Многопоточная симуляция ткани на центральном процессоре методом, основанном на положении

А.А. Куприн [1]

[1] *ООО «Траектория», ул. Ломоносова, д. 29а, этаж/пом 4/6, г. Жуковский, Московская обл., 140181, Россия*

Аннотация

В статье рассматривается тема многопоточной симуляции ткани методом Extended Position Based Dynamics на CPU. Приведено сравнение распространенных подходов к решению этой задачи и выделены преимущества XPBD. Дано теоретическое обоснование выбранного алгоритма и его связь с классической механикой. Найден способ распределения вычислений между несколькими потоками, основанный на графе ограничений и «фантомных» вершинах. Кроме того, описана система поиска и обработки коллизий полотна с самим собой и твёрдыми телами с применением LBVH дерева и предсказанием контактов. Найденные решения были реализованы в программном коде. После анализа устойчивости и качества модели и скорости работы приложения предложены пути дальнейшего повышения производительности.

Ключевые слова

Компьютерная графика, математическая физика, симуляция ткани, многопоточные вычисления.

Parallel Tissue Simulation on a CPU by a Position-based Method

A.A. Kuprin [1]

[1] *LLC «Trayektoriya», Lomonosova St., 29A, floor/room 4/6, Zhukovsky, Moscow Region, 140181, Russia*

Abstract

The article discusses the topic of multithreaded tissue simulation by Extended Position Based Dynamics on the CPU. A comparison of popular approaches to solving this problem is given and the advantages of XPBD are highlighted. The theoretical justification of the chosen algorithm and its connection with classical mechanics is given. A method of distributing computations between several threads based on a graph of constraints and "phantom" vertices has been found. In addition, a system for searching and processing collisions of the canvas with itself and solids using an LBVH tree and predicting contacts is described. The solutions found were implemented in the program code. After analyzing the stability and quality of the model and the speed of the application, ways to further improve performance are proposed.

Keywords

Computer graphic, mathematical physics, cloth simulation, parallel computing

1. Введение

Возникнув в восьмидесятых годах двадцатого века, компьютерное моделирование ткани развивается по сей день. До недавнего времени, наиболее распространенными методами

ГрафиКон 2023: 33-я Международная конференция по компьютерной графике и машинному зрению, 19-21 сентября 2023 г., Институт проблем управления им. В.А. Трапезникова Российской академии наук, г. Москва, Россия
EMAIL: kuprin.2000@gmail.com (А.А. Куприн)
ORCID: 0009-0006-9879-3855 (А.А. Куприн)

симуляции в реальном времени было представление ткани в виде системы масс и пружин, а также способы, основанные на методе конечных элементов.

В 2006 году был создан алгоритм Position Based Dynamics, позволивший получить приемлемое качество симуляции при высокой скорости. Затем, в 2016 году теми же авторами было предложено улучшение оригинальной модели, получившее название Extended Position Based Dynamics. XPBD значительно повысил точность расчётов. В данной работе изучен метод XPBD, а также один и способов распределения вычислений между несколькими потоками. Кроме того, реализована поддержка коллизий и трение.

2. Теоретическая часть

2.1. Распространенные методы симуляции ткани

2.1.1. Представление ткани в виде системы масс и пружин

Материал представляется в виде множества материальных точек, имеющих определённый вес и соединённых упругими стержнями. На очередном шаге симуляции для каждой из вершин вычисляется суммарный вектор сил и тем или иным способом интегрируется уравнение её движения. Чаще всего пользуются явным методом Эйлера, простым или модифицированным методами Верле, а также методом Рунге-Кутты [1]. При своей простоте этот способ имеет ряд недостатков. Во-первых, физическая модель не имеет привязки к конкретным физическим размерностям. Во-вторых, данный метод чувствителен шагу времени при интегрировании.

2.1.2. Метод конечных элементов

Метод конечных элементов можно использовать для симуляции ткани [2]. Преимуществом этого подхода является высокое качество симуляции, однако, как говорилось выше, на каждом шаге требуется решать СЛАУ большой размерности. Отдельно упомянем метод Бараффа и Виткина [3], обрётший в своё время огромную известность в области визуальных эффектов. Доказывается, что алгоритм Бараффа и Виткина является не чем иным как необычным изложением метода конечных элементов.

2.2. Метод PBD

На рисунке 1 представлен псевдокод метода PBD [4].

```
(1)  for all vertices i
(2)     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1 / m_i
(3)  end for
(4)  loop
(5)     for all vertices i
(6)        v_i = v_i + Δt·w_i·f_ext(x_i)
(7)        p_i = x_i + Δt·v_i
(8)     end for
(9)     GenerateCollisions(x_i, p_i)
(10)    for n_iter
(11)       for all constraints j
(12)          calculate ΔP_j
(13)          P_j = P_j + ΔP_j
(14)       end for
(15)    end for
(16)    for all vertices i
(17)       v_i = (p_i − x_i) / Δt
(18)       x_i = p_i
(19)    end for
(20)    EvaluateFriction(v_i)
(21) end loop
```

Рисунок 1 – Алгоритм метода PBD

Ткань представляется как множество материальных точек $\bar{x}_i, i = \overline{1,n}$, с массой $m_i, i = \overline{1,n}$ на которые действуют ограничения, описывающие внутренние силы ткани. Они задаются в виде скалярных функций $C_j(\bar{X}_j), j = \overline{1,k}$, зависящих от некоторых вершин $\bar{X}_j = [\bar{x}_{j1}{}^T, \bar{x}_{j2}{}^T, \dots, \bar{x}_{jn_j}{}^T]^T, n_j \leq n$. Связи делятся на две категории, одни выполнены, если значение $C_j(\bar{X}_j)$ равно нулю, а другие – если значение $C_j(\bar{X}_j)$ больше либо равно нулю.

