Алгоритми пошуку взаємного розташування довільних геометричних фігур

Автор роботи:
Студент групи ДА-62
Тіхонов Андрій Миколайович

Об’єкт дослідження:

Алгоритми розбиття простору.

Предмет дослідження:

Аналіз використання алгоритмів розбиття простору для задачі пошуку взаєморозташування геометричних фігур.

Мета роботи:

Дослідження методів аналізу взаємоположення геометричних тіл для пришвидшення розрахунків в різних сферах, як то: наукоємні симуляції, комп'ютерна графіка, створення електросхем, CAD/CAM системи.

Результат роботи:

Порівняльна таблиця всіх методів в O-нотації, а також детальний аналіз кожного з алгоритмів та власний запропонований підхід для вирішення задачі пошуку перетину фігур з променями.

Актуальність:

- Наразі не існує систематизованої бази знань по даним алгоритмам
- Кожна з наступних галузей потребує обробки все більших і більших наборів даних, що потребує більш ефективних алгоритмів
  - Комп’ютерна графіка
  - Gis системи
  - CAD/CAM
  - Наукоємні симуляції
  - Медицина (наприклад проект FOLDING@HOME)

Які алгоритми було проаналізовано

BSP

KD-tree

BVH

Uniform grid

Quadtree

Octree

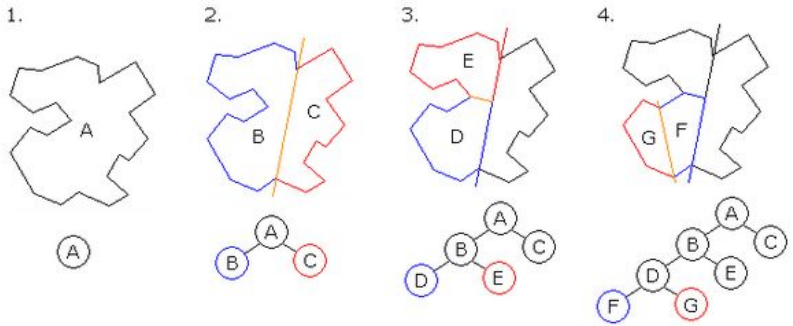
R-tree

Non uniform tree

BINs

Combined algorithms

BSP



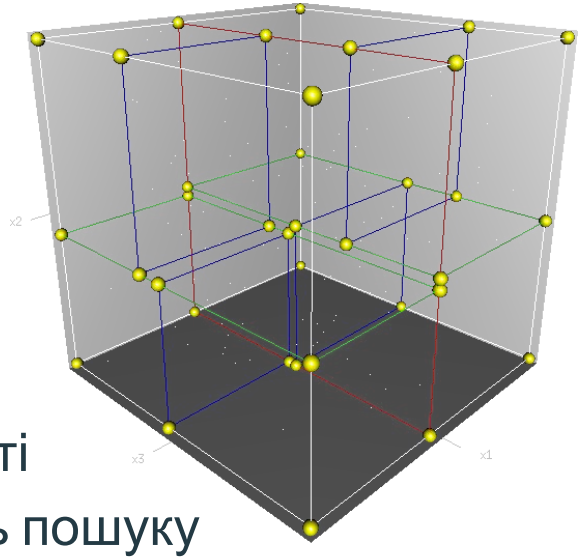
Переваги

- Простий метод, що легко імплементувати в класичній варіації

Недоліки

- Працює тільки з двомірними даними
- Якщо робити розділ не паралельно до вісей координат є складність в розділені не опуклих фігур без створення не зв'язних областей

KD-tree



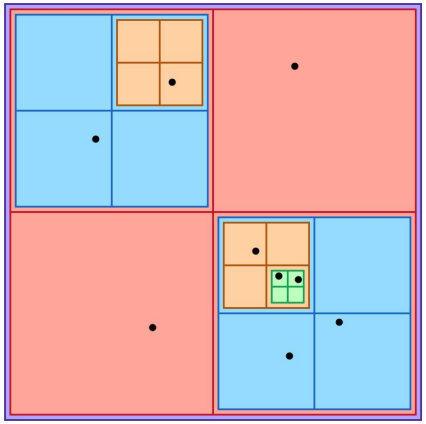
Переваги

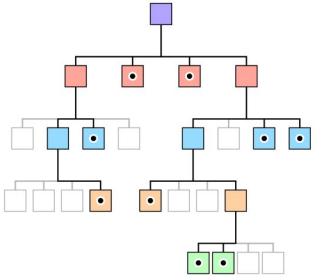
- Робота з даними будь-якої розмірності
- Як і інші дерева має високу швидкість пошуку

