УДК 004.92
DOI 10.25205/1818-7900-2020-18-4-54-65

Некоторые аспекты визуализации трехмерных сферического и эллиптического пространств

Д. И. Мигранов

Новосибирский государственный университет
Новосибирск, Россия

Аннотация

Рассматриваются методы и алгоритмы визуализации геометрических свойств трехмерных сферического и эллиптического пространств с видом от первого лица: предлагается алгоритм реализации управления камерой в таких пространствах, рассматривается подход к визуализации геометрических свойств таких пространств, в основе которого лежат операции над матрицами и векторами, что позволяет перенести необходимые вычисления на графический ускоритель с помощью графических API, описывается методика реализации эффекта дымки в таких пространствах с целью облегчить ориентирование в них. Представлена реализация этих подходов и алгоритмов, позволяющая визуализировать динамические сцены в таких пространствах.

Ключевые слова

неевклидовы пространства, визуализация, компьютерная графика, компьютерное моделирование

Благодарности

Автор выражает глубочайшую признательность д-ру техн. наук, зав. кафедрой компьютерных технологий ФИТ НГУ Владимиру Евгеньевичу Зюбину за советы и ценные замечания.

Для цитирования

Мигранов Д. И. Некоторые аспекты визуализации трехмерных сферического и эллиптического пространств // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 4. С. 54–65. DOI 10.25205/1818-7900-2020-18-4-54-65

Some Aspects of Three-Dimensional Spherical and Elliptical Spaces Visualization

D. I. Migranov

Novosibirsk State University
Novosibirsk, Russian Federation

Abstract

In this paper we examine methods and algorithms for visualizing three-dimensional non-Euclidean spherical and elliptical spaces with a first-person view, from insider's perspective: we propose an algorithm for implementing camera controls in such spaces, address an approach to visualizing properties of such spaces based on matrix and vector operations which enables us to perform some of the computations necessary for the visualization on GPUs using graphics APIs, describe an implementation of fog effects in such spaces for the purpose of simplifying navigation in them. An implementation of these approaches and algorithms that is intended for visualizing dynamic scenes in such spaces is presented.

Keywords

non-Euclidean spaces, visualization, computer graphics, computer modelling

Acknowledgements

We gratefully thank Vladimir E. Zyubin for his valuable advices and suggestions

For citation

Migranov D. I. Some Aspects of Three-Dimensional Spherical and Elliptical Spaces Visualization. *Vestnik NSU. Series: Information Technologies*, 2020, vol. 18, no. 4, p. 54–65. (in Russ.) DOI 10.25205/1818-7900-2020-18-4-54-65

© Д. И. Мигранов, 2020

Введение

Неевклидовы геометрии и пространства находят свое применение в различных областях знаний. Разумеется, такие пространства можно рассматривать сугубо с точки зрения математики, однако их математическое описание может быть сложно для восприятия, особенно людьми, не знакомыми со спецификой предметной области. Возможным также является физическое моделирование, сводящееся к созданию физических объектов с определенной геометрией. Однако создание таких объектов довольно трудоемко; кроме того, этот способ ограничивается двумерным случаем. Возможности же компьютерного моделирования и компьютерной графики позволяют не только визуализировать свойства неевклидовых пространств в наглядной форме, но и предоставить пользователю возможность взаимодействовать с ними.

Одним из подходов к изучению геометрий является их рассмотрение с точки зрения групп пространственных преобразований и свойств, которые остаются инвариантными относительно этих преобразований [1]. Данный подход восходит к Эрлангенской программе Клейна и может быть применен для визуализации свойств однородных пространств, включая и обычное евклидово, и гиперболическое (или пространство Лобачевского), и сферическое, и тесно связанное со сферическим эллиптическое (также известное как пространство Римана; оно может быть получено из сферического пространства путем отождествления его антиподальных точек). Важно то, что для задания преобразований объектов (т. е. их перемещений, поворотов) в этих пространствах можно использовать обычные матрицы (как это производится, например, в традиционной, евклидовой, компьютерной графике) [2]; проецирование объектов на экран также может быть произведено с помощью матриц. В то же время графические ускорители способны с высокой степенью параллелизма производить операции над матрицами и векторами, что обусловливает возможность их применения для визуализации свойств однородных неевклидовых пространств.