Предварительно задаются начальные условия, симуляции. Затем следует главный цикл программы. На очередном шаге явным методом Эйлера вычисляются предполагаемые (пробные) координаты $\bar{p}_i$, куда точки переместились бы без учёта реакции внутренних сил. Процедура GenerateCollisions находит коллизии ткани с самой собой и другими объектами сцены, создавая для каждой из них дополнительное ограничение. В теле процедуры EvaluateConstraints последовательно обходятся функции $C_j(\bar{P}_j)$ и вершины $\bar{P}_j$ перемещаются, так, чтобы очередное ограничение было удовлетворено. Однако, уточняя координаты некоторых вершин $\bar{P}_a$ так, чтобы выполнялось одно условие, можно нарушить другое. Из-за этого приходится совершать несколько уточняющих итераций. Практика показывает, что с каждым повторением модель приближается к состоянию равновесия. Когда выполнено заданное количество итераций, состояние системы обновляется. Наконец, учитывается трение, которое возникло при коллизиях.

Оригинальный метод PBD имеет существенные недостатки. Так, результат моделирования зависит от временного шага и количества уточняющих итераций. Другими словами, в этом отношении он походит на метод масс и пружин.

2.3. Метод XPBD

На рисунке 2 представлен псевдокод метода XPBD [5].

```
(1)  for all vertices i
(2)     initialize x_i = x_i^0, v_i = v_i^0, w_i = 1 / m_i
(3)  end for
(4)  loop
(5)     for all vertices i
(6)        v_i = v_i + Δt·w_i·f_ext(x_i)
(7)        p_i = x_i + Δt·v_i
(8)     end for
(9)     GenerateCollisions(x_i, p_i)
(10)    for all constraints i
(11)       λ_i = 0
(12)    end for
(13)    for n_iter
(14)       for all constraints j
(15)          calculate Δλ_i
(16)          calculate ΔP_j
(17)          λ_j = λ_j + Δλ_j
(18)          P_j = P_j + ΔP_j
(19)       end for
(20)    end for
(21)    for all vertices i
(22)       v_i = (p_i − x_i) / Δt
(23)       x_i = p_i
(24)    end for
(25)    EvaluateFriction(v_i)
(26) end loop
```

Рисунок 2 – Алгоритм метода XPBD

Запишем закон движения системы вершин $\bar{X} = [\bar{x}_1{}^T, \bar{x}_2{}^T, \dots, \bar{x}_n{}^T]^T$ в потенциальном поле

$$M \cdot \ddot{\bar{X}} = -\nabla_{\bar{X}} U^T(\bar{X}), \quad (1)$$

где
M – блочная диагональная матрица масс материальных точек,
$U(\bar{X})$ – потенциальная энергия системы.
Потенциальную функцию $U(\bar{X})$ можно представить в виде

$$U(\bar{X}) = \frac{1}{2} \cdot \bar{C}^T(\bar{X}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}), \tag{2}$$

где α – блочная матрица, на диагонали которой расположены обратные жесткости связей.
После подстановки (2) в (1)

$$M \cdot \ddot{\bar{X}} = -\nabla_{\bar{X}} \bar{C}^T(\bar{X}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}). \tag{3}$$

Запишем разностный аналог уравнения (3)

$$M \cdot \left(\frac{\bar{X}^{n-1} - 2 \cdot \bar{X}^n + \bar{X}^{n+1}}{\Delta t^2}\right) = -\nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \alpha^{-1} \cdot \bar{C}(\bar{X}^{n+1}). \tag{4}$$

Обозначим

$$\bar{\lambda}^{n+1} = -\tilde{\alpha}^{-1} \cdot \bar{\mathrm{C}}(\bar{X}^{n+1}),$$

где $\tilde{\alpha} = \frac{\alpha}{\Delta t^2}$ – жесткость ограничения с поправкой на частоту обновления.
Тогда (4) можно записать в виде

$$M \cdot (\bar{X}^{n-1} - 2 \cdot \bar{X}^n + \bar{X}^{n+1}) = -\nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \bar{\lambda}^{n+1}. \tag{5}$$

Введем также вектор $\bar{P}^n = 2 \cdot \bar{X}^n - \bar{X}^{n-1} = \bar{X}^n + \Delta t \cdot \bar{V}^n$ – предполагаемое положение точек на следующем шаге. Теперь (5) представляется в форме нелинейной системы

$$M \cdot (\bar{X}^{n+1} - \bar{P}^n) - \nabla_{\bar{X}^{n+1}} \bar{C}^T(\bar{X}^{n+1}) \cdot \bar{\lambda}^{n+1} = \bar{0}, \tag{6}$$

$$\bar{\mathrm{C}}(\bar{X}^{n+1}) + \tilde{\alpha} \cdot \bar{\lambda}^{n+1} = \bar{0}, \tag{7}$$

