**K.S. Nikolaev**
Moscow Institute of Electronic Technology

## APPLICATION OF MODERN PARALLEL TECHNOLOGIES TO THE SOLUTION OF THE PROBLEM OF MULTIPLAYING MULTIDIMENSIONAL MATRICES BY THE METHOD OF RECURSIVE DESCENT

**Keywords:** *multidimensional matrices; recursive descent; parallel programming; CUDA; Nvidia; OpenCL; cluster computing.*

**Abstract.** *The article considers the problem of multiplication of multidimensional arrays, interpreted as multidimensional matrices according to N.P. Sokolov. Existing methods and approaches to solving this problem are analyzed. Methods of accelerating the solution of a problem using various technologies of parallel programming are investigated.*

**А.И. Павлюков**
*Смоленский государственный университет*

УДК 004.85

## ШЕЙДЕРНАЯ ГЕОМЕТРИЯ, ПРОЦЕСС ПОЛУЧЕНИЯ КОЛЕБЛЮЩЕЙСЯ ТРАВЫ

Ключевые слова: *шейдерная геометри;, шейдеры; реальное время; анимация; графы; рендеринг в реальном времени; c#,hlsl; unity.*

*Предложена программная реализация задачи крупномасштабного рендеринга травы в реальном времени. Более реалистичный и естественный эффект – когда каждый стебель качается с ветром. Рассмотрен относительно новый, недавно введенный геометрический шейдер для реализации логики создания отдельных стеблей на графическом процессоре. Программная часть выполнена средствами языков c# и hlsl. Предложенный геометрический шейдер может генерировать огромные пласты травы в реальном времени, требует небольшого объёма памяти и обеспечивает высокое качество каждого объекта и реалистичное движение под действием волн ветра.*

**Введение.** Геометрические шейдеры могут дать больше гибкости, когда речь идет об оптимизации и настройке. Unity3D удобно сочетает вершинные, геометрические и фрагментные шейдеры, что открывает новые возможности в создании детализированных объектов.

Детальное моделирование отдельных травинок не имеет смысла, потому что количество полигонов, которое потребовалось бы для более крупных лугов, было бы слишком большим. Сцена с бесчисленными стеблями полигональной травы не будет отображаться в режиме реального времени с графическим оборудованием, доступным сегодня. Для решения этой проблемы использовалась недавно введенная шейдерная геометрия [1], которая позволяет реализовывать логику создания отдельных курсивных стеблей на графическом процессоре.

Схема рендеринга больших участков травы удовлетворяет следующим условиям:

- многоугольников для одной травы не может быть слишком много;
- с разных точек зрения трава должна казаться густой;
- расположение травы не должно быть слишком регулярным, иначе это будет неестественно.

**Создание и настройка шейдера.** Каждый стебель имеет 3 режима качества изображённых на рисунке 1; каждый из них занимает один, три, пять квадратов.

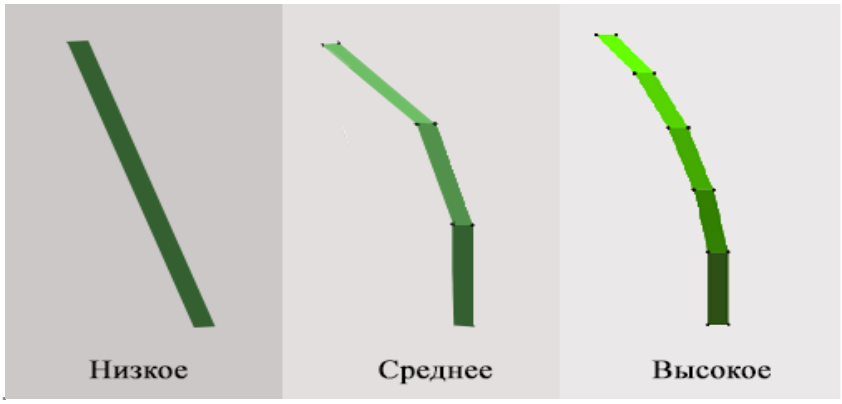


Рис. 1. Три режима качества

Поскольку вход в геометрический шейдер является примитивом (точки, треугольники, …), а не вершиной, нам нужно создать примитив типа точки в CPU на основе случайного местоположения в качестве корневого местоположения этой травы [2]. Затем то самое на GPU через корневую позицию, чтобы сделать стебли травы. Высота и ширина самой травы также задаётся случайно.