Визуализация однородных неевклидовых пространств может быть использована в образовательных и исследовательских целях для наглядной демонстрации свойств таких пространств, в том числе в игровых приложениях и системах виртуальной реальности. Визуализация однородных неевклидовых пространств может использоваться для демонстрации некоторых аспектов общей теории относительности (ОТО); например, для демонстрации свойств однородной и изотропной вселенной Фридмана. Однородные неевклидовы пространства также находят применение в задачах визуализации свойств различных информационных структур, например деревьев; так, для этой цели могут использоваться двумерное [3] и трехмерное [4] гиперболические пространства.

В данной работе рассматриваются методы и алгоритмы визуализации геометрических свойств трехмерных сферического и эллиптического пространств (гиперболическое пространство по своим свойствам больше схоже с евклидовым, так как не является замкнутым, в отличие от сферического и эллиптического); представлена реализация этих подходов и алгоритмов, позволяющая визуализировать сцены в таких пространствах. Существующие системы, позволяющие визуализировать те или иные свойства этих пространств, обладают рядом ограничений. Так, система Curved Spaces [1] предназначена сугубо для визуализации замощений неевклидовых пространств и не позволяет поместить в произвольное положение сцены отдельный объект и изучать его. Кроме того, управление в ней сильно отличается от привычного по многим системам визуализации и компьютерным играм с видом от первого лица: вращение камерой с помощью компьютерной мыши может приводить к крену камеры; кроме того, невозможно одновременно вращать камерой и перемещаться вправо или влево. Представленный и реализованный алгоритм управления камерой решает эти проблемы.

[1] *Weeks J.* Curved Spaces. URL: http://www.geometrygames.org/CurvedSpaces/index.html.en (дата обращения 12.05.2020).

Использование матриц для задания положений объектов в сферическом и эллиптическом пространствах

Как уже было сказано, избранный подход позволяет использовать для визуализации сцен в неевклидовых пространствах обычные матрицы (отметим, что далее все вычисления и матрицы будут приведены в предположении, что используются вектор-строки, на которые матрицы умножаются справа). Прежде чем перейти к рассмотрению собственно сферического и эллиптического пространств, обратимся к евклидову случаю и рассмотрим, элементы каких групп могут использоваться для задания положений в евклидовом пространстве и как они применяются в компьютерной графике.

Напомним, что движениями пространства называются преобразования, сохраняющие его метрику [5]. Группой движений евклидова пространства размерности n является группа $Euc(n)$ (евклидова группа), представляющая собой полупрямое произведение группы сдвигов начала координат (изоморфна группе векторов $\mathbb{R}^n$ относительно операции сложения) и группы n-мерных вращений вокруг начала координат (изоморфна ортогональной группе $O(n)$). Таким образом, элементами группы $Euc(n)$ являются пары (A, ξ), групповая операция для которых определяется как $(A, \xi) * (B, \eta) = (AB, \xi + A\eta)$. Инвариантами этой группы являются, например, расстояния, углы, площади. Легко показать, что эта группа изоморфна группе матриц $(n+1)\times(n+1)$ вида

$$\begin{pmatrix} A & 0 \\ \xi & 1 \end{pmatrix}, \quad A \in 0(n), \quad \xi \in \mathbb{R}^n.$$

Такие матрицы (для случая $n = 3$) получили широкое распространение в современной компьютерной графике и используются в ней для позиционирования камеры и объектов на сцене, которая должна быть изображена. В более общем случае в компьютерной графике для этих целей используются элементы надгруппы евклидовой группы – аффинной группы, которая изоморфна полупрямому произведению группы сдвигов и полной линейной группы. Важно то, что все эти преобразования могут быть заданы квадратными матрицами (для трехмерного случая – размерностью 4). В компьютерной графике элементы этих групп обычно применяются для формирования так называемых мировой и видовой матриц (в англоязычной литературе они именуются *world* и *view matrix* соответственно). Предполагается, что на сцене, которая должна быть изображена, находится несколько объектов, каждый из которых состоит из множества вершин. Координаты вершин, формирующих каждый из этих объектов, представлены в так называемой модельной системе координат, в которой удобно определять один конкретный объект. Для позиционирования каждого из объектов на сцене, их вращения и перемещения, к вершинам каждого из объектов применяются мировые матрицы. Мировая матрица переводит координаты вершины из модельной в так называемую мировую систему координат, где расположены все объекты. Аналогично видовая матрица задает преобразование из мировой системы координат в видовую: это система координат, в которой воображаемая камера находится в начале координат, а ее взгляд направлен по положительному направлению оси Z, перевод в эту систему координат необходим для облегчения проецирования. Обе эти матрицы в евклидовом случае часто являются элементами евклидовой группы, хотя, например, мировая матрица может быть и элементом аффинной группы (например, если нужно произвести масштабирование объекта).