которую решают решать методом Ньютона. Обозначим уравнения (6) и (7) как $\bar{g}(\bar{X}, \bar{\lambda}) = \bar{0}$ и $\bar{h}(\bar{X}, \bar{\lambda}) = \bar{0}$ соответственно. Ниже будем использовать обозначение i вместо n, так как речь будет идти об итерационном процессе. После линеаризации получим следующие уравнения

$$\begin{bmatrix} K & -\nabla_{\bar{X}} \bar{C}^T(\bar{X}^i) \\ \nabla_{\bar{X}} \bar{\mathrm{C}}(\bar{X}^i) & \tilde{\alpha} \end{bmatrix} \cdot \begin{bmatrix} \Delta\bar{X} \\ \Delta\bar{\lambda} \end{bmatrix} = -\begin{bmatrix} \bar{g}(\bar{X}^i, \bar{\lambda}^i) \\ \bar{h}(\bar{X}^i, \bar{\lambda}^i) \end{bmatrix}, \tag{8}$$

где $K = \frac{\partial \bar{g}}{\partial \bar{X}}$ – частная производная (7) по $\bar{X}$.

На каждой итерации можно решать эту СЛАУ для $\Delta\bar{X}, \Delta\bar{\lambda}$ и обновлять координаты вершин, также коэффициенты по формулам

$$\bar{X}^{i+1} = \bar{X}^i + \Delta\bar{X},$$

$$\bar{\lambda}^{i+1} = \bar{\lambda}^i + \Delta\bar{\lambda}.$$

Однако, вычисление матрицы системы (8) будет занимать довольно много процессорного времени. Примем следующие допущения:

1. $K \approx M$, что влечет за собой локальную ошибку порядка $O(\Delta t^2)$ при решении задачи
2. $\bar{g}(\bar{X}^i, \bar{\lambda}^i) = \bar{0}$. Это предположение выполняется на первой итерации. Если же градиент вектора ограничений изменяется медленно, $\bar{g}(\bar{X}^i, \bar{\lambda}^i)$ и дальше будет оставаться малой

С учетом вышеизложенного, система принимает вид

$$\begin{bmatrix} \mathrm{M} & -\nabla_{\bar{X}} \bar{C}^T(\bar{X}^i) \\ \nabla_{\bar{X}} \bar{\mathrm{C}}(\bar{X}^i) & \tilde{\alpha} \end{bmatrix} \cdot \begin{bmatrix} \Delta\bar{X} \\ \Delta\bar{\lambda} \end{bmatrix} = -\begin{bmatrix} \bar{0} \\ \bar{h}(\bar{X}^i, \bar{\lambda}^i) \end{bmatrix}.$$

Используя дополнение Шура, можно получить выражение для $\Delta\bar{\lambda}$

$$[\nabla \bar{C}_{\bar{X}}(\bar{X}^i) \cdot M^{-1} \cdot \nabla \bar{C}_{\bar{X}}(\bar{X}^i)^T + \tilde{\alpha}] \cdot \Delta\bar{\lambda} = -\bar{C}(\bar{X}^i) - \tilde{\alpha} \cdot \bar{\lambda}^i,$$

Откуда

$$\Delta\bar{\lambda} = \frac{-\bar{C}(\bar{X}^i) - \tilde{\alpha} \cdot \bar{\lambda}^i}{\nabla \bar{C}_{\bar{X}}(\bar{X}^i) \cdot M^{-1} \cdot \nabla \bar{C}_{\bar{X}}(\bar{X}^i)^T + \tilde{\alpha}}. \tag{9}$$

Тогда коррекция $\Delta\bar{X}$

$$\Delta\bar{X} = M^{-1} \cdot \nabla_{\bar{X}}(\bar{X}^i)^T \cdot \Delta\bar{\lambda}. \tag{10}$$

Запишем (9), (10) применительно к отдельной функции $C_j(\bar{X}_j)$

$$\Delta\lambda_j = \frac{-C_j(\bar{X}^n{}_j) - \tilde{\alpha}_j \cdot \lambda_j}{\nabla C_{\bar{X}_j}(\bar{X}^n{}_j) \cdot M_j{}^{-1} \cdot \nabla C_{\bar{X}_j}(\bar{X}^n{}_j)^T + \tilde{\alpha}_j}, \tag{11}$$

$$\Delta\bar{x}_{ji} = w_{ji} \cdot \nabla_{\bar{x}_{ji}}(\bar{X}^n{}_j)^T \cdot \Delta\lambda_j.$$

Дополнительное слагаемое $\tilde{\alpha}_j$ в знаменателе (11) играет роль ограничителя перемещения, которое $C_j(\bar{P}_j)$ прикладывает к точкам. Итоговое значение λ_j характеризует суммарное количество силы, приложенное связью к своим вершинам после n_{iter} итераций. Таким образом, удается преодолеть недостатки оригинального метода. Заметим, что при $\alpha_j = 0$, то есть в случае бесконечной жёсткости, метод XPBD вырождается в оригинальный PBD.

