмов разбиения пространства, которые сводят задачу распознавания коллизий большого количества физических объектов в широкой области к обработке множества ограниченных подобластей с малым числом тел.

Далее будут рассмотрены следующие алгоритмы разбиения пространства.

- Quadtree (дерево квадрантов). Потенциал алгоритма заключается в наиболее простом и адаптивном разделении пространства на прямоугольные области.
- BSP (двоичное разбиение). Данный алгоритм показывает высокую гибкость по отношению к форме и размеру сцены.
- Bin algorithm (корзинное разбиение). Алгоритм обеспечивает разбиение пространства на регулярную сетку, что эффективно при большой плотности заполнения сцены физическими объектами.

Рассмотрим данные методы подробнее.

Алгоритм дерева квадрантов (Quadtree)

Относится к числу наиболее известных алгоритмов разбиения пространства и имеет следующий принцип работы.

1. Изначально создаётся структура корневого дерева, в которую будут добавляться объекты.
2. После выполнения ряда шагов по добавлению объектов, когда будет достигнуто их «критическое количество», создаются деревья, разбивающие область на четыре одинаковые прямоугольные части. Общая вершина этих прямоугольников располагается ровно в

центре разделяемого пространства. Объекты распределяются по новым подобластям.

3. После добавления всех объектов, производится обход по дереву и в каждом концевом листе (то есть неразделённой области) применяется алгоритм полного перебора для поиска столкновений.
4. Проверка вхождения объекта (в данном случае – шара) в подобласть производится путем сравнения координат углов прямоугольной подобласти и описанного вокруг него квадрата. Если обнаруживается их пересечение, то считается, что тело принадлежит области.

Графически принцип работы данного метода представлен на рис. 1. Линии показывают границы областей, выделенные деревом квадрантов.

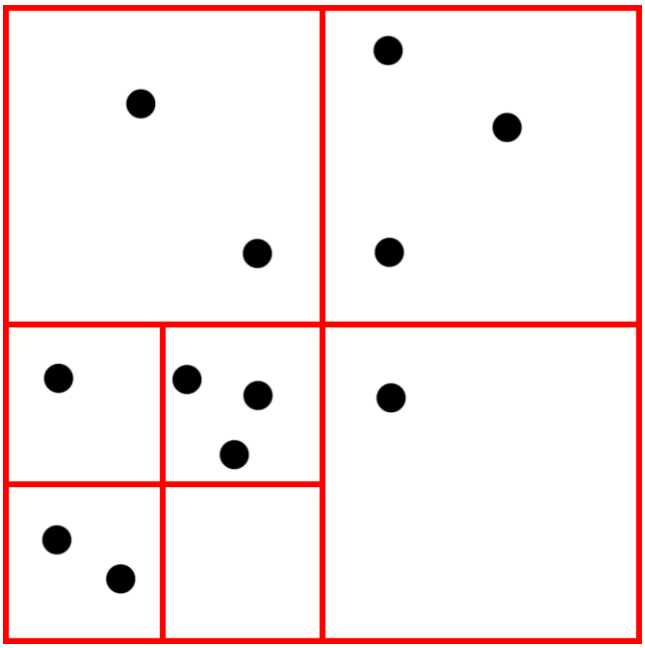

Рисунок 1. Алгоритм Quadtree

Общей проблемой для всех способов пространственного разбиения является определение правил принадлежности для шаров, расположенных на границе двух и более подобластей [4]. Так как такой шар может иметь столкновения с телами в каждой из затронутых им областей, его следует занести в каждую из них.

В ходе изучения принципов работы алгоритма, использующего дерево квадрантов (Quadtree), авторами работы была выдвинута гипотеза о возможности провести его модификации с целью улучшения временных характеристик. Эти предложения описаны ниже.

Алгоритм nona-дерева

Поскольку целью модификации является ускорение работы алгоритма Quadtree, было предложено уменьшить его вычислительные затраты за счёт увеличения количества подобластей. Для этого исследуемое пространство делится не на четыре одинаковые области, а на девять. Данное число областей возникает причине того, что каждая сторона разбивается на три части. В результате уменьшается глубина дерева, что позволяет найти решение за меньшее количество переходов по дереву, а следовательно – сократить время обработки.

