

Parallel N-Body

Проектна
документация

Изготвил
Никола Тотев

Съдържание

1.	ВЪВЕДЕНИЕ	3
1.1	ОПИСАНИЕ НА ЗАДАЧАТА.....	3
1.2	МАТЕМАТИЧЕСКА ФОРМУЛИРОВКА	3
2.	АНАЛИЗ НА СИСТЕМАТА	4
2.1	ЖЕЛАНА ФУНКЦИОНАЛНОСТ	4
2.2	АНАЛИЗ НА АЛГОРИТМИ	4
2.3	АНАЛИЗ НА ЕКЗЕМПЛЯРИ	7
2.4	ИЗБОР НА АЛГОРИТМИ.....	12
3.	РЕАЛИЗАЦИЯ	13
3.1	ИЗБОР НА ЕЗИК И СРЕДА ЗА РАЗРАБОТКА	13
3.2	ОПИСАНИЕ НА ИМПЛЕМЕНТИРАНИТЕ АЛГОРИТМИ.....	13
3.3	ОПИСАНИЕ НА РЕАЛИЗАЦИЯТА	15
3.4	СТРУКТУРА НА ПРОГРАМАТА.....	20
4.	РЕЗУЛТАТИ.....	26
4.1	ТЕСТОВ ПЛАН.....	26
4.2	ПОЛУЧЕНИ РЕЗУЛТАТИ.....	26
4.3	АНАЛИЗ.....	31
4.4	АНИМАЦИЯ	32
5.	УПЪТВАНЕ ЗА УПОТРЕБА	33
5.1	ИНТЕРФЕЙСА	33
5.2	ВИДОВЕ НАСТРОЙКИ	34
5.3	СТЬПКИ ЗА УПОТРЕБА	37
6.	НАСОКИ ЗА БЪДЕЩО РАЗВИТИЕ	38
7.	ИЗТОЧНИЦИ	38

1. Въведение

Този проект се фокусира върху проблема за създаване на N-Body симулация. Дава се описание на задачата, както и математическа формулировка. Извършва се анализ на системата, като се описва желаната функционалност, анализират се съществуващи алгоритми както и екземпляри които използват тези алгоритми. На база на този анализ се избират подходящи алгоритми, които да се използват в системата. След фазата анализ се описва разработката на проекта, анализ на получените резултати, описва се структурата на проекта, дават се описания за употреба на системата и насоки за бъдещо развитие.

1.1 Описание на задачата

Същността на N-Body симулацията е да се проследят взаимодействията между n на брой тела. Тези взаимодействия могат да се подчиняват на закони като гравитацията или други физични закони като динамика на флуидите. Този проект се фокусира върху случая за гравитацията. Този случай е полезен за астрономията, защото позволява да се симулира начина, по който тела във вселената се движат и как нови галактики се създават.

1.2 Математическа формулировка

Тази секция ще разгледа N-Body проблема от математическа гледна точка. Основната цел е да се дефинират формулите, които се използват.

• Изчисления за привличане

Всяка една частица в симулацията се влияе от гравитационното привличане на всички други частици като това привличане се описва от закона на обратния квадрат [8]. Това е научен закон, който гласи, че определена физическа величина е обратно пропорционална на квадрата на разстоянието от източника на тази физическа величина.

Ускорението на всяка частица се изчислява с тази формула [8]:

$$\mathbf{a}_i = G \sum_{j \neq i} m_j \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|^3}$$

Където $G=6.67\times10^{-11}$ $\text{m}^3/\text{kg}\cdot\text{s}^2$.

2. Анализ на системата

2.1 Желана функционалност

Крайната цел на проекта е да се създаде симулация на n на брой тела с еднаква маса.

Трябва да има възможност да се прави автоматично тестване за да може лесно да се сравни последователната и паралелната обработка.

Резултатите от автоматичното тестване трябва да се визуализират в три графики (време за обработка, ниво на паралелизъм, ефективност) и да се запазват във файл.

Симулацията трябва да е 60 кадъра в секунда. Трябва да се генерира кадър по кадър, като всеки кадър се запазва като .png файл. Броя кадри се дефинира от потребителя и след като всички кадри се генерират трябва да се превърнат във видео с формат .mp4.

Трябва да има интерфейс, който е лесен за ползване и да позволява достъп до всички настройки на програмата. Интерфейса трябва да предоставя на потребителя обрана връзка за статуса на извършващата се дейност.

2.2 Анализ на алгоритми

Поради приложението на N-Body симулациите в много области на науката, има голям избор от алгоритми, които могат да се използват за генериране на симулация.

В рамките на този проект се запознах подробно с два основни алгоритъма, като в следващите подточки, ще ги опиша подробно, като след това ще спомена алтернативни алгоритми, който намерих, но нямах възможност да анализирам подробно.

Основния подход на всичките алгоритми е да се направи апроксимация на частиците, като се групират. Това групиране помага да не се правят изчисления всяко с всяко, което спестява време.

Примери за такива алгоритми са алгоритъма на Appel [3], Barnes-Hut [1], които третират телата, който са далече като едно тяло с център, центъра на тежестта на тази група тела. В рамките на този проект открих и Fast Multipole Method (FMM), но поради липса на опит с подобни алгоритми и ограниченото време ще се фокусирам само върху наивния метод и Barnes-Hut.

• Наивен (PWI Pairwise interaction)

Най-простиия алгоритъм, с който се запознах е Pairwise interaction (PWI) при него изчисленията са всяко с всяко. Поради тази причина има сложност $O(N^2)$ и при нарастване на броя частици квадратично нараства времето за изчисление на един кадър от симулацията.

За научни симулации това е голям недостатък, но за този проект това е голямо предимство, защото може лесно да се види разликата между последователната и паралелната обработка.

• Barnes-Hut (BH)

Barnes-Hut алгоритъма е популярен алгоритъм, който се масово се използва за n-body симулации. Този алгоритъм йерархично разделя пространството около телата на кутийки (клетки) докато има само едно тяло във всяка клетка. Това позволява алгоритъма бързо да апроксимира силите, които действат върху частиците [2].

Тази апроксимация работи на база на разстоянието между частицата за която се изчисляват силите и разстоянието между нея и останалите клетки. Ако центъра на тежестта на дадена клетка е разстояние двойно колкото размера на най-дългата страна на клетката, всички тела в тази клетка се третират като едно тяло. Силата между частицата и клетката използва като тежест сумата на всички тегла в клетката и център, центъра на тежестта. Ако разстоянието не изпълнява това условие, рекурсивно се влиза в подклетките докато условието не се изпълни или докато има само едно тяло в клетката. [1]

Ако има само едно тяло в клетката, се получава директно взаимодействие между две частици.

Както споменах в предишната точка, наивния метод има сложност $O(N^2)$, за разлика от това, благодарение на апроксимацията, която се извършва с това разделение на пространството, BN алгоритъма има сложност $O(N \log N)$ [4].

Важно нещо, което трябва да се отбележи, е че при симулация на малък брой тела, не се наблюдава подобно ускорение. Този метод се използва при по-голям брой частици. В рамките на този проект, предимствата от BN алгоритъма се наблюдават при повече от 1000 тела.

• Други

Тук искам да спомена и други алгоритми, които са по-сложни и заради липса на опит в тази сфера не успях да разгледам по подробно. Това са FMM(Fast Multipole Method) и (PMTA) Parallel Multipole Tree Algorithm. [4]

• Алгоритми за придвижване на частиците

Другия алгоритъм, който е важен за реализацията на N-Body симулацията е алгоритъма, който се използва за интеграция. Този алгоритъм отговаря за движението на частиците.

При проучването на проблема попаднах само на един подходящ алгоритъм който използва схемата „leap-frog“ или (kick-drift-kick) където изчисленията за движение се разделят на две. За всяка стъпка във времето Δt , всяка частица получава „половин побутване“ (kick)**[8]**. Това побутване се описва с формулата:

$$\mathbf{v}_i = \mathbf{v}_i + \frac{\Delta t}{2} \times \mathbf{a}_i$$

След побутване следва дрифт. При тази стъпка се получава преместването на частицата на база на скоростта, ѝ:

$$\mathbf{r}_i = \mathbf{r}_i + \Delta t \times \mathbf{v}_i$$

След този дрифт се изчисляват наново силите, които действат на частиците използвайки алгоритъм (PWI или BH). След изчислението на силите частиците получават ново побутване.