2.4. Некоторые ограничения, применяемые для симуляции ткани

2.4.1. Реалистичное ограничение на растяжение

Это ограничение основано на механике сплошных сред [6]. Оно действует на треугольник ткани, стремясь сохранить его форму. Для корректной работы данной связи этапе инициализации симуляции требуется задать тензор упругости материала C. Также необходимо вычислить площадь $A_{\text{нач}}$ и матрицу формы $D_{\text{нач}}$ каждого конечного элемента в состоянии покоя

$$D_{\text{нач}} = \begin{pmatrix} x_1 - x_3 & x_2 - x_3 \\ y_1 - y_3 & y_2 - y_3 \end{pmatrix},$$

где x_i, y_i – координаты вершин на плоскости треугольника.

При применении ограничения нужно сначала найти градиент деформации F конечного элемента

$$F = D_{\text{текущ}} \cdot D_{\text{нач}}{}^{-1},$$

а затем вычислить тензор деформации Коши-Грина ε. Заметим, что ε является симметричной матрицей второго ранга, однако этот тензор можно представить в виде столбца из трех элементов. Таким образом, находят тензор напряжений S

$$S = C \cdot \varepsilon.$$

К нему применяется правило, описанное выше, но в обратную сторону. Теперь необходимо вычислить плотность энергии напряжений

$$\psi_s = \frac{1}{2} \cdot tr(\varepsilon^T \cdot S).$$

Наконец, найдём энергию напряжений в треугольнике, которая, равна потенциальной энергии, накопленной в конечном элементе вследствие деформации

$$C(P_1, P_2, P_3) = A_{\text{нач}} \cdot \psi_s.$$

Частные производные этой функции равны

$$[\nabla_{\bar{P}_1} C(\bar{P}_1, \bar{P}_2, \bar{P}_3), \nabla_{\bar{P}_2} C(\bar{P}_1, \bar{P}_2, \bar{P}_3)] = A_{\text{нач}} \cdot P \cdot D_{\text{тек}}{}^{-T},$$

$$\nabla_{\bar{P}_3} C(\bar{P}_1, \bar{P}_2, \bar{P}_3) = -\sum_{i=1}^{2} \nabla_{\bar{P}_i} C(\bar{P}_1, \bar{P}_2, \bar{P}_3),$$

где $P = F \cdot S$ – тензор напряжений Пиолы-Кирхгоффа.

2.4.2. Реалистичное ограничение на изгиб

Это ограничение основано на методе конечных элементов. Данная связь действует на два смежных треугольника ткани. Пронумеруем вершины и ребра треугольников, как показано на рисунке 3.

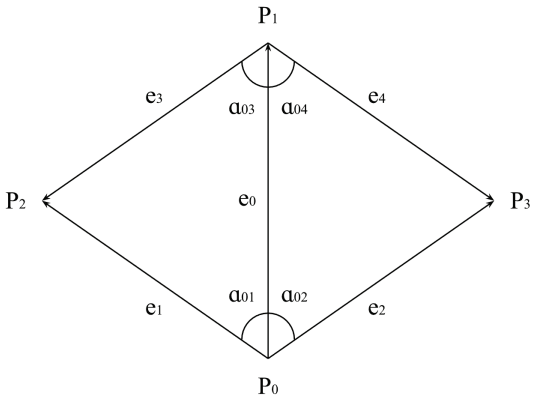


Рисунок 3 – Реалистичное ограничение на изгиб

На этапе инициализации для каждой такой пары необходимо вычислить матрицу Гесса

$$Q = \frac{3}{A_1 + A_2} \cdot \overline{K}^T \cdot \overline{K},$$

где
A_1, A_2 – площади треугольников,
$\overline{K} = [c_{03} + c_{04}, c_{01} + c_{02}, -c_{01} - c_{03}, -c_{02} - c_{04}]$ – вектор, составленный из $c_{ij} = ctg(\alpha_{ij})$.
Тогда функция, соответствующая этому ограничению, имеет вид

$$C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = \frac{1}{2} \cdot \sum_{i,j=0}^{4} Q_{i,j} \cdot \bar{P}_i \cdot \bar{P}_j. \quad (12)$$

Частные производные (12) равны

$$\nabla_{\bar{P}_i} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = \sum_{j=1}^{4} Q_{i,j} \cdot \bar{P}_j.$$

2.4.3. Коллизия ткани типа вершина-треугольник

Это ограничение, созданное коллизией [7]. Как оно возникает, будет описано ниже. Функция C в этом случае действует на четыре вершины ткани, из которых $\bar{P}_1, \bar{P}_2, \bar{P}_3$ принадлежат одному треугольнику, и имеет вид

$$\mathrm{C}(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = \bar{n} \cdot (\bar{P}_0 - \bar{P}_1) - 2 \cdot r, \quad (13)$$

где
$\bar{n}$ – единичная нормаль к $[\bar{P}_1, \bar{P}_2, \bar{P}_3]$, взятая со знаком плюс, если $\bar{P}_0$ движется к нему сверху, и со знаком минус в противном случае,
r – толщина материала.

Производные функции (13) равны

$$\nabla_{\bar{P}_0} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = \bar{n},$$

$$\nabla_{\bar{P}_1} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_2 - \bar{P}_3) \times N \cdot \bar{n} - \bar{n}$$

$$\nabla_{\bar{P}_2} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_3 - \bar{P}_1) \times N \cdot \bar{n} - \bar{n},$$

$$\nabla_{\bar{P}_3} C(\bar{P}_0, \bar{P}_1, \bar{P}_2, \bar{P}_3) = (\bar{P}_2 - \bar{P}_1) \times N \cdot \bar{n} - \bar{n},$$

где
$N = \frac{(E - \bar{n}^T \cdot \bar{n})}{d}$ – матрица геометрической жесткости,
$d = |(\bar{P}_2 - \bar{P}_1) \times (\bar{P}_3 - \bar{P}_1)|$.