Иллюстрация работы данного метода приведена на рис. 2.

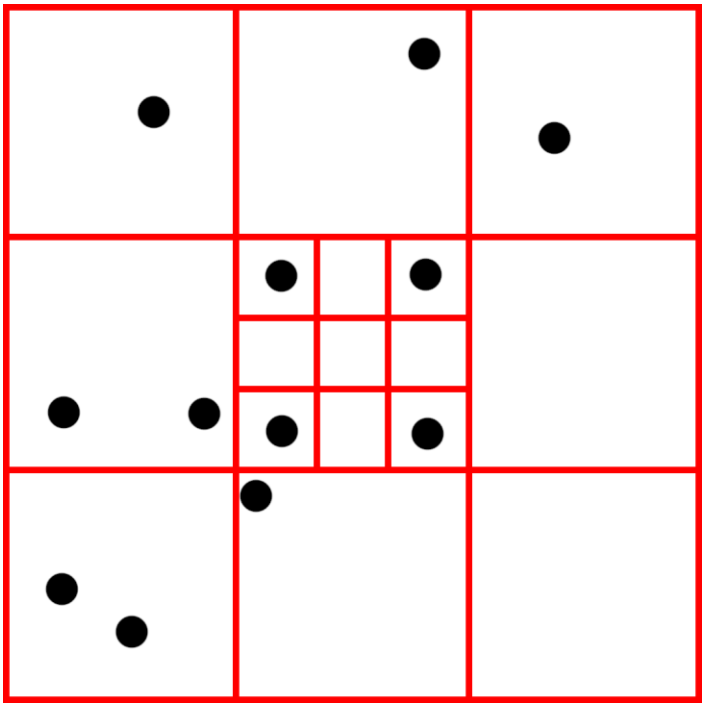

Рисунок 2. Алгоритм nona-дерева

Еще одной проблемой, которая возникает при использовании алгоритмов пространственного разбиения, является невозможность

равномерно распределить объекты по подобластям. Для ее решения была предложена еще одна модификация алгоритма дерева квадрантов, основанная на перемещении центра разбиения.

Алгоритм дерева квадрантов с динамическим центром

Для того, чтобы достичь желаемого результата, чаще всего координаты точки разбиения задают как среднее от координат всех центров объектов (в данной модели оно совпадает с центром масс системы).

По достижении критического числа объектов в области в первую очередь будет определяться сумма координат всех объектов. Поделив её на количество объектов, вычисляется центр разбиения. Производится разделение пространства на подобласти с распределением в них объектов (рис. 3).

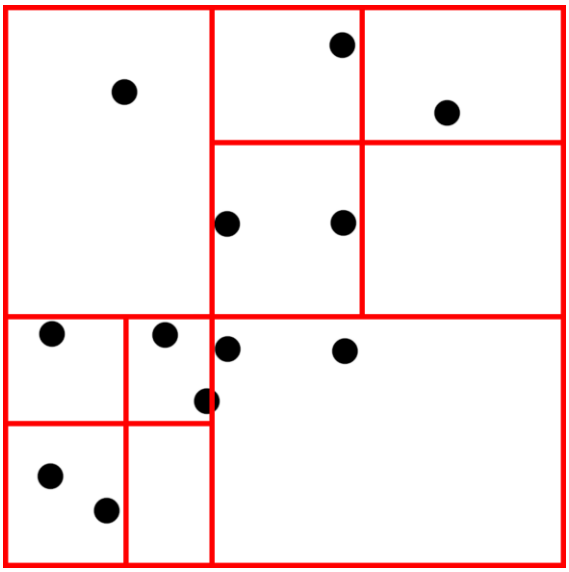

Рисунок 3. Алгоритм дерева квадрантов с динамическим центром

Вычисление координат средней точки всех тел является затратным по времени действием. Центр каждого тела будет учитываться при каждом разбиении, поэтому в среднем один объект будет учтён log4(n) раз, где n – число объектов в системе. Всего дополнительные вычисления будет произведены примерно n*log4(n) раз.