2.3 Анализ на екземпляри

• PEPC [5]

PEPC е съкращение за “Pretty Efficient Parallel Coluombsolver”. Този труд първоначално е създаден за mesh-free (безмрежово) моделиране на плазмени системи, но с времето проекта се е разширил да се динамика на флуидите, плазмени симулации и гравитационни симулации.

Този труд е базиран на труда на Salmon **[7]**, който използва ‘hashed octree (HOT) подход с фиксиран многополюсна експанзия (до 4 полюса)

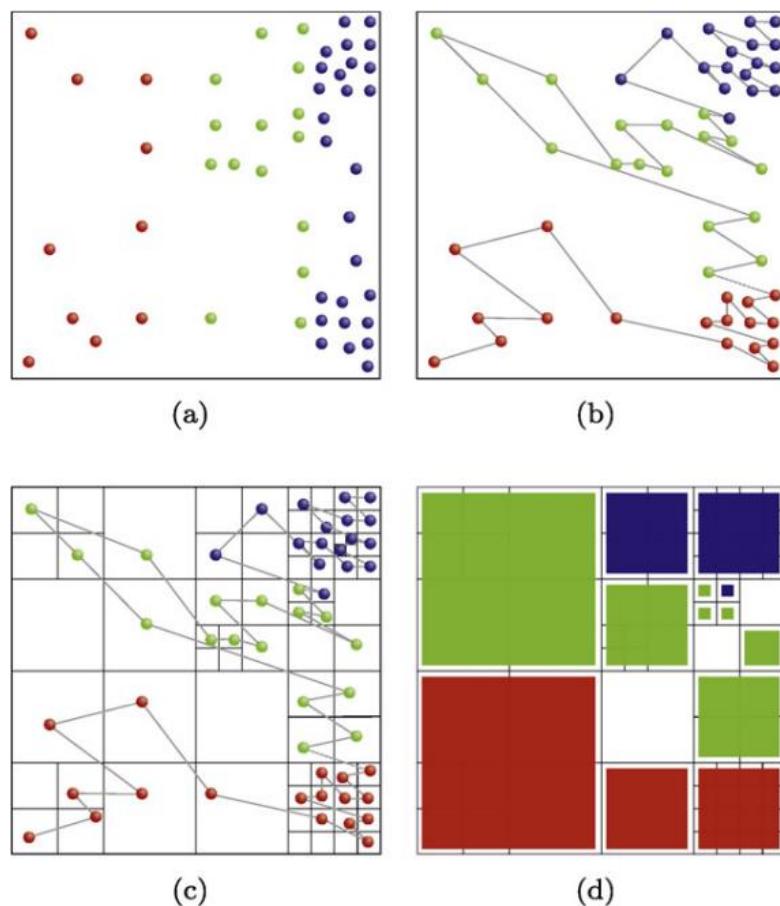
PEPC е разработен на езика c++ и използва MPI и Posix (Pthreads).

Алгоритъма се разделя на 3 части: декомпозиция на работната област и разпределяне на частиците, построяване на дървото и обхождане на дървото.

Прост двуизмерен пример за декомпозицията може да се види на Фиг. 1, като се използва Мортонов код за съпоставяне на многомерни данни с едно измерение, като се запазва местоположението на точките с данни. След декомпозицията всеки процес създава локално дърво следвайки HOT схемата на Salmon.

Възлите на клоните на дървото служат като входни точки за нелокални дървета. По този начин, всеки процес може да поиска информация за другите дървета в процеса на обхождане.

РЕРС е разработен за работи на суперкомпютъра JUGENE.



Фиг. 1 Създаване на дървото при РЕРС

Характеристики на JUGENE

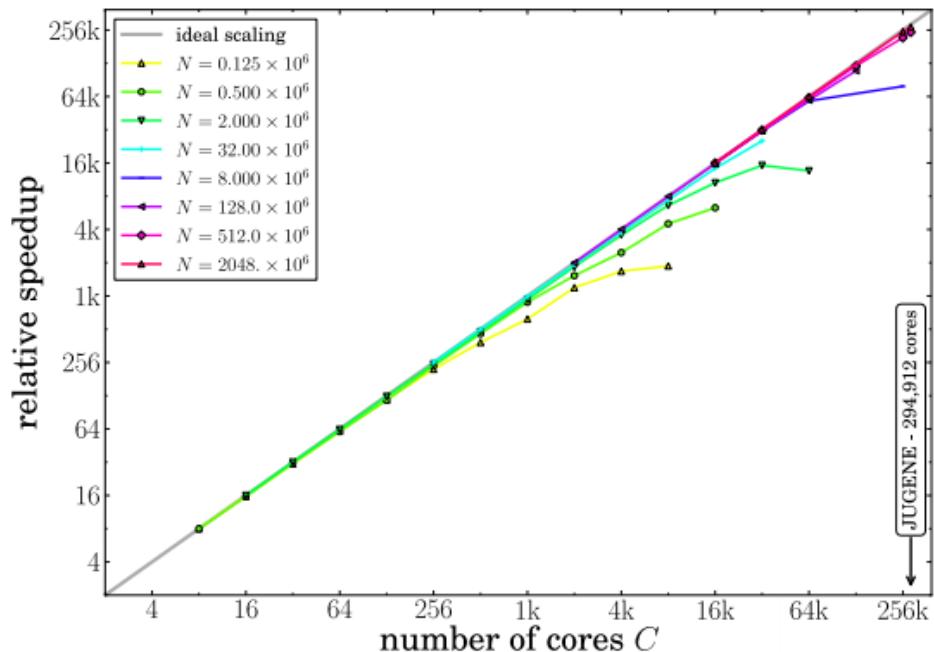
Брой ядра: 294,912

RAM: 144 TB

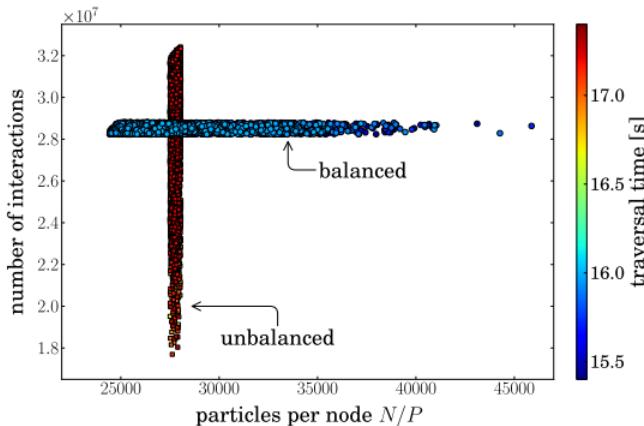
Памет: 6PB

Постигнати резултати

Благодарение на иновативния алгоритъм РЕРС има хубави ускорения, както се вижда на Фиг.



На Фиг. 2 се показва как се балансират частиците и по колко частици се падат на възел.



Фиг. 2 Разпределение на частиците

•PSP (Problem Space Promotion) [6]

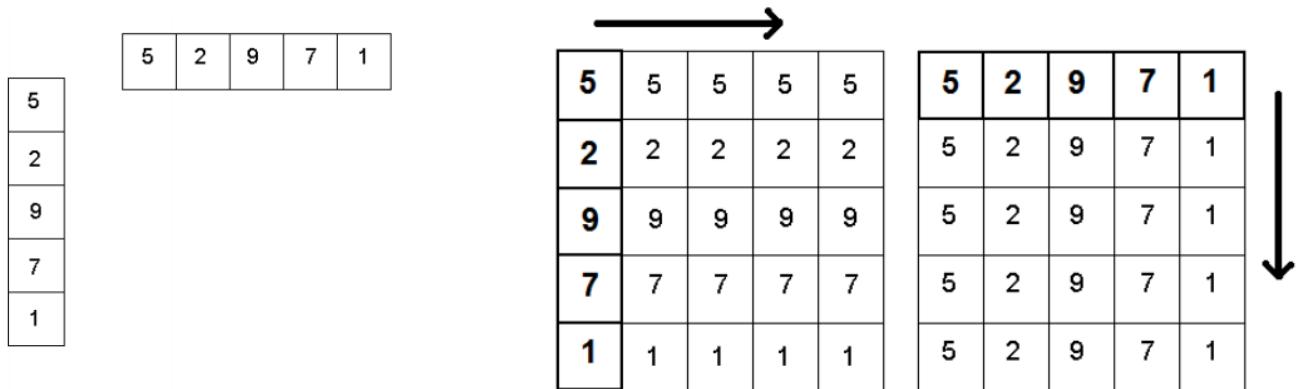
Този труд е по-различен от предишния и други като Salmon [7], защото използва езика ZPL – език в специално разработен във Вашингтонския университет за паралелно програмиране.