После задание свойств создаётся новая вершина, имитирующую внешний вид листа [3]. Для формирования травы требуется 12 различных вершин, но, поскольку здесь не используется индекс, в итоге получается 30 вершин, чтобы составить 5 квадратов (рис. 2.

Основываясь на этом простом графе, вычисляем положение каждой вершины в зависимости от положения корня.

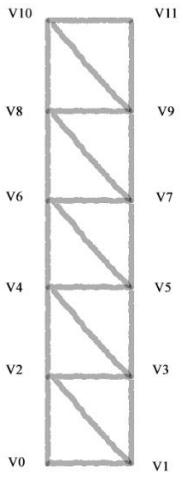


Рис. 2. Граф стебля

Также находим, что четные вершины соответствуют uv-координатам (0, v), а нечетные вершины – UV-координатам (1, v); здесь v находится в UV-координатах, поэтому вычисляем uv-координаты каждой вершины [4]. Таким образом, сетка стебля создается на графическом процессоре.

Для отображения травы обрабатываем её контур, вместо привычного использования альфа-смеси здесь используем альфа-покрытие, потому что при работе с сильно уложенными листьями травы у смеси есть некоторые проблемы с порядком отображения. Что касается использования альфа-покрытия, оно улучшает внешний вид краёв и точность наложения, в зависимости от используемого уровня MSAA.

Цвет задаётся по Полу-Ламберту, то есть получаем скалярное произведение вектора направленного света и вектора нормали мирового пространства, которое затем умножаем на 0,5 и добавляем к нему 0,5 [5]. Это делается, чтобы не освещенная напрямую трава была не полностью темной, а имела более плавный градиент. В цветовом канале использовались различные оттенки зеленого и желтого, чтобы получить лучшую дифференциацию одиночных стеблей

**Сетка земли.** В природе трава никогда не бывает равномерно распределена по земле. Различные внешние явления вносят хаос: вид грунта, наличие воды, камней, дорог и т.д. Все это влияет на плотность травы. Мы определяем плотность травы как количество травинок на единицу поверхности. Плотность травы в точке земли, которую мы называем локальной плотностью, можно определить с помощью карты плотности, нанесенной на рельеф местности. Поскольку верхняя граница вершин сетки для Unity равна 65 000, то можно предположить, что размер наземной сетки будет 250*250 [6]. Таким образом, создается естественная и реальная сетка земли (рис. 3).