Но, для уменьшения затрат вычисления средних координат можно использовать небольшое количество случайно выбранных шаров.

Теоретически, две представленные модификации могут улучшить производительность алгоритма дерева квадрантов. Ниже приводятся экспериментальные данные, подтверждающие целесообразность их практического применения для решения поставленной задачи.

Однако, квад-деревья не являются единственным подходом для иерархического разбиения пространства. Принципиально другое решение данной проблемы предлагает алгоритм двоичного разбиения.

Алгоритм двоичного разбиения (BSP-дерево)

Как следует из названия, в данном методе разбиение пространства происходит на две равные части. Сцена имеет прямоугольную форму, поэтому её деление производится по диагонали и приводит к образованию двух равных прямоугольных треугольников. Их дальнейшее разбиение осуществляется по медиане к самой длинной стороне. Полученные части удобнее всего обозначить как левая и правая относительно вектора, лежащего вдоль границы (при разбиении прямоугольника направление вектора берётся от нижней вершины, в треугольнике – от наибольшего угла).

При подобном делении усложняется задача определения принадлежности объектов к областям. Её иллюстрация представлена на рис. 4.

Определить положение точки относительно разделяющего отрезка можно с помощью векторного произведения вектора, лежащего вдоль него, и вектора от конца отрезка до рассматриваемой точки. На рисунке эти векторы окрашены зелёным и коричневым цветом соответственно. Точки, лежащие правее разделителя, будут иметь положительное векторное произведение, левее – отрицательное.

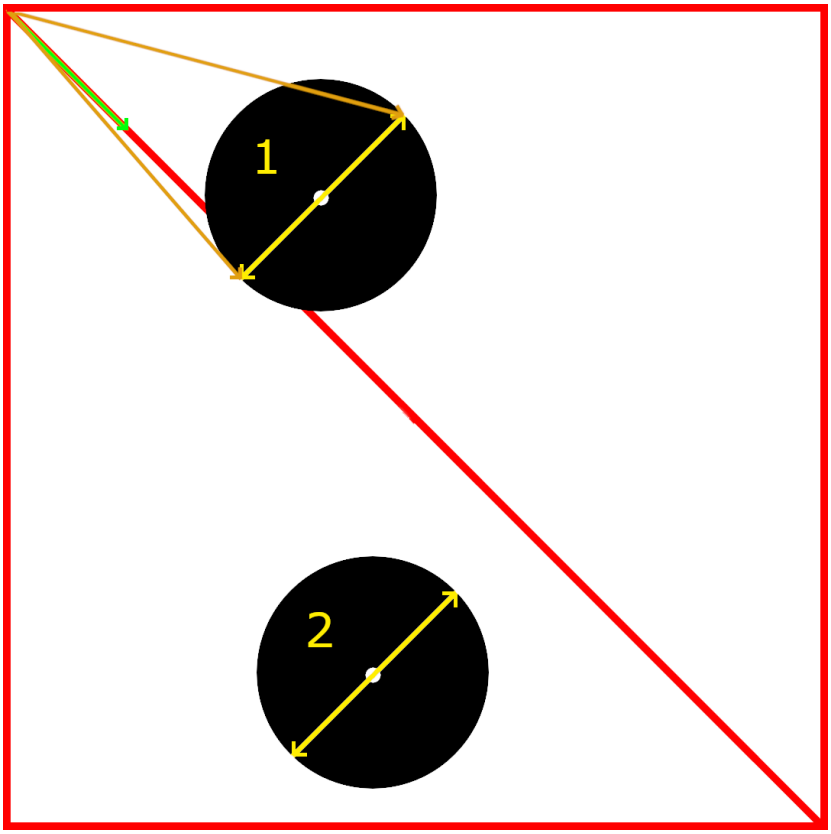

Рисунок 4. Задача определения принадлежности объектов в BSP дереве

Для проверки шара на принадлежность к правой области следует использовать самую «правую» точку окружности. Положение точки вычисляется как сумма координат центр шара и вектора радиуса, направленного перпендикулярно к разделителю. Аналогично – для определения принадлежности к левой области. На рис. 4 вычисление этих точек представлено с помощью векторов.