Аналогично движениям евклидова пространства можно рассматривать движения сферического пространства, также составляющие группу. Рассмотрим для начала группу движений обычной двумерной сферы, которую можно рассматривать как поверхность в трехмерном евклидовом пространстве. Преобразования, переводящие сферу в сферу и сохраняющие расстояния и углы на ней, составляют группу $O(3)$ – ортогональную группу размерностью 3 (это вращения пространства, собственные и несобственные). В общем случае группой движений

n-мерной сферы будет $O(n+1)$. Таким образом, для задания преобразований объектов в трехмерном сферическом пространстве могут использоваться элементы группы $O(4)$ – ортогональные матрицы. В данной работе для этого использовались элементы ее подгруппы, группы $SO(4)$, т. е. ортогональные матрицы с определителем, равным единице, задающие только собственные вращения. Так, элементом этой группы является матрица вращения R_{zw}, имеющая вид

$$R_{zw}(\varphi)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \cos\varphi & -\sin\varphi\\ 0 & 0 & \sin\varphi & \cos\varphi\end{pmatrix},$$

где φ – расстояние (в радианах), на которое будет сдвинута точка гиперсферы, к которой применена эта матрица. При применении этого преобразования точка (0, 0, 0, 1), которую далее будем рассматривать в качестве точки начала координат, будет сдвинута вперед в направлении оси *Z*. Такие матрицы (следуя использованной выше нотации, обозначим эти матрицы как R_{xw}, R_{yw}, R_{zw}) могут быть использованы для перемещения объектов по сцене (включая камеру). Другие элементы группы $SO(4)$, сохраняющие точку (0, 0, 0, 1) при их применении, могут быть использованы для вращения объектов (это матрицы R_{xy}, R_{yz}, R_{xz}). Такие матрицы совпадают с матрицами вращения для евклидова случая. Все эти матрицы, по аналогии с евклидовым случаем, могут быть использованы для формирования мировой и видовой матриц посредством умножения; так, если необходимо повернуть объект на α радиан вокруг оси *Y*, а потом переместить его на β радиан в направлении оси *Z*, его мировая матрица примет вид $R_{xz}(\alpha) * R_{zw}(\beta)$ (сначала выполняется поворот, затем перемещение). Вопрос о формировании видовой матрицы будет более подробно рассмотрен далее, при описании алгоритма реализации управления камерой.

Что касается эллиптического пространства, для задания положения объектов будут использоваться те же матрицы, что и в случае со сферическим пространством. Отличия между двумя этими случаями будут изложены далее, при описании алгоритмов рендеринга и реализации дымки.

Реализация управления камерой в сферическом и эллиптическом пространствах

Как было сказано выше, видовая матрица предназначена для перехода из мировой системы координат в видовую, в которой камера находится в начале координат, а ее взгляд направлен по положительному направлению оси *Z*. Она обратна матрице, переводящей точку начала координат (0, 0, 0, 1) в точку, где расположена камера, и придающей ей нужное направление, – матрице трансформации камеры. Эта матрица может быть найдена как $M = R \times T$, где *R* – матрица вращения, предназначенная для поворота камеры в необходимом направлении, а *T* – матрица трансляции, перемещающая камеру в необходимую точку (напомним, что рассматривается случай вектор-строк, т. е. сначала применяется вращение, потом трансляция). Поэтому видовая матрица может быть найдена по формуле $V = T' \times R'$, где *T'* – матрица, обратная *T*, а *R'* – матрица, обратная *R*. Ввиду этого дальнейшие рассуждения будут приведены в терминах матрицы трансформации камеры, так как, зная алгоритм ее нахождения, мы легко сможем найти видовую матрицу.