Эта связь должна иметь бесконечную жесткость, и, следовательно, вычисляется методом PBD, а не XPBD. Заметим, что коллизия типа вершина-треугольник с твердым телом обрабатывается аналогичным образом. Единственное отличие заключается в том, что масса вершин, принадлежащих твердому телу, считается бесконечной.

2.4.4. Коллизия типа ребро-ребро

Как и в прошлом пункте, исходим из того, что ограничение уже создано. Пусть обнаружена коллизия между ребрами ткани $[\bar{P}_1, \bar{P}_2]$ и, $[\bar{P}_3, \bar{P}_4]$ и ближайшие точки этих двух отрезков имеют параметры α и β соответственно. Тогда связь задается функцией

$$C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = (\bar{P}_\alpha - \bar{P}_\beta) \cdot \bar{n} - 2 \cdot r, \tag{14}$$

где

$\bar{n}$ – единичный вектор между $\bar{P}_\alpha$ и $\bar{P}_\beta$ на предыдущем кадре,

r – толщина материала.

Производные функции (14) равны

$$\nabla_{\bar{P}_1} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -(1-\alpha) \cdot \bar{n},$$

$$\nabla_{\bar{P}_2} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -\alpha \cdot \bar{n},$$

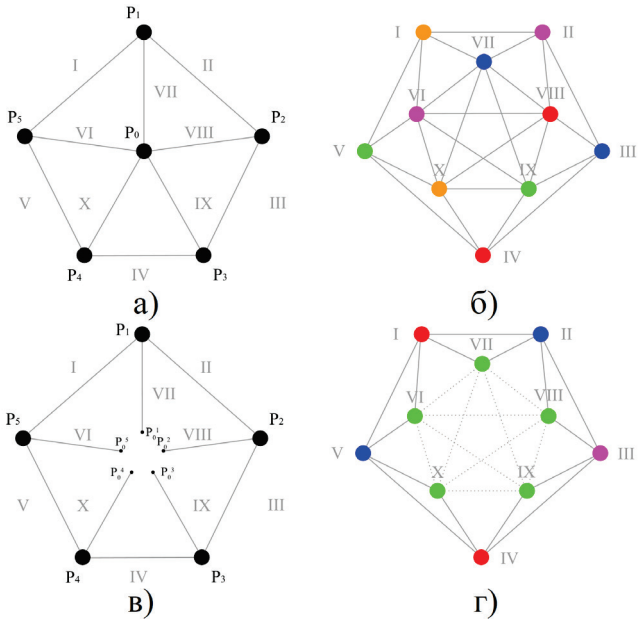
$$\nabla_{\bar{P}_3} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -(1-\beta) \cdot \bar{n},$$

$$\nabla_{\bar{P}_4} C(\bar{P}_1, \bar{P}_2, \bar{P}_3, \bar{P}_4) = -\beta \cdot \bar{n}.$$

Для этого ограничения справедливы те же замечания, что и для прошлого.

2.5. Граф ограничений и фантомные вершины

При параллельном вычислении ограничений могут появиться ошибки. Если $C_a(\bar{P}_a)$ и $C_b(\bar{P}_b)$ влияют одну и ту же точку, возникает гонка потоков. Следовательно, необходимо распределить $C_j(\bar{P}_j)$ таким образом, чтобы связи из любой пары потоков не имели общих вершин. Представим $C_j(\bar{P}_j)$ в виде узлов графа $A_j, j = \overline{1,k}$. Будем строить ребро B_{ab}, если $\bar{P}_a \cap \bar{P}_b \neq \emptyset$ [8]. Тогда проблема сводится к задаче о раскраске $G = [A, B]$. Воспользуемся многопоточным алгоритмом. Таким образом, можно разделить ограничения на «порции» $C_j = [C_{j1}(\bar{P}_{j1}), C_{j2}(\bar{P}_{j2}), \dots, C_{jn_j}(\bar{P}_{jn_j})], j = \overline{1, n_{\text{цвет}}}, n_j \leq k$ и применять связи, принадлежащие каждой из них, параллельно. Пример графа для ткани, состоящей из шести точек, на которую действуют ограничения на растяжение, показан на рисунке 4.



а – ткань, б – оригинальный граф ограничений,

в – ткань с фантомной вершиной, г – упрощённый граф ограничений

Рисунок 4 – Граф ограничений

Однако, хотелось бы, чтобы количество «порций» было минимальным. Заменим точку $\bar{P}_0$ пятью «виртуальными» вершинами $\bar{P}_0^i, i = \overline{1,5}$ и заместим в ограничениях упоминание оригинала на соответствующую фантомную точку $\bar{P}_0^i$. Тогда число $n_{\text{цвет}}$ уменьшится. Доказывается, что если нужно раскрасить граф ограничений в $n_{\text{цвет}}$ цветов, достаточно найти в нем все клики $K_j = [A_{j1}, A_{j1}, \dots A_{jn_j}], j = \overline{1, n}_{\text{клик}}, n_j \le k$, такие, что $n_j \ge n_{\text{цвет}}$, а затем для каждой клики K_j заменить общую точку ткани на фантомную. Поиск клик выполняется алгоритмом Брона-Кербоша. Важно после каждой итерации усреднять координаты $\bar{P}_j^i$, для чего создаются дополнительные, фантомные, ограничения. Использование таких вершин вносит погрешность в симуляцию, однако её можно свести к минимуму, сократив временной шаг или увеличив количество итераций.