Более наглядно алгоритм представлен на рис. 5.

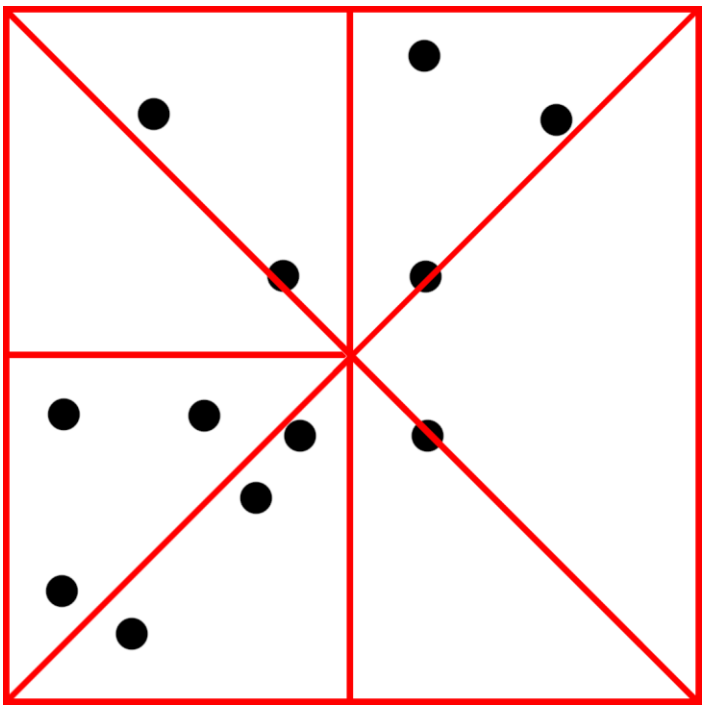

Рисунок 5. Алгоритм BSP

Алгоритм корзинного разбиения (Bin algorithm)

Все ранее описанные алгоритмы используют иерархическое разбиение пространства, основанное на утверждении, что доступ к нужной области обеспечивается через ряд последовательных разбиений более обширных областей. Отличие заключается в принципах построения иерархий.

Однако разбиение пространства можно организовать с помощью одинаково равноправных разбиений. Такой принципиально отличный подход используется в алгоритме корзинного разбиения.

Всё поле делится на одинаковые прямоугольные области (называемые «корзинами», от анг. bin) [5]. В общем случае алгоритм предназначен для оценки наличия пересечения множества прямоугольных областей. Для этого каждая корзина представляется массивом объектов, которые частично или полностью попадают в данную область.

Для включения объекта в структуру нужно найти области, в которых расположены его вершины с минимальными и максимальными координатами. После чего, рассматриваемый объект заносится во все корзины, которые расположены между найденными.

Применительно к данной задаче, шары описываются прямоугольниками с координатами углов (x-r, y-r) и (x+r, y+r), где (x, y) - центр, r - радиус. Стоит отметить, что использование размера корзин меньше 2r не имеет смысла, так как некоторые области будут полностью внутри шаров.

После добавления всех шаров в структуру для обнаружения их столкновений достаточно проверить корзины, в которых находятся два и более шара.

Графически алгоритм показан на рис. 6.