Недоліки

- При додаванні відсортованих даних потім потребує балансування

Quadtree





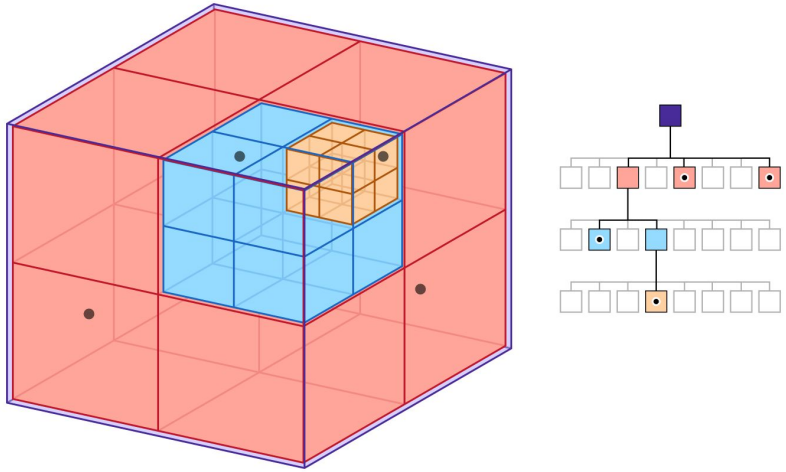
Переваги

- Автобалансування
- Ефективна робота з 2d даними

Недоліки

- Може працювати тільки з 2d чи з 3d при умові, що об’єкти розташовані близько до осі Z

Octree



Переваги

- Автобалансування
- Ефективна робота з 3d даними

Недоліки

- Без додаткової структури відсутня локальність даних збережених у пам'яті, що веде до кеш промахів

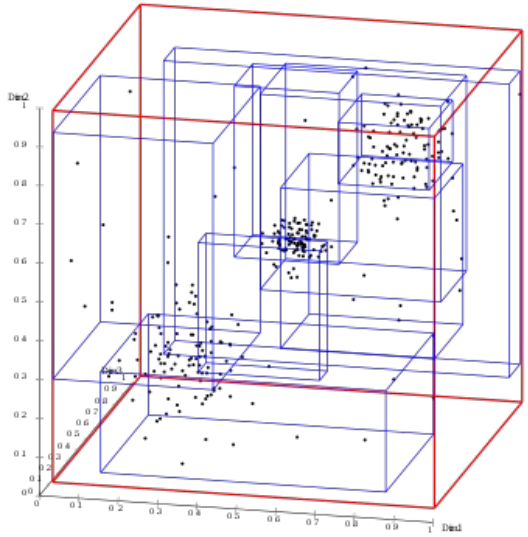
R-tree

Переваги

- Добре підходить для пошуку сусідніх об’єктів до заданого

Недоліки

- Погано працює з пошуком за параметром



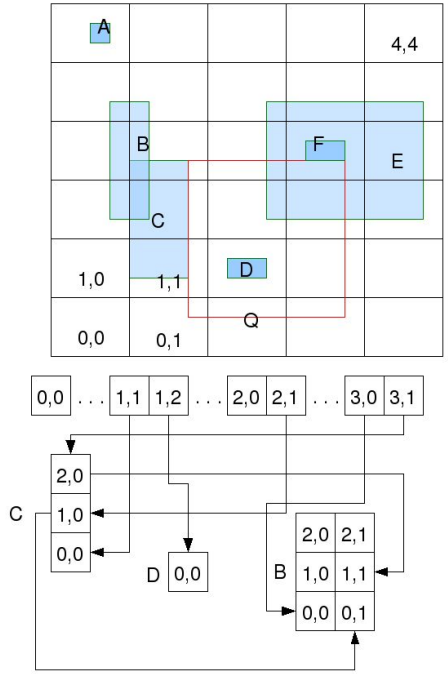
BIN

Переваги

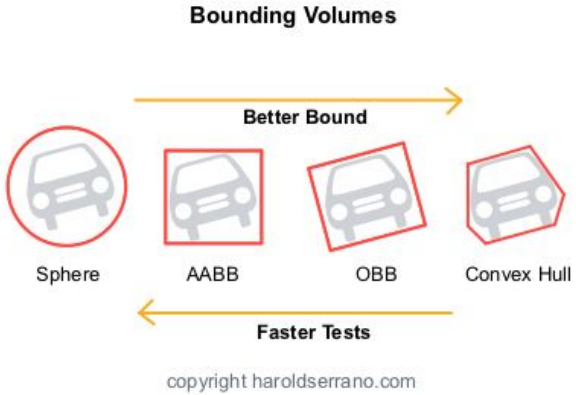
- Швидкий пошук цілих проміжків
- Локальність даних, менша кількість кеш промахів

Недоліки

- Повільна робота пошуку за параметром
- Працює тільки з 2d даними



BVH



Переваги

- Легка паралелізація для пришвидшення на GPU

Недоліки

- Багато варіацій методу, потрібно кожного разу підбирати потрібний для кожної конкретної задачі оптимізації

Uniform grid

Переваги

- Дуже проста реалізація
- В комбінації з іншими алгоритмами дає прискорення до 3 разів