2.6. Система поиска коллизий ткани

2.6.1. Поиск пар-кандидатов

Работа системы поиска коллизий начинается после того, как вычислены пробные координаты точек. На первом этапе составляется массив пар-кандидатов, то есть пар примитивов, которые либо уже находятся слишком близко друг к другу, либо могут сблизится на следующем шаге. Для пространственного поиска используется LBVH дерево, которое стротся с помощью многопоточного алгоритма. Имеет смысл хранить в дереве Axis-Aligned Bounding Box для каждого из треугольников, который бы захватывал и текущее, и предсказанное его положение, причём с отступом, равным длине максимального перемещения среди вершин ткани.

Перейдём ко второму этапу. Каждая, найденная на первом этапе, пара треугольников порождает шесть пар-кандидатов типа вершина-треугольник и девять пар-кандидатов типа ребро-ребро. Необходимо выделить из этого множества те пары, между которыми действительно произойдёт коллизия. Покажем, как выполняется эта проверка.

2.6.2. Проверка пары-кандидата типа вершина-треугольник

Пусть известно, что вершина $\bar{X}_0$ находится слишком близко к треугольнику $[\bar{X}_1, \bar{X}_2, \bar{X}_3]$. Найдём точку $\bar{X}^*$. на треугольнике, ближайшую к $\bar{X}_0$, и её естественные координаты $[a, b, c]$. Затем, вычислим пробные координаты $\bar{X}^*$ с помощью линейной интерполяции

$$\bar{P}^* = a \cdot \bar{P}_1 + b \cdot \bar{P}_2 + c \cdot \bar{P}_3.$$

Наконец, построим единичный вектор

$$\bar{n} = \frac{\bar{X}^* - \bar{X}_0}{|\bar{X}^* - \bar{X}_0|}$$

и спроецируем на него перемещения точек $\bar{X}_0$ и $\bar{X}^*$. Вычислим кратчайшее расстояние d между проекциями и выполним проверку

$$d < 2 \cdot r + r_{\text{крит}}, \tag{15}$$

где

r – толщина ткани,

$r_{\text{крит}}$ – критическое расстояние для создания ограничения,

на основании который принимается решение о создании ограничения.

2.6.3. Проверка пары-кандидата типа ребро-ребро

Пусть известно, что ребра $[\bar{X}_1, \bar{X}_2]$ и $[\bar{X}_3, \bar{X}_4]$ находятся слишком близко. Сначала найдём ближайшие точки на этих отрезках. Предположим, что ближайшие точки этих двух отрезков имеют параметры α и β соответственно. Далее предскажем координаты этих точек с помощью линейной интерполяции

$$\bar{P}_\alpha = (1 - \alpha) \cdot \bar{P}_1 + \alpha \cdot \bar{P}_2,$$

$$\bar{P}_\beta = (1-\beta)\cdot\bar{P}_3 + \beta\cdot\bar{P}_4.$$

Наконец, построим единичный вектор

$$\bar{n} = \frac{\bar{P}_\alpha - \bar{P}_\beta}{|\bar{P}_\alpha - \bar{P}_\beta|}$$

и спроецируем на него перемещения вершин $\bar{X}_\alpha, \bar{X}_\beta$. Найдём расстояние d между проекциями.

Как и прошлый раз, решение о создании ограничения принимается на основе условия (15).

2.6.4. Метод Representative triangles для проверки пары-кандидата

Пусть на втором этапе удалена большая часть пар-кандидатов. Однако, среди оставшихся могут быть дубликаты, в случае, если вершина или ребро, участвующее в коллизии, принадлежит нескольким треугольникам. Чтобы не допустить этого, на этапе инициализации необходимо пройти по всем конечным элементам ткани и твердых тел, указав, какими из примитивов владеет очередной треугольник так, чтобы каждый примитив принадлежал только одному из них. На втором этапе необходимо создавать пары только из тех примитивов, которые принадлежат треугольникам.

2.7. Трение ткани

Для симуляции трения была выбрана модель Амонтона-Кулона. После очередного шага для каждой коллизии типа вершина-треугольник необходимо корректировать тангенциальную скорость вершины относительно треугольника $\bar{V}_т$ по формуле

$$\bar{V}'_т = max\left(1-\mu\cdot\frac{\Delta\bar{V}_н}{|\bar{V}_т|}, 0\right)\cdot\bar{V}_т.$$

где

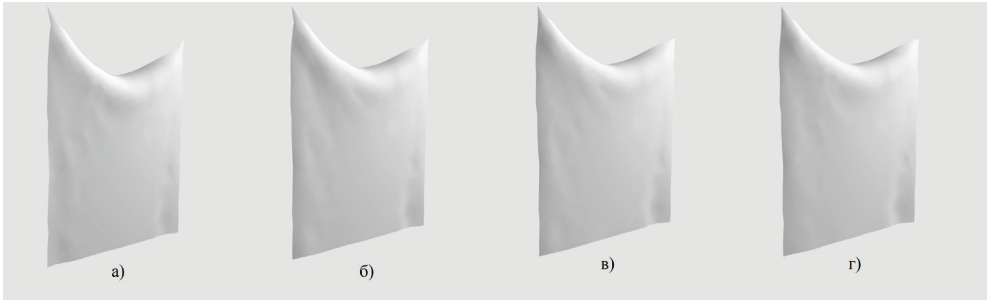
$\bar{V}_н$ – нормальная составляющая скорости вершины относительно треугольника,

μ – коэффициент трения.