Предимството на PSP е, че може да създаде N-Body симулация с добър паралелизъм. PSP намалява проблемите с комуникацията и синхронизацията като увеличава размерността на пространството. В случая на N-Body проблема това означава масива с тела да стане двуизмерен, NxN масив в който всеки ред е копие на оригинала. Създавайки такъв двуизмерен масив премахва нуждата за цикъл, който да обхожда всички двойки тела.

След създаване на този масив (вече матрица) се създава и транспонираната матрица. След което елемент по елемент се извършва операция, на която резултата се записва в трета матрица.

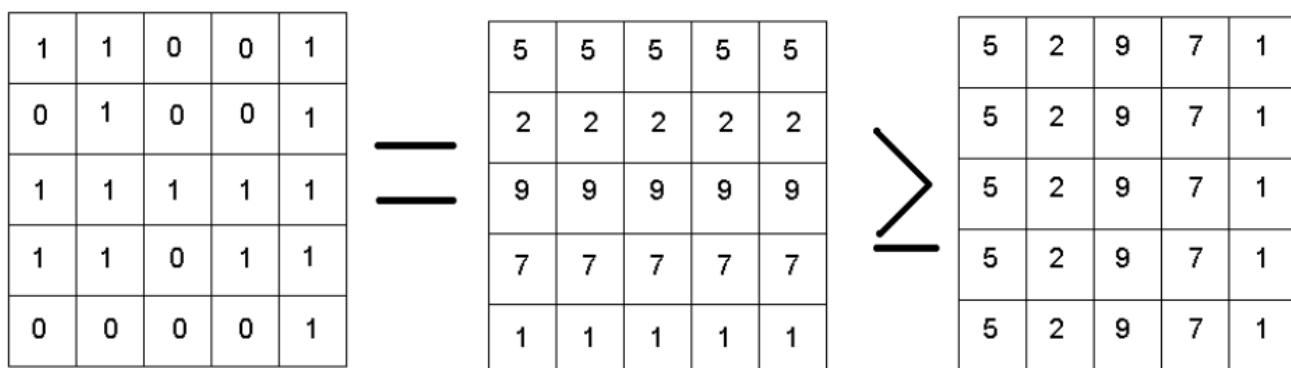
Последната стъпка е редукция, която сумира стойностите по колони и създава нов едномерен масив, който съдържа резултатите от операцията.

Нагледен пример може да се види на Фиг. 5. В примера се използва операцията по-голямо или равно.



Фиг. 3 Оригинален и транспониран масив

Фиг. 4 Създаване на матриците



Фиг. 5 Извършване на операция

1	1	0	0	1
0	1	0	0	1
1	1	1	1	1
1	1	0	1	1
0	0	0	0	1

3	4	1	2	5
---	---	---	---	---

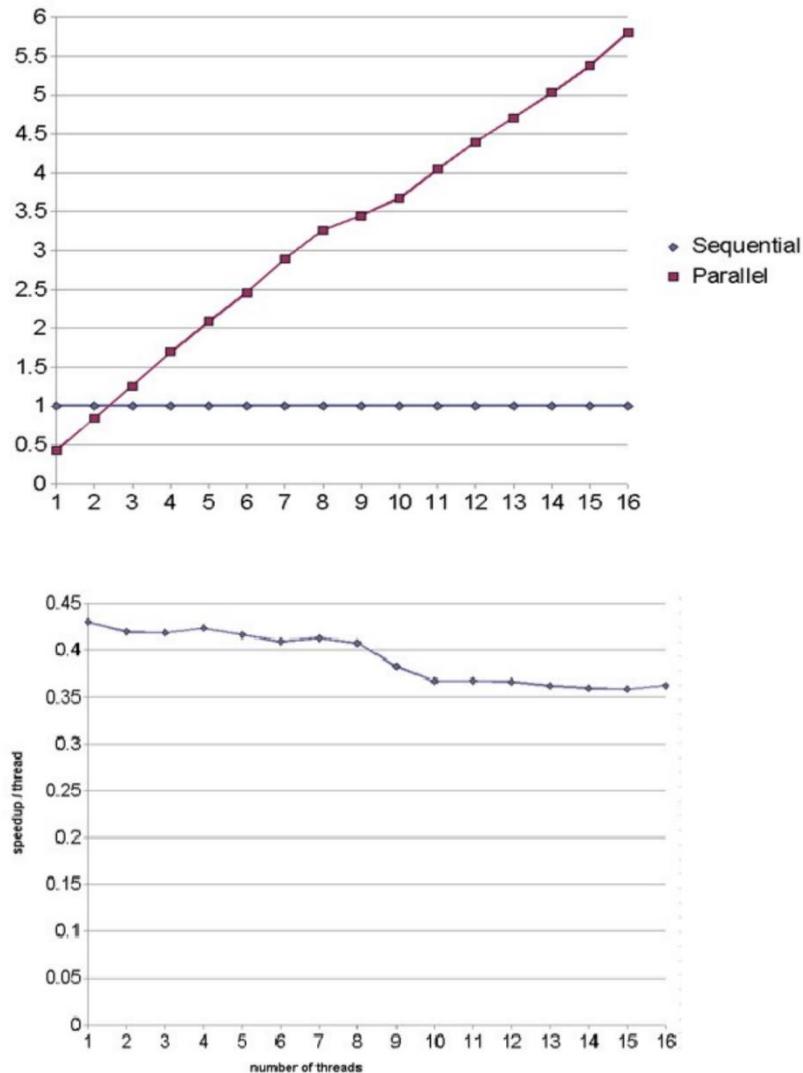
Фиг. 6 Стъпката на редукция (получаване на крайния резултат)

Характеристики на тестовата машина

Sun Fire T2000 Server

- 8 core 1.2GHz UltraSPARC T1 processor
- 16GB DDR2 memory (16 * 1GB DIMMs)
- 2 * 73GB 2.5" 10K rpm SAS hard disk drives
- Solaris 10 OS

Постигнати резултати



2.4 Избор на алгоритми

Въпреки, че всички екземпляри използват алгоритми различни от Pairwise interaction, този проект го имплементира заедно с ВН алгоритъма.

Това решение беше взето, защото при наивния метод може лесно да се покаже какво влияние има паралелната обработка, докато ВН алгоритъма може да покаже как последователни части в паралелна програма оказват влияние върху паралелизма на програмата.

3. Реализация

3.1 Избор на език и среда за разработка

За език проекта използва C#, а средата на за разработка е Visual Studio 2019. Използвам C#, защото с него имам най-много опит и езика има добри възможности за паралелизъм.

Друга причина да използвам C# е възможностите за визуализация и взаимодействие с потребителите. За създаване на интерфейса използвам WPF, а за визуализацията използвам библиотеката SkiaSharp за генериране на отделните кадри на симулацията и библиотеката Xade.FFMPEG използвам за генериране на видео от отделните кадри.

Интерфейса се реализира с помощта на WPF(Windows Presentation Foundation).

3.2 Описание на имплементираните алгоритми

- Pairwise interaction (Наивен подход)**

Този алгоритъм се имплементира като два вложени цикъла при последователната обработка. При паралелната обработка общия масив с частици се разделя равномерно на броя нишки и всяка нишка изпълнява последователния алгоритъм с два вложени цикъла.

- BH (Barnes-Hut)**

Имплементирания BH алгоритъм е съвсем базов. Състои се от три фази: Рекурсивна декомпозиция на пространството докато има само една частица във всяка клетка, изчисляване на взаимодействията на клетките чрез обхождане на дървото и последната стъпка е преместването на частиците в зависимост от силите, които действат върху тях.

От тези стъпки обхождането и преместването на частиците, докато декомпозицията е последователна. В процеса на събиране на информация, късно попаднах на паралелен алгоритъм за построяване на дървото. Този алгоритъм е Ортогонално балансиране (ORB) и се използва от Salmon [7], но поради ограниченото време не успях да го изследвам и имплементирам.