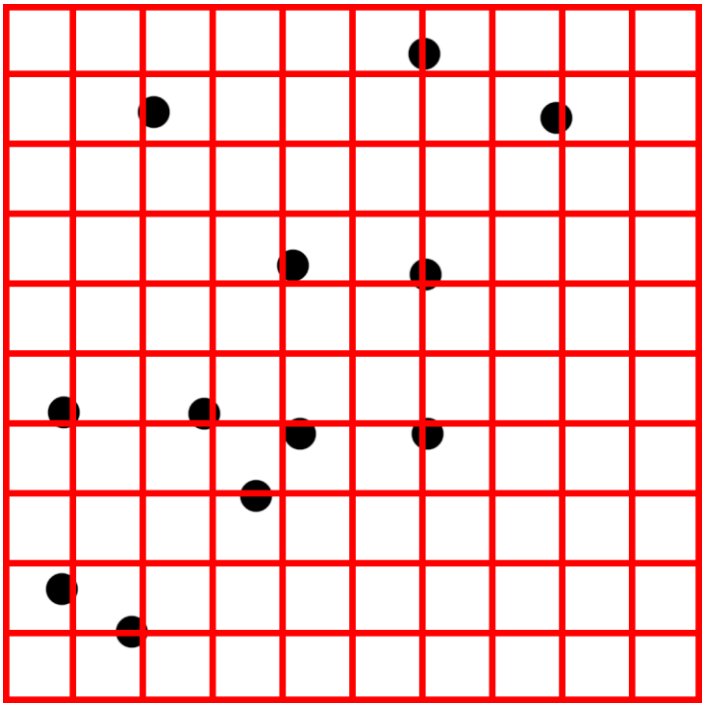

Рисунок 6. Bin алгоритм

При небольшой плотности заполнения поля производить обход всех областей неэффективно. Для этого можно воспользоваться дополнительной структурой связанного списка. В него заносятся клетки, в которых число шаров больше одного.

Явным недостатком является зависимость занимаемой памяти от площади контролируемого пространства, что делает проблемным его использование для обширных сцен.

Результат проведенного анализа алгоритмов сведены в таблицу 1, в которой n – количество частиц, S – площадь сцены.

	Полный перебор	Quad дерево	Динами-ческий центр	Nona дерево	BSP	Bin
Сложность	$O(n^2)$	O(n* ln(n))	O(n* ln(n))	O(n* ln(n))	O(n* ln(n))	O(S*n) или O(n)
Дополни-тельная память	0	O(n)	O(n)	O(n)	O(n)	O(S)
Гибкость по отношению к пространству	Сильная	Средняя	Средняя	Средняя	Сильная	Слабая
Склонностью к параллелизму	Низкая	Средняя	Средняя	Средняя	Средняя	Высокая

Таблица 1 - Сравнение алгоритмов

Верификация полученных результатов анализа алгоритмов вычисления столкновений физических объектов в моделях их взаимодействия была проведена с помощью специально написанной для этой цели программы.

Сравнительный анализ эффективности реализации алгоритмов обнаружения столкновений

Ниже приведены графики зависимости количества обновлений сцены (также называемое FPS - frames per second) за одну секунду от числа шаров

при использовании каждого из рассмотренных алгоритмов. В обновление сцены включается операция по перемещению шаров, поиску их столкновений с бортиком и между собой.