3. Практическая часть

3.1. Зависимость симуляции от параметров

Рассмотрим простейший пример – квадратный фрагмент ткани, состоящий из пятисот пятидесяти вершин, в котором действуют реалистичные ограничения. Покажем, как влияет на симуляцию варьирование временного шага и количества итераций при неизменных значениях остальных параметров. На рисунке 5 приведены результаты симуляции при различном количестве итераций для шага по времени, равного одной сотой секунды.

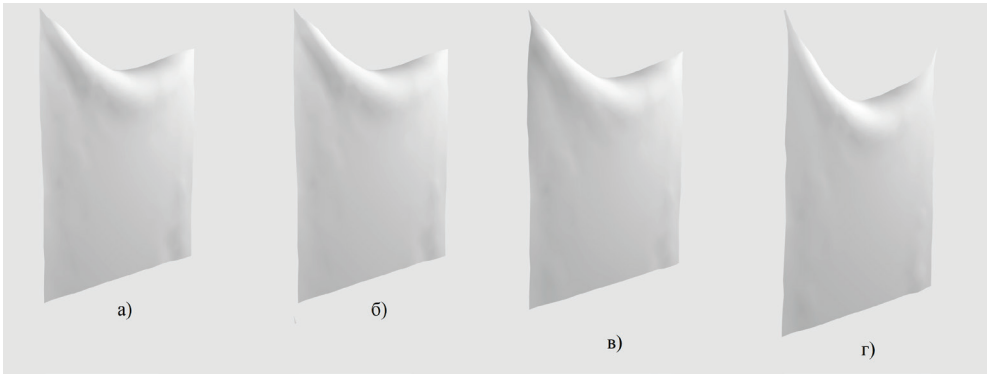


а – двадцать итераций, б – сорок итераций,

в – восемьдесят итераций, г – сто шестьдесят итераций

Рисунок 5 – Зависимость симуляции от числа итераций

Можно видеть, что количество итераций практически не влияет на итог моделирования. На рисунке 6 приведены результаты симуляции для разного шага по времени при сорока итерациях. В последнем случае видны заметные изменения, но в целом симуляция стабильна.



а – одна тысячная секунды, б – одна пятисотая секунды,

в – одна сотая секунды, г – одна пятидесятая секунды

Рисунок 6 – Зависимость симуляции от временного шага

3.2. Анализ производительности программы

Рассмотрим следующий пример: фрагмент ткани, состоящий из тысячи двухсот двух вершин примеряется на манекен из такого же количества точек**.** «Одежда» представляет собой ту же модель, что и твердое тело, но растянута вдоль осей с помощью коэффициента, равного одной целой и двум десятым. В начальный момент времени ткань расположена чуть выше манекена. Измерим, сколько миллисекунд требуется для вычисления двухсот кадров при временном шаге, равном одной сотой секунды, и двадцати итерациях.

Пример представлен на рисунке 7.

Рисунок 7 – Тестовый пример

Тест проводится для одного, двух, четырех и восьми потоков. Фантомные вершины не использовались. Изменения проводились на CPU Intel Core i7-6700K.

Результаты тестов приведены в таблице 1.

Таблица 1 – Производительность программы

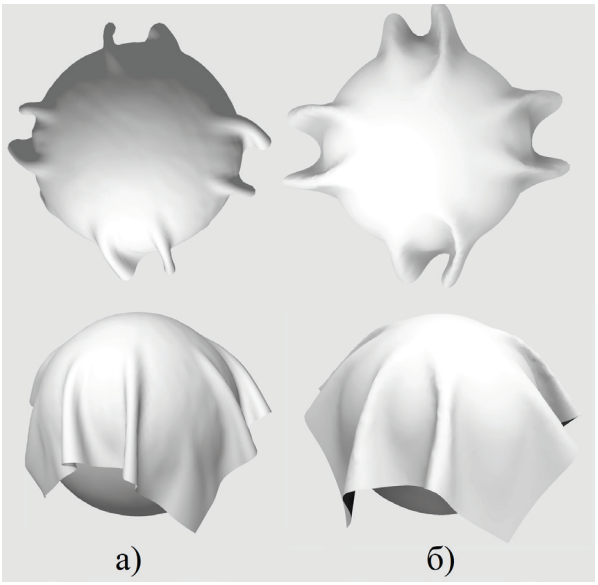
Задача	Количество потоков				
	1	2	4	6	8
Создание LBVH дерева, мс.	6	12	14	34	43
Поиск коллизий-кандидатов, мс.	1699	909	622	492	953
Проверка коллизий-кандидатов, мс.	283	139	20	9	11
Построение и раскраска графа коллизий, мс.	111	969	1152	1359	1479
Применение ограничений и трения, мс.	2601	2200	1700	1590	1609
Другие задачи, мс.	696	677	743	775	786
Итого, мс.	5396	4906	4251	4259	4881

Можно видеть, что с увеличением числа потоков с одного до шести уменьшается время, затрачиваемое на выполнение всех задач кроме построения и раскраски графа коллизий. За счет этого удается ускорить программу на двадцать процентов.