3.3 Описание на реализацията

• Кратко описание

Реализацията на проекта се разделя на две основни части, интерфейса и библиотека (class library). Интерфейса съдържа единствено логиката за обработка на входните данни, създаване на инстанция на основния клас от библиотеката (ProgramManager), извикване на функции от този клас и визуализиране на информация относяща се до статуса на изпълняващата се операция. Комуникацията между библиотеката и интерфейса е чрез извикване на функции (от страна на интерфейса) и генериране на събития (events) от страна на библиотеката.

Благодарение на това разделение, библиотеката може да се използва в бъдеще отделно от сегашния интерфейс..

Раздела „Упътване за употреба“ има подобно описание на интерфейса и всички функции.

• Генериране на частиците

Програмата разполага с пространство за симулация с размери по X: 737px, по Y: 979px. При генериране на частиците, се създава матрица с размерите на симулационното пространство, където всяка „клетка“ представлява един пиксел.

Генерирането позициите на частиците е произволно и при създаване на всяка една частица се извършва проверка дали няма частица на тази позиция с помощта на матрицата, ако има се генерира нова позиция.

Това е необходимо, за да не се генерират две частици в една и съща точка, защото това би нарушило структурата на дървото, което се използва при ВН алгоритъма.

• Симулация

В тази точка, ще опиша процеса на изчисляване и визуализиране на един кадър от симулацията.

Изчисления

За да се създаде един кадър трябва да се направят следните изчисления:

- Изчисляване на силата на привличане между двета обекта
- Изчисляване на разстоянието между два обекта
- При ВН алгоритъма се проверява дали разстоянието между частица и клетка е повече от два пъти дължината на страната на клетката. Като в случая, поради правоъгълната форма на клетките като „дължина на страна“ се взима по-дългата страна.
- Изчисляване движението на частицата. Това е изчислението на позицията, която частицата ще заеме в следващия момент от време.

Стъпки

Изчисленията описани в предходната точка трябва да се изпълнят в определена последователност. Диаграмата на Фиг. 7 показва стъпките за създаване на симулация използвайки наивния алгоритъм а на Фиг. 8 се показват стъпките при създаване на симулация използвайки ВН алгоритъма.

Стъпки при Pairwise interaction

1

- Изчисляване на скоростите на частиците
- Изчисляване на новия център на всяка частица
- Проверка дали някоя от частиците се е ударвала в някоя от стените

2

- Изчисляване на силите между телата използвайки Pairwise Interaction

3

- Отново се преизчисляват скоростите на телата, използвайки новите ускорения получени при стъпка 2.

Фиг. 7 Стъпки при PWI

Стъпки при Barnes-Hut

1

- Изчисляване на скоростите на частиците
- Изчисляване на новия център на всяка частица
- Проверка дали някоя от частиците се е ударвала в някоя от стените

2

- Разделяне на пространството на клетки докато има само 1 частица във всяка клетка.

3

- Отново се преизчисляват скоростите на телата, използвайки новите ускорения получени при стъпка 2.

4

- Корена на дървото генерирано в търка 3 се премахва и се подготвя за ново разделение. Това се прави при всеки кадър.

Фиг. 8 Стъпки при BH

Генериране на резултати

Резултатите от симулацията са .png изображения с размера на симулационното пространство (737x979). Запазват се в директория като се именуват с поредни номера.

• Генериране на видео

Обосновка

Реших да генерирам кадри и видео вместо да показвам симулацията в „реално време“ защото поради ограничения на технологията за интерфейса, е невъзможно да се направи гладка симулация и в същото време тази симулация да се запази във файл. Тъй като симулацията е единствения начин, по който човек може да се види резултата от кода за мен беше важно крайния резултат да е гладка анимация, която да е 60 кадъра в секунда.

Библиотеката FFmpeg

За генериране на видеото от поредица от изображения използвам външната библиотека Xade.FFmpeg, която е адаптация на библиотеката FFmpeg за C#. Реших да използвам външна библиотека, защото прецених, че задачата за генериране на видео е извън обхватата на този проект.

С помощта на тази библиотека с един ред код всичките генериирани изображения се преобразуват във .mp4 видео което може да се гледа в 60 кадъра в секунда.

• Автоматично тестване

За улеснение и точност тестването на програмата е автоматично.

Автоматичното тестване измерва за колко време се изчислява един кадър от симулацията при различен брой нишки.

Автоматичното тестване може да се настрои да тества един от двата алгоритъма, колко нишки да могат да се използват, вида на нишките (дали са ръчно направени или се използва TPL библиотеката на C#) и колко пъти се повтаря даден тест за да филтрира ефекта на фоновото натоварване.

Измерване на време

Времето за изпълнение се измерва с вградения клас на C# „Stopwatch“. Таймера се пуска точно преди да започнат изчисленията за един кадър и се спира веднага след края. Времето за рисуване не се отчита, защото при автоматично тестване не се рисуват кадрите.

Генериране на резултати

Резултатите от автоматичното тестване са време за изпълнение, ниво на паралелизъм и ефективност. Данните се визуализират на три отделни графики. По-подробна информация може да се види в раздел 4 „Резултати“

3.4 Структура на програмата

•Обща диаграма

UI

Class Library (PNB_Lib)

Program Manager

Съдържа и управлява програмата. Създава инстанция на QuadTree. Инстанция на ProgramManager се прави от кода на интерфейса

QuadTree

В този клас се съдържа логиката на програмата. QuadTree класа се грижи за:

- Генериране на частици
- Симулацията
- Автоматичното тестване
- Генериране на резултати от симулацията и автоматичното тестване
- Управлява нишките които се използват при симулации и тестване
- Управлява дървото което разделя пространството при използване на BN алгоритъма

Node

Клетка от дървото. Съдържа функции за:

- Добавяне на частица в дървото рекурсивно
- Управления на информацията за инстанцията на клетката

Particle

Модел на частицата. Използва се да пази информация за частиците в системата. Съдържа следната информация:

- Центъра на частицата
- Маса на частицата
- Информация за ускорението в посоките x и y
- Информация за скоростта в посоките x и y
- Цвета на частицата

Библиотека LiveCharts

Външа библиотека която генерира таблици.

Използва се за визуализация на резултатите от автоматичното тестване.

Библиотека Xade.FFMPEG

Външа библиотека която е адаптация на FFMPEG за C#.

Използва за генериране на видео от отделните кадри, които са генериирани от Симулацията.

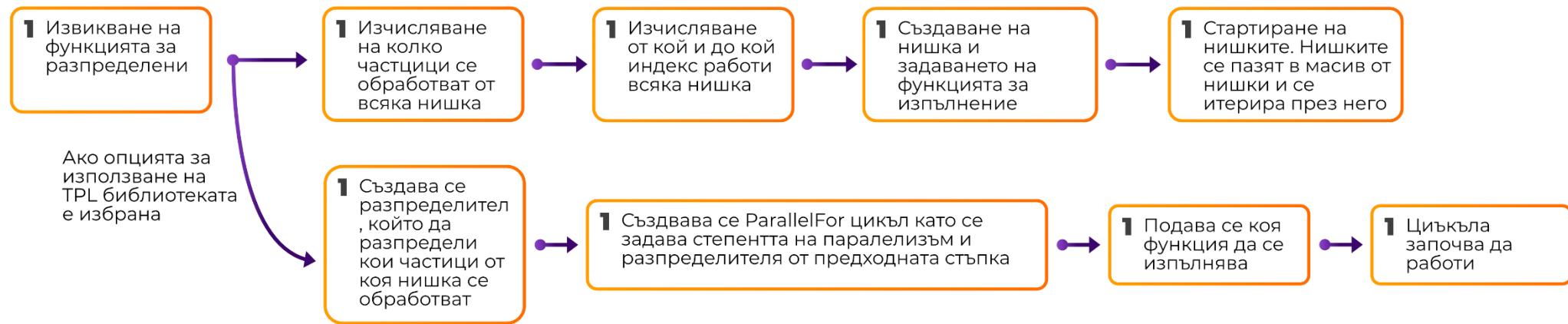
- Диаграма за разпределение на пространството