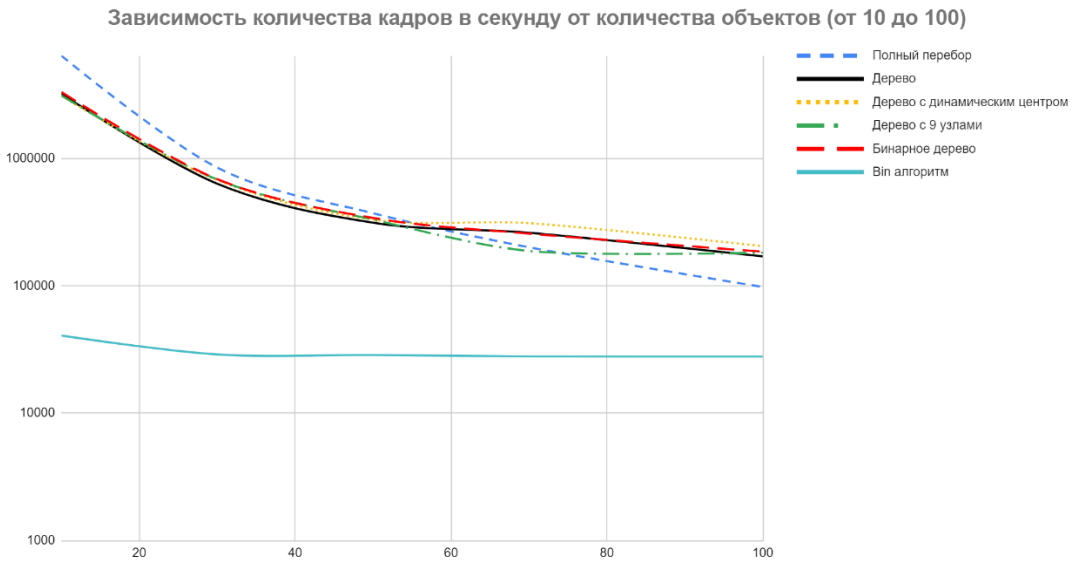

Рисунок 7. Зависимость количества кадров в секунду от количества объектов (от 10 до 100)

Приведённые на рисунке 7 данные позволяют сделать следующие выводы.

- При малом количестве объектов (n < 100) лучшие показатели демонстрирует алгоритм полного перебора, поскольку в ходе его использования не создаются дополнительные структуры данных, использование которых в данном случае не оправдано. Но ближе к 100 он уступает древовидным алгоритмам.
- Время работы Bin алгоритма практически не изменяется в заданном диапазоне. Количество обновлений сцены значительно ниже по сравнению со всеми остальными алгоритмами.

При количестве объектов от 100 до 1000 (рис. 8):

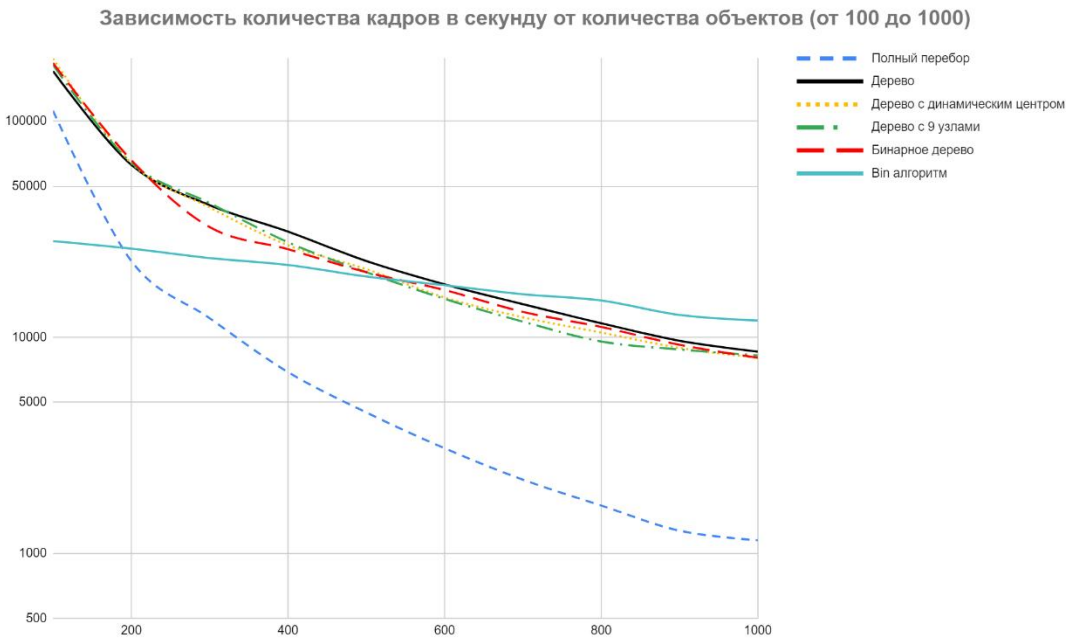

Рисунок 8. Зависимость количества кадров в секунду от количества объектов (от 100 до 1000)

- Полный перебор начинает ощутимо проигрывать остальным алгоритмам по скорости работы. Его использование становится нерациональным.
- При n ближе к 1000 Bin алгоритм начинает превосходить древовидные алгоритмы по времени работы. Однако, временное преимущество незначительно, но требует значительно больших затрат по памяти. Поэтому в данном диапазоне наибольший интерес представляют алгоритмы, использующие деревья.