Как найти матрицу *R*? В евклидовом случае для реализации вращения камеры часто используются углы Эйлера (точнее, углы Тейта-Брайана): поворот камеры задается тремя углами, которые в литературе обозначаются как «тангаж», «рыскание», «крен» (по-англий-

ски *pitch*, *yaw* и *roll* соответственно). Этот подход предлагается использовать для формирования матриц и в случае рассматриваемых пространств. Для того чтобы реализовать вращение камеры с помощью мыши, необходимо отслеживать относительное перемещение указателя мыши по экрану в сравнении с предыдущим кадром. На основе полученных значений изменяются и накапливаются значения углов тангажа и рыскания: угол тангажа *pitch* изменяется пропорционально изменению положения мыши по оси Y в экранных координатах, угол рыскания *yaw* – по оси X; крен не учитывается. Дополнительно можно следить за тем, чтобы общий угол тангажа оставался в пределах от $-\pi/2$ до $+\pi/2$, чтобы не происходило переворота камеры, который может привести к дезориентации пользователя. После этого на основе полученных значений углов тангажа и рыскания можно рассчитать матрицы поворота $R_{yaw} = R_{xz}(yaw)$ и $R_{\text{pitch}} = R_{yz}(\text{pitch})$ вокруг осей Y и X соответственно. Итоговая матрица R получается как произведение R_{pitch} и R_{yaw}.

Перемещение реализуется посредством отслеживания нажатий на клавиши перемещения на клавиатуре. Если какая-то из клавиш, отвечающих за перемещение, нажата, формируется вектор $dV = (dx, dy, dz)$, где dx отвечает за перемещение влево или вправо, dy – вверх или вниз, dz – вперед или назад. В евклидовом случае обычно отслеживается положение камеры, которое обновляется путем прибавления повернутого, чтобы учесть текущее направление камеры, вектора dV; на основе положения камеры формируется матрица T. Однако в случае рассматриваемых пространств позицию камеры не так легко отслеживать: нельзя получить новое положение камеры путем простого прибавления вектора в декартовой системе координат, как это происходит в евклидовом случае. Поэтому был избран другой подход к формированию матрицы трансляции камеры, инкрементальный: изначально T – единичная матрица; при каждом нажатии на клавиши перемещения ее новое значение получается умножением матрицы, зависящей от dV, на значение матрицы T на предыдущем шаге (T_{prev}). Как было сказано выше, для перемещения объектов по сцене могут быть использованы матрицы R_{xw}, R_{yw}, R_{zw}. Однако матрица dT, сформированная путем умножения только таких матриц, не будет учитывать поворот камеры. Чтобы учесть его, необходимо изменить ось поворота, применяя известную формулу из линейной алгебры. Таким образом, получаем формулу для матрицы трансляции камеры:

$$T = R^T * dT * R * T_{\text{prev}}, \quad dT = R_{zw}(dz) * R_{yw}(dy) * R_{xw}(dx).$$

Можно также ограничить движение наблюдателя по пространству одной плоскостью, например, $y = 0$; в этом случае T будет находиться по формуле

$$T = R_{yaw}^T * dT * R_{yaw} * T_{\text{prev}}, \quad dT = R_{zw}(dz) * R_{xw}(dx).$$

Чтобы теперь найти положение камеры, надо всего лишь применить матрицу T к точке (0, 0, 0, 1). Знание матрицы трансформации камеры позволяет легко найти видовую матрицу, которая будет использована для изображения сцены с видом из воображаемой камеры, находящейся в данном положении. Отметим, что и R, и T формируются посредством перемножения элементов ортогональной группы, поэтому и сами они, и их произведение также являются ее элементами. По этой причине обратные им матрицы легко найти, применив транспонирование.

Для случая эллиптической геометрии алгоритм тот же.