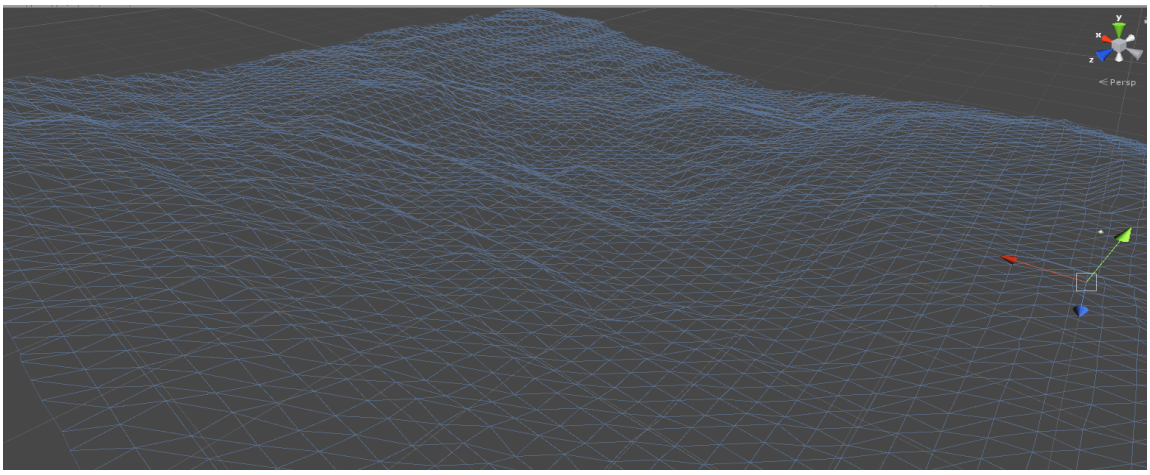

Рис. 3. Сетка земли

После этого появилась трава. Генерируется некоторая вершина, которая передается как корневая позиция стебля до завершения шейдера. Важно отметить, что, поскольку плотность травы должна быть достаточно большой, требуется более одной сетки травы: например, если мы хотим посадить 200 000 травинок, нам нужно 3 сетки. После создания сетки земли вы можете видеть, что расположение корней случайным образом распределяется по земле в количестве миллионов (рис. 4).

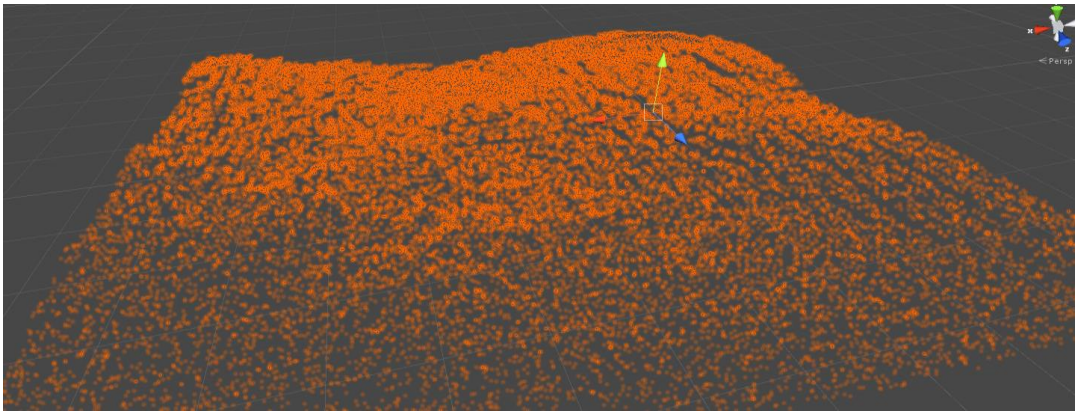

Рис. 4. Корневые позиции

**Моделирование ветра.** Для создания естественности надо, чтобы трава двигалась, имитировать эффект ветра. Идея состоит в том, чтобы использовать тригонометрическую функцию, заставить лист травы качаться. Для этого обеспечим начальную фазу тригонометрической функции в соответствии с положением корня травы, а затем добавляем некоторую случайность внутри, чтобы сделать эффект более естественным. Но каждый стебель имеет независимую сетку, чем ближе к верхушке листа, тем сильнее он подвержен ветру. Будем увеличивать влияние ветра на положение вершин: чем выше вершина, тем выше степень воздействия. В вершинном шейдере легко отличить эти

вершины от нижних, изучив координаты текстуры. Все верхние вершины имеют одну и ту же координату v для текстуры травы1 на кончике травинок и 0 у основания, за счёт этого достигается более реалистичный эффект ветра. Результат работы можно увидеть на рисунке 5.