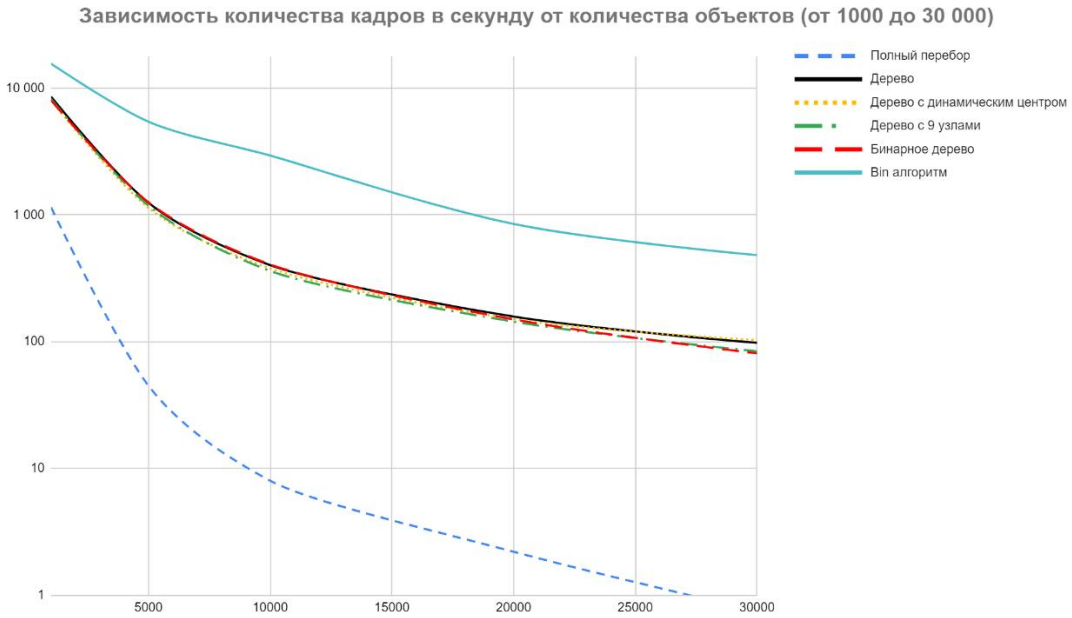

Рисунок 9. Зависимость количества кадров в секунду от количества частиц (от 1000 до 30 000)

Из графика, представленного на рисунке 9, следует, что:

- алгоритм полного перебора неприменим для большого (n > 1000) количества объектов;
- Bin алгоритм демонстрирует скорость работы на порядок большую, чем у алгоритмов, использующих деревья. Поэтому при таком количестве объектов его использование является наиболее оправданным.

Большое количество взаимодействующих объектов приводит в каждом из алгоритмов к образованию большого количества областей. Как отмечалось выше, области являются практически полностью изолированными друг от друга. Поэтому на следующем этапе эксперимента была применена многопоточная реализация всех перечисленных алгоритмов, кроме алгоритма полного перебора. Результаты эксперимента представлены на рис. 10.