Отрисовка сцен в трехмерных сферическом и эллиптическом пространствах

Кроме мировых и видовых матриц в компьютерной графике используются матрицы проекции, являющиеся элементами проективной группы. Матрица проекции задает преобразо-

вание из видового пространства в так называемое пространство отсечения (clip space). При этом все поле зрения камеры (ограниченное геодезическими рассматриваемого пространства) отображается в прямоугольный параллелепипед [–1, 1] × [–1, 1] × [0, 1] в нормализованных однородных координатах (так называемый clipping box). Точки, не входящие в поле видимости, оказываются за пределами параллелепипеда и отбрасываются (это осуществляется автоматически в ходе *этапа растеризации* в современных графических ускорителях). От итоговых значений координат x и y будет зависеть положение вершины на экране, z – отдаленность от камеры, что позволяет понимать, какие объекты находятся ближе, а какие – дальше, и изображать их в правильном порядке. Предполагается, что точки, к которым применяется преобразование проекции, находятся в видовой системе координат, поэтому вид матрицы проекции одинаков для всех положений камеры.

В случае визуализации свойств неевклидовых пространств нужны будут такие матрицы проекции, которые переводят геодезические рассматриваемого пространства в прямые евклидова пространства, так как графические ускорители работают именно с прямыми и в конечном счете при отображении на экран растрироваться будут именно отрезки прямых. Рассмотрим в качестве примера двумерную сферическую геометрию. Проекцией, удовлетворяющей этому требованию, является центральная (гномоническая) проекция, получаемая проектированием точек сферы из ее центра на плоскость. Действительно, геодезические сферы (большие окружности) являются сечениями сферы плоскостями, проходящими через ее центр. Тогда линии, соединяющие проецируемые точки геодезических с их проекциями на плоскость, будут лежать в этих плоскостях. Пересечения этих плоскостей с плоскостью, на которую осуществляется проекция, будут прямыми, являющимися искомыми проекциями геодезических.

Перейдем к трехмерному случаю. Для простоты описания далее будем рассматривать гиперсферу (трехмерное сферическое пространство) единичного радиуса. Все ее точки могут быть представлены в декартовых координатах (x, y, z, w), причем для всех точек $x^2 + y^2 + z^2 + w^2 = 1.$

Центральная проекция гиперсферы на гиперплоскость (т. е. обычное трехмерное евклидово пространство) $w = 1$ может быть получена простым делением всех компонент на четвертую координату w. Эта операция переводит точки гиперсферы в точки евклидова пространства, поэтому далее возможно спроецировать их на экран, применив обычную матрицу проекции. Изображение, полученное таким способом, будет в определенном смысле корректным, поле зрения, как показано в работе [6], будет ограничено экватором. Для решения этой проблемы в той же работе приводятся матрица проекции и алгоритм рендеринга, исправляющие этот недостаток и увеличивающие поле зрения (в оригинальной работе был приведен вид матрицы только для фиксированных соотношения сторон (1 : 1) и угла обзора в 90°; в данной работе был использован более общий вид матрицы):

$$M_{\text{pro}\,j} = \begin{pmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & 1/2 & 1 \\ 0 & 0 & -z_0/2 & 0 \end{pmatrix}, \quad h = \frac{1}{\operatorname{tg}\left(\dfrac{fovY}{2}\right)}, \quad w = \frac{h}{\text{aspect}}, \tag{1}$$

где z_0 – это расстояние в радианах, aspect – соотношение ширины клиентской области окна к высоте, $fovY$ – угол зрения по вертикали. В случае использования этой матрицы полем зрения с вертикальным углом обзора в $fovY$ и соотношением сторон aspect будет область, начинающаяся на расстоянии z_0 от камеры и заканчивающееся на расстоянии z_0 до достижения точки, противоположной точке, где находится камера. Можно также отметить схожесть этой матрицы с матрицей перспективной проекции, применяющейся в традиционной компьютерной графике. Применение матрицы (1) позволяет решить проблему с отображением точек, находящихся спереди наблюдателя ($z > 0$ в видовых координатах), но за экватором ($w < 0$):

значение координаты z после использовании обычной матрицы проекции становится для них слишком большим (а именно, большим единицы в однородных координатах), они не попадают в поле зрения и не отображаются на экране [7]. Использование этой матрицы также позволяет добиться соответствующего сферической геометрии изменения размеров объектов при их перемещении относительно наблюдателя.