Недоліки

- Погана алгоритмічна складність

Non-uniform grid

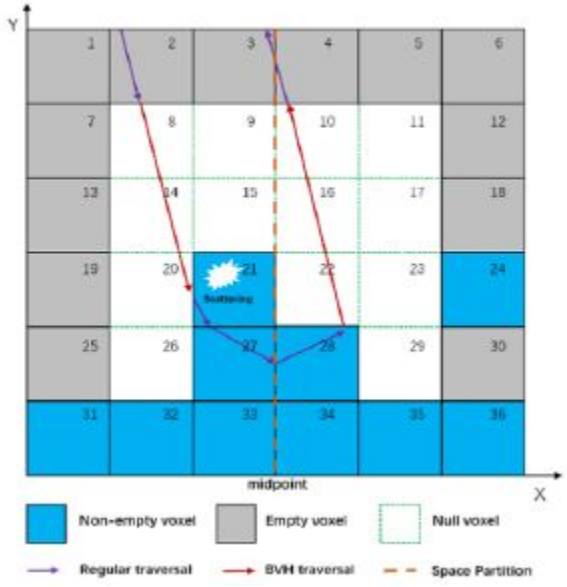
Переваги

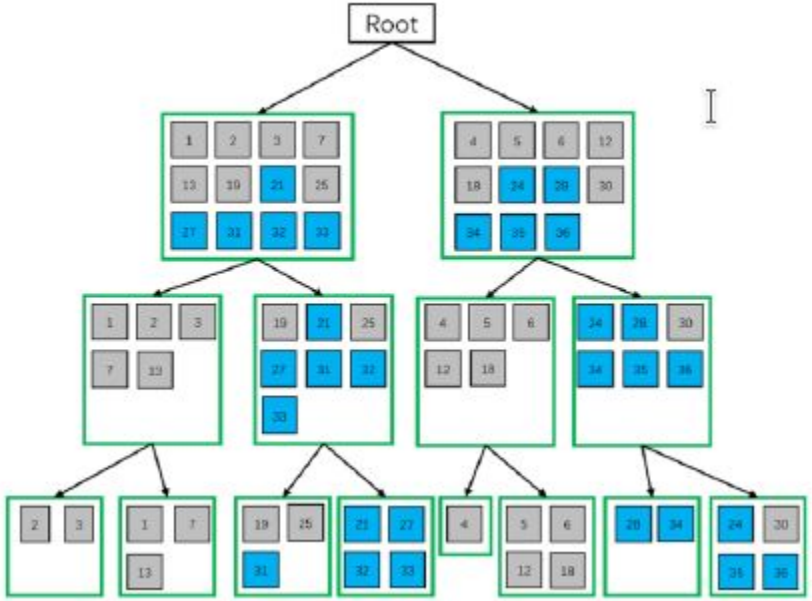
- Збільшена швидкодія в порівнянні з минулим алгоритмом

Недоліки

- Погана алгоритмічна складність
- Складна реалізація

Combined methods





Combined methods

Наразі майже не існує публікацій щодо використання змішаних алгоритмів. А в тих публікаціях, що вдалося знайти, показані результати роботи кількох алгоритмів за якими можна судити про високу ефективність у вигляді зменшення часу розрахунків в 3 рази.

Також мною було виявлено декілька гіпотез за підтвердження яких можна отримати ще більш ефективні алгоритми

Результати

	В середньому				Найгірший випадок			
	Створення	Вставка	Видалення	Пошук	Створення	Вставка	Видалення	Пошук
BVH	N^2	N	log(N)	log(N)	N^2	N	N	N
Octree	log(N)	log(N)	log(N)	log(N)	N	N	N	N
KD-tree	log(N)	log(N)	log(N)	log(N)	N	N	N	N
Uniform grid	N	log(N)	1	N	N log(N)	N	N	N
Non uniform grid	N	log(N)	1	N	N log(N)	N	N	N
Uniform grid + BVH	N^2	log(N)	log(N)	log(N)	N^2	N	N	N
Octree + BVH	N^2	log(N)	log(N)	log(N)	N^2	N	N	N

Результати

		Dragon	Manuscript
BVH	Створення	47439	26236
	Рендер	612	707
Octree	Предоброка данних	817984	412125
	Створення	4025	2017
	Рендер	417	665
KD-tree	Створення	10113	5701
	Рендер	515	815
KD-tree з невідсортованим масивом даних	Почергова вставка всіх елементів	54942299	30256838
	Висота дерева	24	24
KD-tree з відсортованим масивом даних	Почергова вставка всіх елементів	9856998	5140657
	Висота дерева	56	57

План подальшої роботи

- Дослідити альтернативні комбіновані алгоритми
- Імплементувати всі алгоритми у вигляді бібліотеки на одній мові з однаковими оптимізаціями
- Дослідити паралелізацію на змішаних платформах

Висновки

В ході роботи вдалося:

- Оцінити алгоритмічну складність алгоритмів
- Визначити напрямок подальших досліджень
- Дослідити найпопулярніші алгоритми на реальних даних
- Визначити особливості кожного метода

Дякую за увагу
студент групи ДА-62
Тіхонов Андрій Миколайович