Рекурсивно разделяне на пространството



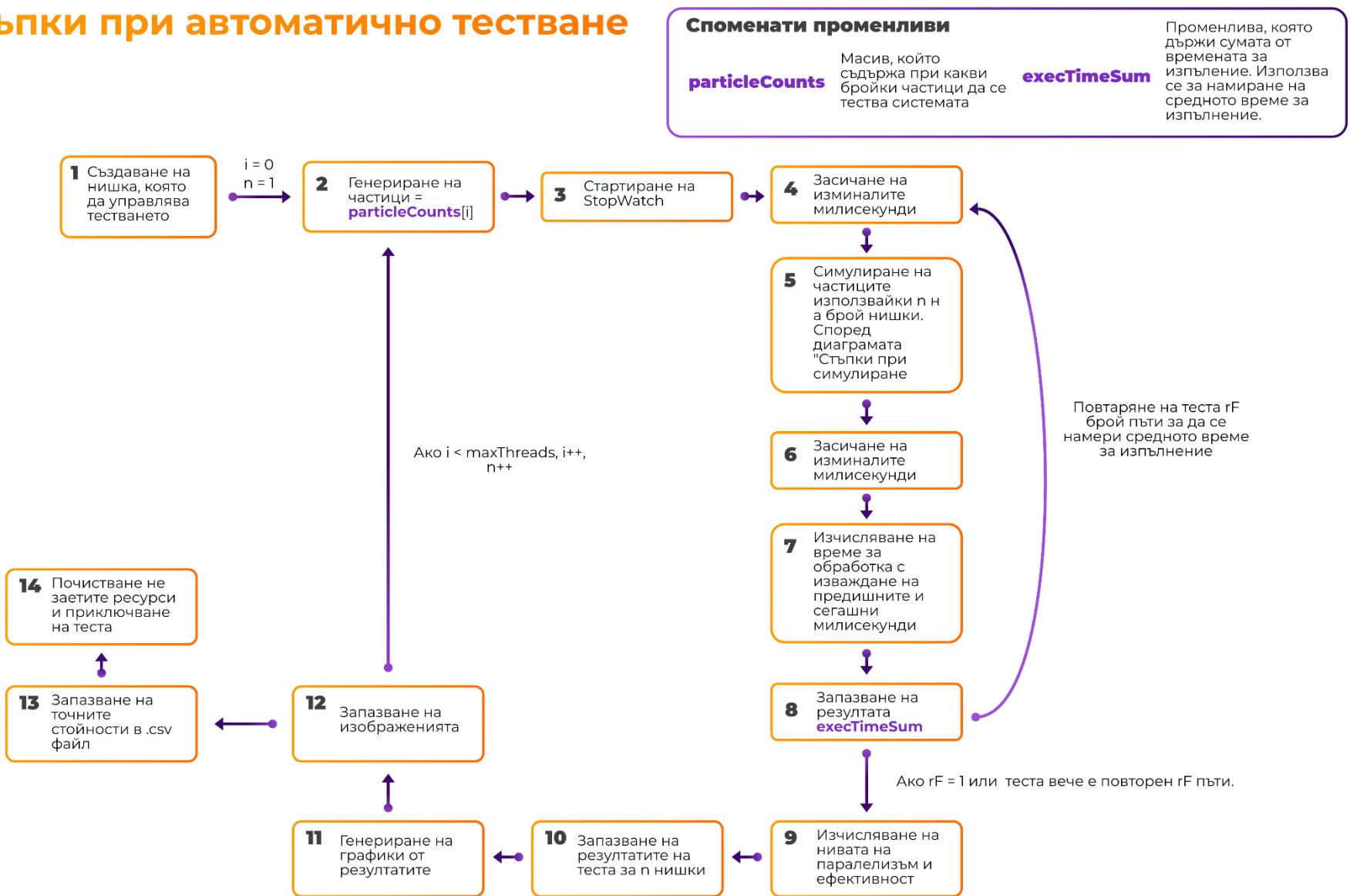
- Диаграма за разпределение на работата

Стъпки при разпределяне на работата



• Диаграми за автоматично тестване

Стъпки при автоматично тестване

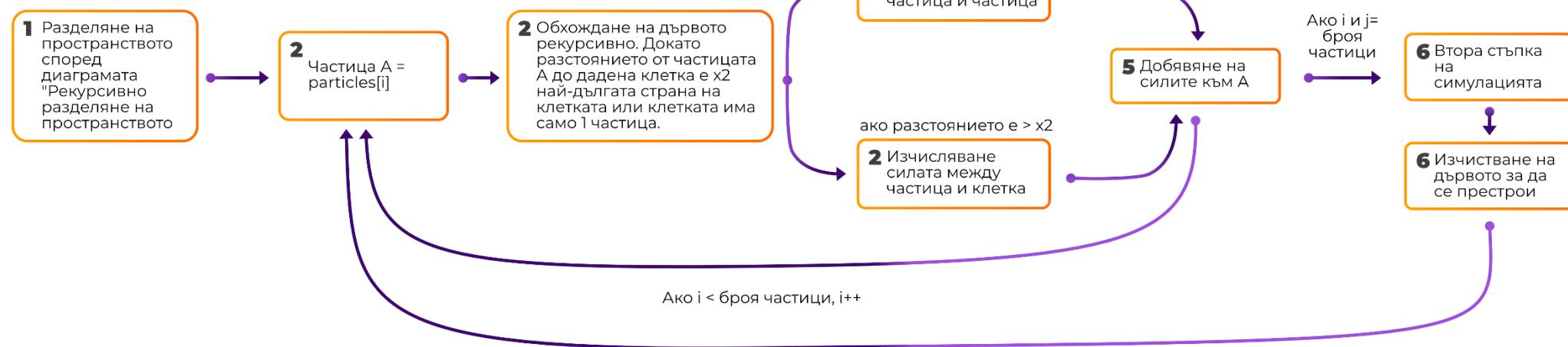


- Диаграми на алгоритмите

Диаграма на PWI алгоритъма



Диаграма на ВН алгоритъма



• Диаграма на стъпките за симулация

Стъпки при симулиране (генериране на 1 кадър)



4. Резултати

4.1 Тестов план

Тестването на програмата става автоматично. Параметрите, които се задават няколко параметъра: бройките частици, с които ще се тества системата, максималния брой нишки, които да се тестват, колко пъти да се повтаря тест за определен брой нишки. Резултатите се запазват в csv за точните стойности и .png за графиките.

Проведох тестове за PWI алгоритъма при 1500, 2500, 3500, 5000 тела, като над 5000 времето за тестване стана прекалено дълго. Избегнах стойности под 1500 за да се види по-добре как паралелизма оказва влияние върху работата на програмата.

Тестовете за BH алгоритъма проведох при 1000, 1500 и 3500 частици. При по-големи стойности

- **Тестова машина**

Процесор: Intel® Core™ i7-8750H @2.2Ghz – 6 физически ядра

RAM: 32GB(31.9 usable)