Однако применение таких матриц все равно не позволит изобразить свойства сферического пространства во всей полноте: на экране будут изображены только объекты, находящиеся спереди наблюдателя, объекты же, находящиеся относительно него сзади (но которые обитатель такого пространства все равно может видеть ввиду поведения геодезических, формирующих поле зрения), не будут отрисованы. Дело в том, что при проверке того, какие точки попали в поле зрения, а какие нет (как было сказано выше, в современных графических ускорителях это осуществляется автоматически на этапе растеризации, программист при работе с графическими API не имеет над этим контроля), вместо уравнения $0 < Zp/Wp \leq 1$, где Zp и Wp – координаты в пространстве отсечения, в целях оптимизации используется уравнение $0 < Zp \leq Wp$. Фактически рассматривается только случай $Wp > 0$, что эквивалентно условию $Zv > 0$ в видовых координатах. В евклидовом случае это упрощение оправдано: точки, находящиеся позади наблюдателя ($Zv < 0$), и не должны быть изображены. Однако в случае со сферическим пространством это не так. Чтобы обойти эту проблему, каждый объект сцены необходимо отрисовать дважды, с двумя разными видовыми матрицами и матрицами проекции, как это было предложено в работе [6]. Первая пара матриц проекции и вида отвечает за отрисовку объектов, находящихся перед наблюдателем; в качестве матрицы вида будет взята матрица для положения, в котором находится камера, в качестве матрицы проекции – матрица проекции, переводящая поле зрения в переднюю половину параллелепипеда отсечения ($0 < Zp/Wp \leq 1/2$). Вторая пара отвечает за отрисовку объектов, находящихся позади наблюдателя, но которые он все равно будет видеть в силу свойств сферического пространства. В качестве матрицы вида будет взята матрица, соответствующая точке, противоположной положению камеры, а в качестве матрицы проекции – матрица, переводящая поле зрения в заднюю половину параллелепипеда отсечения ($1/2 < Zp/Wp \leq 1$). Этот подход также позволяет разделять объекты, находящиеся в передней и задней частях гиперсферы относительно наблюдателя: все объекты, расположенные в задней части будут отрисованы позади объектов, находящихся в передней части гиперсферы. Требуемые матрицы проекции примут вид

$$M_{\text{pro}\,j,\,\text{front}} = \begin{pmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & 1/4 & 1 \\ 0 & 0 & -z_0/4 & 0 \end{pmatrix}, \quad M_{\text{pro}\,j,\,\text{back}} = \begin{pmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & 3/4 & 1 \\ 0 & 0 & -z_0/4 & 0 \end{pmatrix},$$

где w и h имеют тот же смысл, что в формуле (1).

В случае рендеринга эллиптического пространства каждая сцена также должна быть отрисована дважды с двумя разными видовыми матрицами. Однако матрица проекции на этот раз будет иметь вид (1) для обоих случаев: в отличие от сферического пространства, в этом случае не будет двух частей гиперсферы, объем эллиптического пространства в два раза меньше объема сферического.

Дымка в сферическом и эллиптическом пространствах

До этого в наших рассуждениях мы практически не использовали метрические свойства пространства, нас не интересовали, например, расстояния между объектами; нам было достаточно знания того, что они сохраняются при движениях пространства. Однако рассмотрение этих свойств может облегчить ориентирование в таких пространствах и повысить реалистич-

ность получаемого изображения. Так, для облегчения ориентирования в пространствах со свойствами, настолько отличающимися от привычных человеку, может иметь смысл добавления эффекта дымки (тумана), который позволит лучше понимать, какие объекты находятся ближе, а какие – дальше. Эффект заключается в том, что объекты, расположенные дальше от камеры, предстают для наблюдателя менее контрастными, вплоть до полного исчезновения. Этот эффект, широко использующийся в компьютерной графике, встречается и в реальной жизни вследствие рассеяния и поглощения света частицами воздуха.