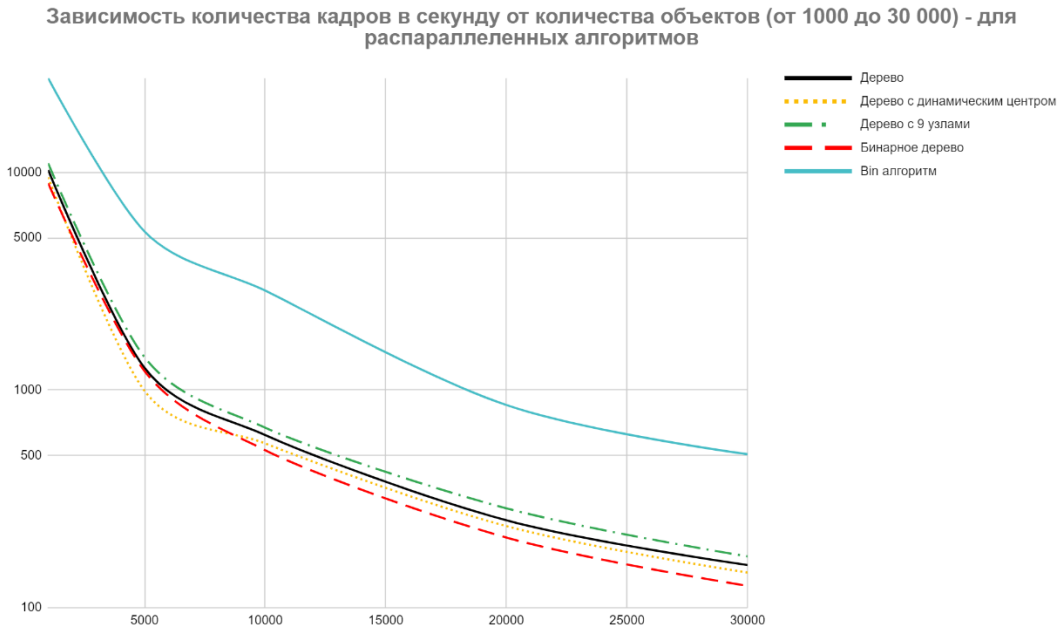

Рисунок 10. Зависимость количества кадров в секунду от количества объектов (от 1000 до 30 000) для распараллеленных алгоритмов.

Полученные результаты показывают, что использование многопоточности позволяет несколько улучшить временные показатели всех алгоритмов.

Проведенный анализ позволил сформулировать следующие рекомендации по использованию алгоритмов обнаружения столкновений.

- Для малого количества частиц (n < 100) целесообразно использовать алгоритм полного перебора.
- Для среднего количества частиц (100 < n < 1000) наиболее предпочтительным является алгоритм дерева квадрантов.
- Для большого количества частиц (n > 1000) предпочтение следует отдать Bin алгоритму, по возможности – многопоточному.
- Использование алгоритма с динамическим центром показывает наилучшие временные характеристики, когда количество частиц

примерно равно 100. В такой сцене равномерность распределения существенно не проявляется.

- Алгоритм nona-дерева более расположен к параллелизму, чем оригинальный алгоритм Quadtree.
- Алгоритм двоичного разбиения пространства также в основном уступает дереву квадрантов.

В заключение можно отметить, что проведенный анализ показал, что алгоритмы вычисления столкновений движущихся объектов, основанные на пространственном разбиении и использовании деревьев, имеют потенциал для дальнейшей модификации и развития в сторону уменьшения временных характеристик. Наиболее очевидными являются модификации, которые позволяют наследовать структуры разбиений из предыдущего момента времени вместо их полного пересоздания.

Список литературы

1. Алгоритмы. Построение и анализ : пер. с анг. / Кормен Т., Лейзерсон Ч., Ривест Р. [и др.]. - 3-е изд. - М. : Вильямс, 2018. - 1323 с. : ил.
2. Алгоритмы. Руководство по разработке. – 2-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2018. – 720 с.: ил.
3. Собинов Д. И., Коробицын В. В. Алгоритмы обнаружения столкновений // МСиМ. 2010. №1 (21). URL: https://cyberleninka.ru/article/n/algoritmy-obnaruzheniya-stolknoveniy (дата обращения: 20.03.2021).
4. Тотмаков А. С. Оптимизация доступа к пространственным данным // Исследовано в России. 2005. №. URL: https://cyberleninka.ru/article/n/optimizatsiya-dostupa-k-prostranstvennym-dannym (дата обращения: 20.03.2021).

5. Ericson, Christer Real-Time Collision Detection. - ISBN: 1-55860-732-3 изд. - San Francisco: Elsevier Inc., 2005. - 632 с.
6. Cohen, Jonathan D.; Lin, Ming C.; Manocha & Ponamgi, Madhav K. (April 9-12, 1995), I–COLLIDE: an Interactive and Exact Collision Detection System for Large Scale Environments), Proceedings of the 1995 Symposium on Interactive 3D Graphics (Monterey, CA), с. 189–196