OS: 64-bit Windows

L1 cache: 384KB, L2 cache: 1.5MB, L3 cache: 9MB

4.2 Получени резултати

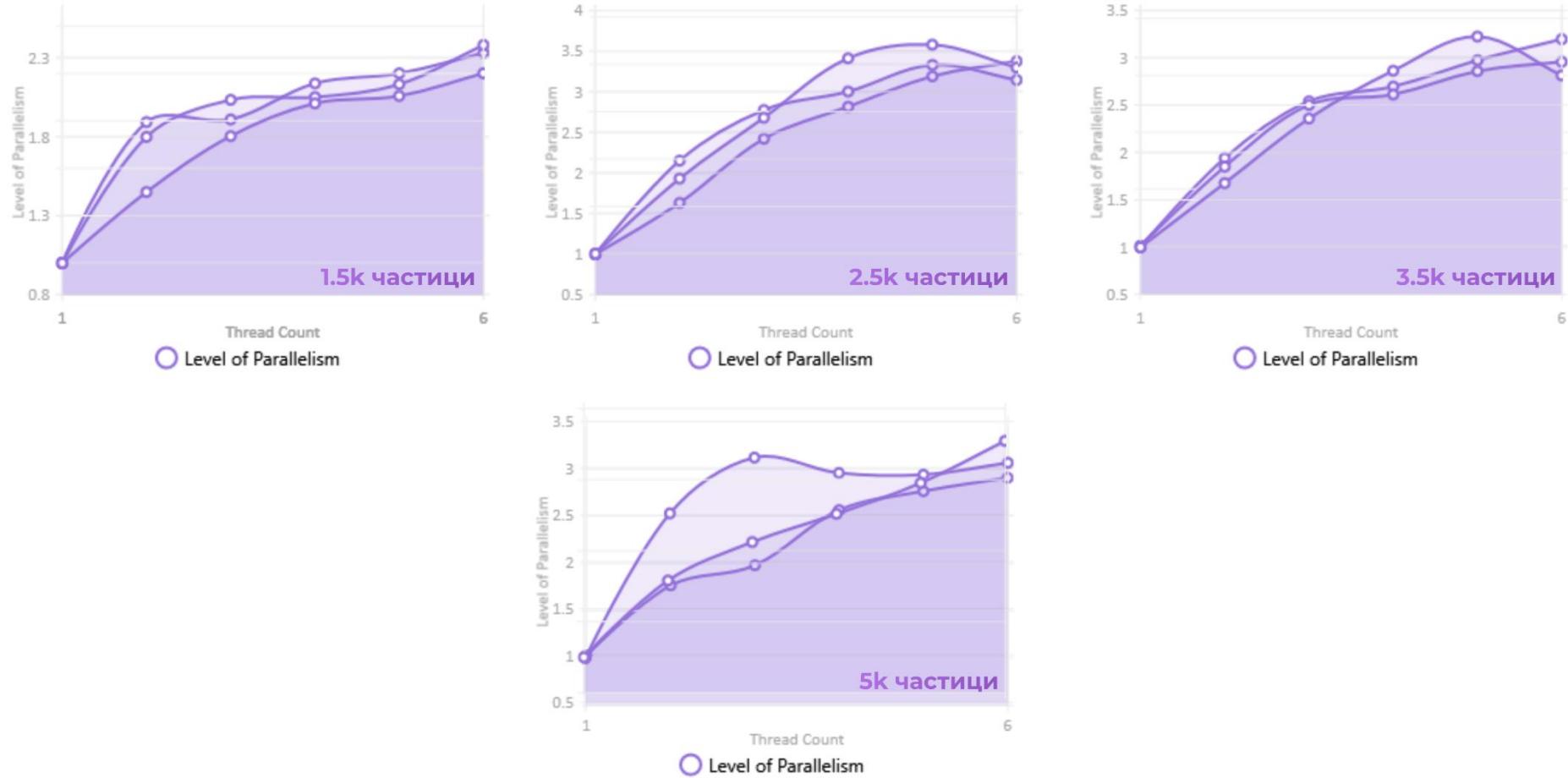
P = брой нишки, **tT** = тип на нишките (custom/TLP), **p**: брой частици, **rF**: колко пъти се тества кода при р нишки.

Custom нишки – ръчно конфигуриирани стартирани, TPL – използване на вградена библиотека за паралелна обработка

Получени резултати (Паралелен PWI)

Алг.	p	tT	n	rF	T _p ⁽¹⁾	T _p ⁽²⁾	T _p ⁽³⁾	T _{p min}	S _p ⁽¹⁾	S _p ⁽²⁾	S _p ⁽³⁾	S _{p max}	E _p ⁽¹⁾	E _p ⁽²⁾	E _p ⁽³⁾
PWI	1	Custom	1500	2	1388.81	1377.67	1230.52	1230.52	1	1	1	1	1	1	1
~	2	~	~	~	733.86	765.86	703.28	703.28	1.89	1.79	1.74	1.89	0.94	0.89	0.87
~	3	~	~	~	727.31	677.46	525.88	525.88	1.90	2.03	2.33	2.33	0.63	0.67	0.77
~	4	~	~	~	649.67	671.84	458.18	458.18	2.13	2.05	2.68	2.68	0.53	0.51	0.67
~	5	~	~	~	630.77	646.04	445.25	445.25	2.20	2.13	2.76	2.76	0.44	0.42	0.55
~	6	~	~	~	595.92	578.61	409.65	409.65	2.33	2.38	3.00	3.00	0.38	0.39	0.50
~ 1 ~ 2500 ~															
~	1	~	2500	~	4374.10	4696.63	3590.95	3590.95	1	1	1	1	1	1	1
~	2	~	~	~	2684.67	2180.00	2002.15	2002.15	1.63	2.15	1.79	2.15	0.81	1.08	0.90
~	3	~	~	~	1807.87	1690.68	1474.10	1474.10	2.42	2.78	2.44	2.78	0.81	0.93	0.81
~	4	~	~	~	1554.47	1562.66	1172.11	1172.11	2.81	3.01	3.06	3.06	0.70	0.75	0.77
~	5	~	~	~	1371.62	1409.35	1119.47	1119.47	3.19	3.33	3.21	3.33	0.64	0.67	0.64
~	6	~	~	~	1296.18	1491.45	1210.66	1210.66	3.37	3.15	2.97	3.37	0.56	0.52	0.49
~ 1 ~ 3500 ~															
~	1	~	3500	~	7886.97	7189.96	6946.72	6946.72	1	1	1	1	1	1	1
~	2	~	~	~	4704.81	4237.45	3579.00	3579.00	1.68	1.70	1.94	1.94	0.84	0.85	0.97
~	3	~	~	~	3344.63	3209.59	2735.35	2735.35	2.36	2.24	2.54	2.54	0.79	0.75	0.85
~	4	~	~	~	2754.69	3083.38	2576.11	2576.11	2.86	2.33	2.70	2.86	0.72	0.58	0.67
~	5	~	~	~	2446.07	2833.78	2336.02	2336.02	3.22	2.54	2.97	3.22	0.64	0.51	0.59
~	6	~	~	~	2802.93	2744.86	2172.98	2172.98	2.81	2.62	3.20	3.20	0.47	0.44	0.53
~ 1 ~ 5000 ~															
~	1	~	5000	~	17977.1	31821.2	13721.2	13721.2	1	1	1	1	1	1	1
~	2	~	~	~	8642.06	7841.67	7843.07	7841.67	2.08018	4.05796	1.74947	4.05796	1.04009	2.02898	0.87473
~	3	~	~	~	6857.6	6076.74	6985.64	6076.74	2.62148	5.23657	1.9642	5.23657	0.87383	1.74552	0.65473
~	4	~	~	~	5955.02	6477.6	5372.99	5372.99	3.01881	4.9125	2.55374	4.9125	0.7547	1.22813	0.63843
~	5	~	~	~	5205.04	6527.89	4978.18	4978.18	3.45378	4.87466	2.75626	4.87466	0.69076	0.97493	0.55125
~	6	~	~	~	4442.91	6208.59	4731.6	4442.91	4.04623	5.12535	2.89991	5.12535	0.67437	0.85423	0.48332

Сравнение на паралелизма на тестовете (При паралелен PWI)



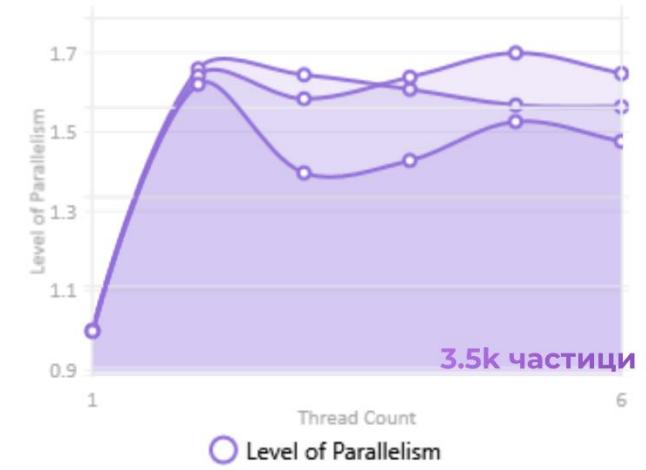
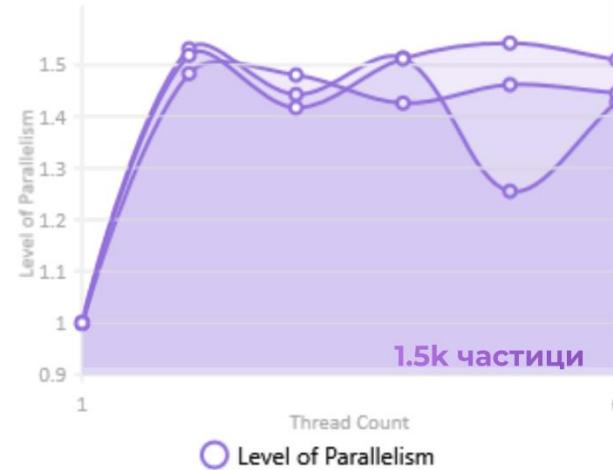
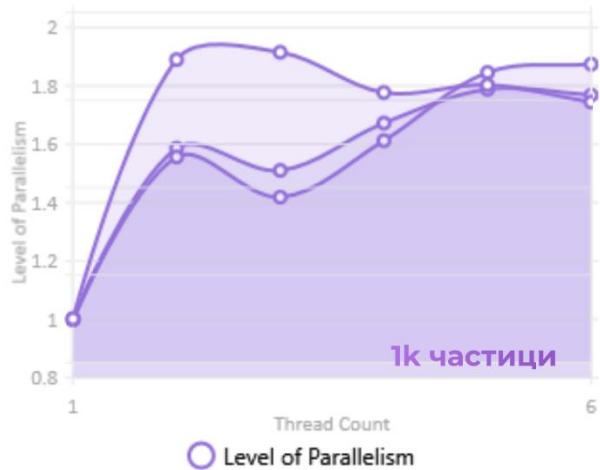
Фиг. 9 Сравнение на паралелизма