В компьютерной графике применяется множество моделей дымки. Многие из них основываются на смешении для каждого пикселя изображения двух цветов: исходного цвета пикселя *originalColor* и цвета дымки *fogColor*, в качестве которого можно рассматривать, например, цвет фона. Если используется альфа-смешивание, то результирующий цвет пикселя будет вычисляться как *fogFactor* * *originalColor* + (1 – *fogFactor*) * *fogColor*. Здесь *fogFactor* – коэффициент дымки, который может принимать значения от нуля до единицы. Как можно видеть, если значение коэффициента равно нулю, то пиксель примет значение, соответствующее цвету дымки. Если же его значение равно единице, то эффект дымки, наоборот, никак на нем не скажется.

Существует различные способы вычислить коэффициент дымки. Очевидно, что его значение должно зависеть от расстояния между камерой и объектом. А именно: если камера расположена близко к объекту, то значение коэффициента должно быть близко к единице, что снизит влияние эффекта дымки на объект. Напротив, если камера сильно удалена от объекта, значение коэффициента должно быть близко к нулю, в результате чего данный объект будет затуманен.

Одна из самых простых моделей дымки – линейная (linear fog): величина коэффициента дымки линейно интерполируется в зависимости от расстояния между изучаемой точкой объекта и камерой. Однако куда более точно отражает реальность другая модель – экспоненциальная дымка (exponential fog). В реальном мире интенсивность света (и, как следствие, видимость) уменьшается экспоненциально с увеличением расстояния в результате поглощения света частицами среды (в соответствие с законом Бугера – Ламберта – Бера). В случае экспоненциальной дымки значение коэффициента также будет экспоненциально уменьшаться с увеличением расстояния, и формула для его вычисления примет вид:

$$fogFactor = \exp(-distance * density),$$

где *density* – показатель поглощения (или плотность тумана).

В компьютерной графике также используется еще одна разновидность экспоненциальной дымки – квадратичная экспоненциальная дымка (exponential-squared fog). Использование этой модели дает более быстрое падение видимости с ростом расстояния. Формула для получения коэффициента дымки в этом случае примет вид

$$fogFactor = \exp(-(distance * density)^2).$$

Для вычисления коэффициента дымки в вершине необходимо вычислить расстояние между этой вершиной и камерой. Вычисление расстояния – это то, в чем будет заключаться различие в моделировании эффекта дымки для евклидова и неевклидова случаев.

Рассмотрим сначала случай сферической геометрии. Чтобы облегчить нахождение расстояния между камерой и точкой изучаемого объекта, применим сначала мировую, а затем видовую матрицы к вектору, в котором хранится положение вершины в четырехмерном пространстве, в результате чего будут получены координаты вершины в видовом пространстве, в котором камера всегда находится в точке (0, 0, 0, 1) и направлена по положительному направлению оси *Z*. Теперь задача сведена к тому, чтобы найти расстояние между точками *A* = (0, 0, 0, 1) и B – положение вершины в видовом пространстве. Мы легко можем посчитать длину хорды, связывающей две эти точки в объемлющем четырехмерном пространстве. Зная длину хорды *c* и радиус гиперсферы *r* (который, как мы условились, равен единице), мы

можем найти угол *AOB*, где *O* – центр гиперсферы. Тогда длина дуги *l* равна углу, умноженному на радиус (равный единице):

$$l = r * AOB = AOB = 2 * \arcsin\left(\frac{c}{2r}\right) = 2 * \arcsin\left(\frac{c}{2}\right). \quad (2)$$