Четвертая строка таблицы включает построение графа коллизий, которое выполняется в однопоточном режиме, и раскраску графа коллизий с помощью параллельного алгоритма. Если программа работает последовательно, то раскраска графа не требуется, поэтому значение в первом столбце так сильно отличается от остальных. Кроме того, рост числа потоков негативно влияет на скорость построения графа. Причину этого эффекта установить не удалось.

3.3. Сравнение с Marvelous Designer

Сравним наше приложение с известной средой Marvelous Designer. В обоих случаях создадим сферу с радиусом, равным тридцати пяти сантиметрам, состоящую из двух тысяч пятисот вершин, и поместим над ней на высоте пяти сантиметров квадратный фрагмент ткани со стороной, равной одному метру и состоящий из двух тысяч семисот вершин. Постараемся задать в программах похожие свойства ткани несмотря на то, что в одном случае материал описывается тензором эластичности и жёсткостью ограничения на изгиб, а в другом – параметрами KES. Результаты симуляции представлены на рисунке 8.



а – наше приложение, б - Marvelous Designer

Рисунок 8 – Сравнение с Marvelous Designer

Качественно результаты схожи. Так, при виде сверху в обоих случаях прослеживается по восемь симметричных складок. При взгляде сбоку подобие результатов также не вызывает сомнений.

В таблице 2 приведены средние значения количества кадров в секунду для первых двухсот кадров симуляции при временном шаге, равном одной сотой секунды. Наше приложение совершает тридцать итераций, а в Marvelous Designer установлен режим Accurate Fabric. Можно видеть, что разрыв в скорости не превышает двадцати пяти процентов.

Таблица 2 – Сравнение с Marvelous Designer

Среднее количество кадров в секунду, шт.	Количество потоков				
	1	2	4	6	8
Наше приложение	13	20	21	21	21
Marvelous Designer	13	20	27	26	26

3.4. Пути дальнейшего улучшения программы

Можно сделать вывод, что хотя увеличение количества ядер повышает скорость программы, но выигрыш достаточно мал. Чтобы исправить это, необходимо предпринять следующие шаги: реализовать многопоточное создание графа ограничений, провести дальнейшую оптимизацию алгоритмов и, наконец, перенести параллельные вычисления на GPU.

4. Благодарности

Хочу выразить благодарность всему преподавательскому составу кафедры ФН-11 «Математика и компьютерные науки» МГТУ им. Баумана за их профессионализм и знания, передаваемые ими в течение четырех лет обучения, а также всем сотрудникам ООО «Траектория» за поддержку.

5. Список использованных источников

[1] Dynamic Cloth Simulation: A Comparative Study of Explicit and Implicit Numerical Integration / L. L. D. Carvalho, C. A. Vidal, J. B. Cavalcante-Neto, S. M. F. d. Oliveira // 14th Symposium on Virtual and Augmented Reality (Рио-де-Жанейро, 5 мес. 2012 г.) / IEEE. Лос Аламитос: IEEE, 2012. С. 56-65. DOI: 10.1109/SVR.2012.11.

[2] O. Etzmuss, M. Keckeisen, W. Strasser A fast finite element solution for cloth modelling // 11th Pacific Conference on Computer Graphics and Applications (Канмор, 10 мес. 2003 г.) / IEEE. Лос Аламитос: IEEE, 2003. С. 244-251. DOI: 10.1109/PCCGA.2003.1238266.

[3] D. Baraff, A. Witkin Large Steps in Cloth Simulation // SIGGRAPH ’98 (Орландо, 7 мес, 1998 г.) / SIGGRAPH. Нью-Йорк: Association for Computing Machinery, 1998. С. 43-54. DOI: 10.1145/280814.280821.

[4] Position Based Dynamics / M. Matthias, B. Heidelberger, M. Hennix, J. Ratcliff // VRIPHYS 2006 (Мадрид, 11 мес. 2006 г.) / VRIPHYS. США: Academic Press, Inc., 2007. С. 109-118. DOI: 10.1016/j.jvcir.2007.01.005.

[5] M. Miles, M. Matthias, C. Nuttapong XPBD Position-Based Simulation of Compliant Constrained Dynamics // MIG’16. (Берлингейм, 10 мес. 2016 г.) / SIGGRAPH. Нью-Йорк: Association for Computing Machinery, 2016, С. 49-54. DOI: 10.1145/2994258.2994272.

[6] Position-Based Simulation of Continuous Materials / J. Bender, D. Koschier, P. Charrier, D. Weber // Computers and Graphics. 2014. №44. С. 1-10. DOI: 10.1016/j.cag.2014.07.004.

[7] C. Lewin. Cloth Self Collision with Predictive Contacts // GDC 2018. (Сан-Франциско, 3 мес. 2018 г.) / GDC. 2018.

[8] M. Fratarcangeli, F. Pellacini Scalable Partitioning for Parallel Position Based Dynamics // Computer Graphics Forum. 2015. № 34. С. 405-413. DOI: 10.1111/cgf.12570.