Получени резултати (Паралелен ВН)

Алг.	p	tT	n	rF	T _p ⁽¹⁾	T _p ⁽²⁾	T _p ⁽³⁾	T _{p min}	S _p ⁽¹⁾	S _p ⁽²⁾	S _p ⁽³⁾	S _{p max}	E _p ⁽¹⁾	E _p ⁽²⁾	E _p ⁽³⁾	
VH	1	Custom	1000	2	4959.95	5933.06	5317.48	4959.95	1	1	1	1	1	1	1	
~	2	~	~	~	3114.83	3811.60	3352.00	3114.83	1.59	1.56	1.59	1.59	0.80	0.78	0.79	
~	3	~	~	~	3082.27	4181.47	3520.69	3082.27	1.61	1.42	1.51	1.61	0.54	0.47	0.50	
~	4	~	~	~	3269.13	3679.45	3179.07	3179.07	1.52	1.61	1.67	1.67	0.38	0.40	0.42	
~	5	~	~	~	3232.14	3213.62	2973.16	2973.16	1.53	1.85	1.79	1.85	0.31	0.37	0.36	
~	6	~	~	~	3316.93	3164.98	3006.15	3006.15	1.50	1.87	1.77	1.87	0.25	0.31	0.29	
~	1	~	1500	~	7466.76	6965.13	6951.87	6951.87	1	1	1	1	1	1	1	
~	2	~	~	~	4916.96	4692.94	4540.77	4540.77	1.52	1.48	1.53	1.53	0.76	0.74	0.77	
~	3	~	~	~	5269.09	4702.01	4819.44	4702.01	1.42	1.48	1.44	1.48	0.47	0.49	0.48	
~	4	~	~	~	4938.98	4881.79	4595.33	4595.33	1.51	1.43	1.51	1.51	0.38	0.36	0.38	
~	5	~	~	~	4842.08	4762.45	5538.52	4762.45	1.54	1.46	1.26	1.54	0.31	0.29	0.25	
~	6	~	~	~	4947.22	4816.70	4866.98	4816.70	1.51	1.45	1.43	1.51	0.25	0.24	0.24	
~	1	~	3500	~	17621.62	17174.24	16445.56	16445.56	1	1	1	1	1	1	1	
~	2	~	~	~	11351.57	10933.14	9911.74	9911.74	1.55	1.57	1.66	1.66	0.78	0.79	0.83	
~	3	~	~	~	13014.22	11300.75	10002.03	10002.03	1.35	1.52	1.64	1.64	0.45	0.51	0.55	
~	4	~	~	~	12752.78	10951.20	10228.94	10228.94	1.38	1.57	1.61	1.61	0.35	0.39	0.40	
~	5	~	~	~	11998.13	10585.24	10487.72	10487.72	1.47	1.62	1.57	1.62	0.29	0.32	0.31	
~	6	~	~	~	12368.84	10894.42	10516.12	10516.12	1.42	1.58	1.56	1.58	0.24	0.26	0.26	

Сравнение на паралелизма на тестовете

(При паралелен ВН)



4.3 Анализ

Тестовете се извършват по описанието в раздела „Автоматично тестване“. Времето в таблиците е в милисекунди.

Таблица с характеристики

Система	JUGENE	SunFire	Моята
Процесор	?	e 1.2GHz UltraSPARC T1	Intel® Core™ i7-8750H @2.2Ghz
Брой ядра	294,912	8	6
RAM	144 TB	16GB	32GB
Памет	6PB	146GB	1TB
L1	?	?	384KB
L2	?	?	1.5MB
L3	?	?	9MB

• Резултати при PWI

Резултатите при паралелния PWI ясно показват как по-големият брой нишки водят до ускорение. Интересно е да се отбележи, че при третия тест се наблюдава по-бързо изпълнение на кода. Не мога да потвърдя точните фактори, които доведоха до това, но е възможно че преди провеждането на този тест да съм затворил няколко програми и да съм намалил фоновото натоварване.

Вижда се от графиките, че ускорението е сравнително консистентно между различните тестове. Няма линейно ускорение, според мен това се дължи на факта, че има части от кода, които са последователни и това оказва влияние.

Друг фактор, който има силно влияние е фоновото натоварване. По време на тестовете на машината работеха и други процеси. При премахване на това натоварване мисля, че ще се наблюдават по-добри резултати.

Графиките на Фиг. 9 съм ги приравнял да започват от една позиция, за да се види по-добре разликата между тестовете.

• Резултати при ВН

Резултатите при ВН според мен за разочароваващи, но очаквани. Най-тежката част от алгоритъма – декомпозицията на пространството в този проект не е паралелна и представлява тежка последователна част от кода в средата на изчислението на един кадър.

Въпреки това, има и ясно ускорение, което се покачва при използването на повече нишки.

При следваща версия на кода, ако се имплементира и паралелна декомпозиция, може да се постигнат по-добри резултати.

Поради тези слаби ускорения, в този случай ВН алгоритъма показва какво влияние може да има една последователна част от кода. Дори и всички останали изчисления да са паралелни, ако има една тежка операция, която е последователна, паралелизма рязко спада.

4.4 Анимация

Генерираните анимации могат да се изгледат на следния тук:

5 частици (първа анимация)