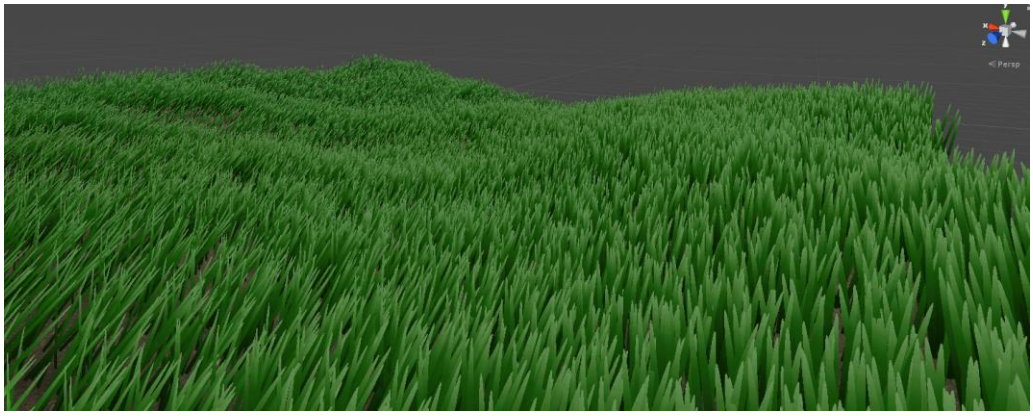

Рис. 5. Колеблющая трава

**Заключение**. Предложена программная реализация процесса получения колеблющейся травы в реальном времени, основанного на геометрическом шейдере. Она имеет ряд преимуществ. А именно, сложные вычисления анимации производятся с помощью алгоритмов на базе процессора, здесь нет никаких искажений, из-за постоянного расстояния верхних вершин многоугольника, изменение положения вершины в шейдере вершин позволяет обеспечить непрерывность пульсирования волн ветра, кластеры неразличимы. Несмотря на свою компактную структуру. Система чрезвычайно быстрая по сравнению со своими предшественниками, требует только миллисекунды времени на расчёт, обеспечивая при этом огромное количество рассеивания точек по площади. Предложенный подход показал свою эффективность и может быть использован для построения крупных природных ландшафтов в реальном времени.

## Литература


1. GrassShader. URL: https://roystan.net/articles/grass-shader.html.
2. Tessellation Subdividing Triangles. URL: https://catlikecoding.com/unity/tutorials/advanced-rendering/tessellation/.
3. My take on shaders: Grass Shader. URL: https://halisavakis.com/my-take-on-shaders-grass-shader-part-ii/
4. [Unity3D] Intro to Geometry Shader. URL: https://jayjingyuliu.wordpress.com/2018/01/24/unity3d-intro-to-geometry-shader/.
5. Shaders in Unity. URL: http://shaderslab.com/shaders.html.
6. Harry A.D.My take on shaders: Geometry Shaders. URL: https://halisavakis.com/my-take-on-shaders-geometry-shaders/.
7. Chapter 7. Rendering Countless Blades of Waving Grass. URL: https://developer.nvidia.com/gpugems/gpugems/part-i-natural-effects/chapter-7-rendering-countless-blades-waving-grass.

8. Geometry Shader Stage. URL: https://docs.microsoft.com/ru-ru/windows/win32/direct3d11/geometry-shader-stage?redirectedfrom=MSDN.
9. Rendering Grass Terrains in Real-Time with Dynamic Lighting. URL: http://kevinboulanger.net/publications/grassSiggraph2006ppt.pdf.

**A.I. Pavlykov**
Smolensk State University

## SHADER GEOMETRY, PROCESS OF RECEIVING OSCILLATING HERBS

**Keywords**: *shader geometry; shaders; real time; animation, graphs; real-time rendering; c #, hlsl; unity.*

**Abstract.** *A software implementation for the task of large-scale grass rendering in real time is proposed. A more realistic and natural effect when each stem sways with the wind. A relatively new, recently introduced geometric shader for implementing the logic of creating individual stems on a GPU is considered. The software part is made using the c # and hlsl languages. The proposed geometric shader can generate huge layers of grass in real time, requires a small amount of memory and provides high quality of each object and realistic movement under the influence of wind waves.*

**Е.И. Парфенова**
*Национальный исследовательский университет «МИЭТ»*

УДК 004.424.22

## О ФУНКЦИИ, ИСПОЛЬЗУЕМОЙ В РАЗРАБОТКЕ РЕДАКТОРА МЕНЮ ТЕРМИНАЛОВ

Ключевые слова: *функция; редактор меню терминалов; цифровая подстанция.*

*В данной статье рассматривается пример функции, написанной для редактора меню терминалов программного комплекса САПР «Сириус». Приводится объяснение, для чего нужна данная среда и что выполняет данная функция. Описаны результаты выполненной работы.*

**Введение.** В статье рассматривается пример функции, написанной для редактора меню терминалов – одного из редакторов узкоспециализированной среды, известной как программный комплекс «Система автоматизированного проектирования Сириус» (САПР