<https://www.youtube.com/watch?v=CkZEjVyKGoo>

100 частици

https://youtu.be/_uPe-pi0fCo

1000 частици

https://www.youtube.com/watch?v=_J8Hj4qGN6A

60 частици с визуализация на дървото при ВН

<https://youtu.be/HC-l2mcnBiE>

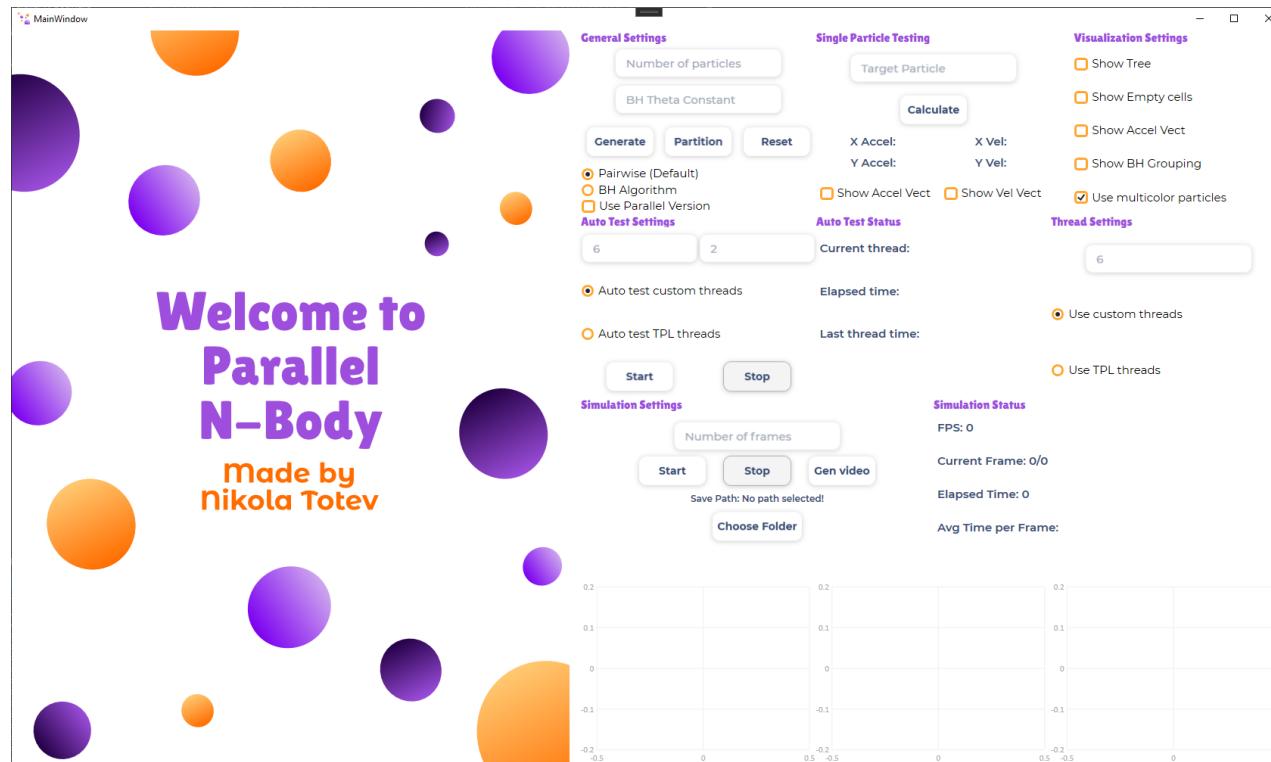
5. Упътване за употреба

В този раздел ще покажа екранни снимки, че опиша интерфейса и всичките важни елементи от него. Описвам и видовете настройки и какво правят, както и стъпките за извършване на най-важните функции.

5.1 Интерфейса

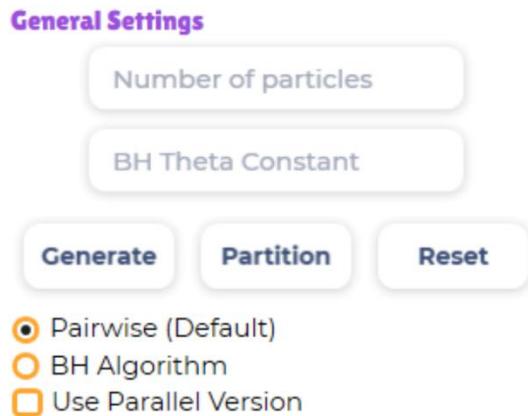
Изображението в лявата част се използва като място за визуализация на частици.

Разделите в дясната част съдържат различните настройки. Тези раздели са описани в следващата точка.



5.2 Видове настройки

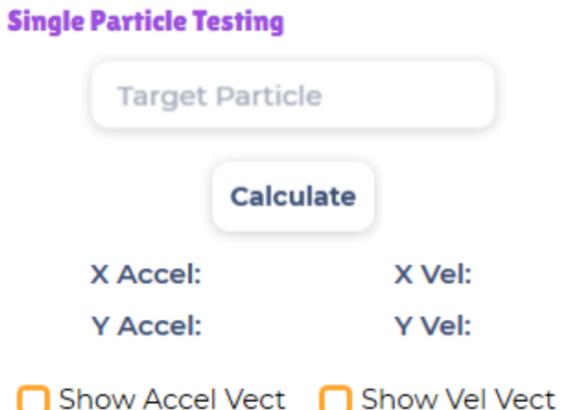
• General Settings



От този раздел потребителите могат да:

- Въвеждат колко частици да се симулират
- Да променят Theta константата за BH алгоритъма
- Да генерираят частици, да декомпозират пространството и да изчистят списъка с частици.
- Да изберат алгоритъма, който се ползва
- Да изберат дали да се използва паралелната версия на избрания алгоритъм

• Single Particle Testing



От този раздел потребителите могат да:

Тестват една частица по избор и да видят скоростта ѝ, ускорението и да визуализират тези два параметъра

• Visualization Settings

Visualization Settings

- Show Tree
- Show Empty cells
- Show Accel Vect
- Show BH Grouping
- Use multicolor particles

От този раздел потребителите могат да:

Достъпят настройките за визуализация:

- Показване на декомпозицията на клетки
- Показване кои клетки от са празни
- Визуализация на силите които действат на частиците
- Показване на групирането на частиците при BH
- Да изберат дали частиците да са с различни цветове

• Auto Test Settings

Auto Test Settings

6 2

- Auto test custom threads
- Auto test TPL threads

Start

Stop

От този раздел потребителите могат да:

- Настройват максиналния брой нишки, които да се тестват
- Колко пъти теста при n на брой нишки да се провежда
- Да избира какъв вид нишки да се използват

• Thread Settings

Thread Settings

6

Use custom threads

Use TPL threads

От този раздел потребителите могат да:

- Настройват колко нишки се използват при симулацията
- Вида нишки, които се използват

• Simulation Settings

Simulation Settings

Number of frames

Start

Stop

Gen video

Save Path: No path selected!

Choose Folder

От този раздел потребителите могат да:

- Изберат колко кадъра да се генерират
- При нужда да прекратят симулацията
- Да генерират видео
- Да изберат къде да се запазват кадрите и видеото

5.3 Стъпки за употреба

• Стъпки за симулация

1. Въвеждане на желания брой частици да се симулират
2. Генериране на частиците
3. Избиране на алгоритъма, който да се ползва
4. Избиране дали да е паралелен
5. Настройване на максималния брой нишки (ако е паралелен)
6. Избиране на типа нишки (Custom/TLP)
7. Въвеждане на броя кадри, които да се симулират
8. Започване на симулацията с бутона “Start” в раздела Simulation Settings
9. Генериране на видео с бутона „Gen Video”

• Стъпки за автоматично тестване

1. Въвеждане на цифра >250 в полето за брой частици
2. Избиране на алгоритъма за тестване
3. Въвеждане на максималния брой нишки за тестване и броя повторения
4. Натискане на бутона „Start” в раздела Auto Test Settings

6. Насоки за бъдещо развитие

Поради ограничено време и липсата на опит в разработването на подобни системи, има още какво може да се подобри в програмата.

Може да се направи по-добра паралелна имплементация на Barnes-Hut която да има паралелна декомпозиция на пространството използвайки (ORB).

Може да се увеличи мястото за симулация, за да има място за по-голям брой частици.

7. Източници

[1] Barnes, J., Hut, P. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature* 324, 446–449 (1986).
<https://doi.org/10.1038/324446a0>

[2] Burtscher, Martin & Pingali, Keshav. (2011). An Efficient CUDA Implementation of the Tree-Based Barnes Hut n-Body Algorithm. *GPU Computing Gems Emerald Edition*. 10.1016/B978-0-12-384988-5.00006-1.

[3] Appel, A.. “An Efficient Program for Many-Body Simulation.” *Siam Journal on Scientific and Statistical Computing* 6 (1985): 85-103.

[4] Blelloch, G. and G. Narlikar. “A practical comparison of N-body algorithms.” *Parallel Algorithms* (1994).

[5] Mathias Winkel, Robert Speck, Helge Hübner, Lukas Arnold, Rolf Krause, Paul Gibbon,A massively parallel, multi-disciplinary Barnes–Hut tree code for extreme-scale N-body simulations,*Computer Physics Communications*, Volume 183, Issue 4, 2012

[6] Parallel N-Body Simulation Using Problem Space Promotion, Brandon Farrell, https://s3-us-west-2.amazonaws.com/www-cse-public/ugrad/thesis/TR08-02_Farrell.pdf

[7] Salmon, John K. (1991) Parallel hierarchical N-body methods. Dissertation (Ph.D.), California Institute of Technology. doi:10.7907/3J5V-VR08.

<https://resolver.caltech.edu/CaltechTHESIS:04112011-113813016>

[8] Create Your Own N-body Simulation (With Python), Philip Mocz, Sep 14, 2020, <https://medium.com/swlh/create-your-own-n-body-simulation-with-python-f417234885e9>