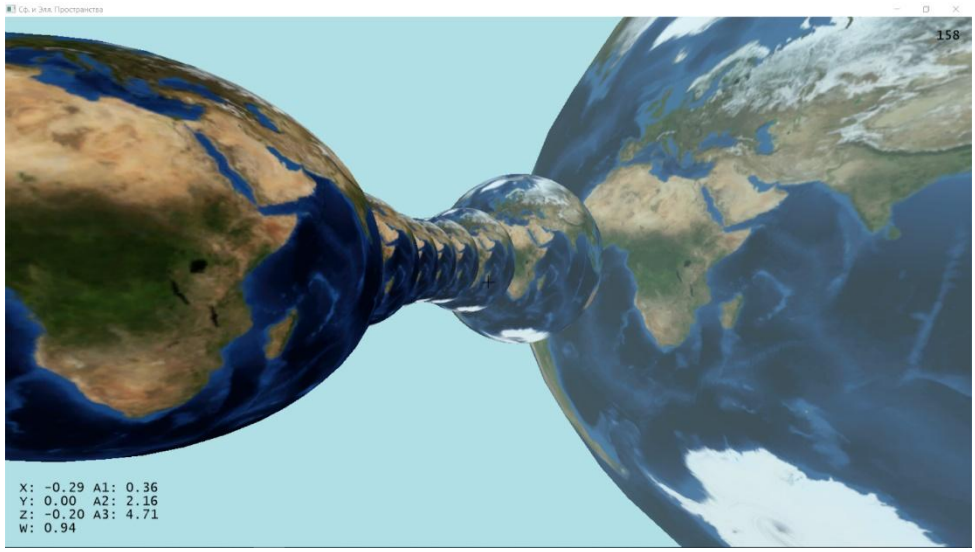
Таким образом, расстояние между двумя точками будет измеряться в радианах. Так как длина хорды всегда положительна, расстояние между двумя точками на гиперсфере, вычисленное подобным образом, будет лежать в пределах от 0 до π. Эта формула позволяет найти кратчайшее расстояние между двумя точками на гиперсфере, однако применительно к нашей задаче ее необходимо модифицировать, так как нам необходимо расстояние, пройденное лучом, выпущенным из камеры, а не просто длина дуги между двумя точками. Если рассматриваемая точка находится в передней половине гиперсферы относительно наблюдателя, то расстояние находится по формуле (2). Если же точка находится позади наблюдателя (напомним, в этом случае будет использоваться матрица вида для точки, антиподальной той, где находится камера), к расстоянию, вычисленному по формуле (2), надо будет также прибавить π. Расстояние, вычисленное подобным образом, будет лежать в пределах от 0 до 2π, что соответствует нашим требованиям. Больше расстояние быть не может из-за замкнутости и ограниченного объема пространства.

В случае эллиптической геометрии расстояние должно находиться в пределах от 0 до π (объем эллиптического пространства в два раза меньше объема сферического), для его вычисления должна применяться формула (2).

Реализация

Изложенные выше подходы были реализованы на языке C++ с использованием графического API Direct3D 11, который позволяет использовать возможности графических ускорителей для нужд компьютерной график и визуализации. Для отрисовки объектов сцены дважды на каждом кадре (с разными матрицами, но с одной и той же информацией о вершинах, формирующих объекты) использовался механизм инстансинга (geometry instancing), что позволяет сократить количество вызовов отрисовки на каждом кадре. С помощью реализованных классов и методов возможно добавление объектов на сцену (объект можно задать, определив информацию о его точках программно, загрузив эту же информацию из файла или использовав один из предопределенных типов фигур), их текстурирование с помощью загруженных из файла текстур. Каждому объекту сцены в соответствие также могут ставиться обновители, отвечающие за обновление положений объектов сцены в зависимости от прошедшего времени или поступившего пользовательского ввода, что позволяет создавать интерактивные и динамические сцены; каждый объект сцены также может иметь родительский объект, в этом случае движение объекта-ребенка привязано к движению объекта-родителя. Пользователь может перемещаться по сферическому и эллиптическому пространствам с видом от первого лица, возможно динамическое переключение между ними. Исходные тексты системы размещены в открытом доступе по адресу github.com/dmigranov/SphEll3D.

Пример изображения, получаемого с помощью системы, приведен на рисунке. На примере этого рисунка можно также видеть, что, в отличие от евклидова пространства, где видимые размеры объектов с увеличением расстояния между ними и наблюдателем линейно уменьшаются, в эллиптическом пространстве видимые размеры объектов начинают увеличиваться после их удаления на расстояние $\pi r/2$ от наблюдателя (ввиду поведения геодезических, формирующих его поле зрения).



Восемь текстурированных объектов одного размера
в трехмерном эллиптическом пространстве, эффект дымки
Eight same-sized textured objects in three-dimensional elliptical space, fog effect

Скорость работы системы зависит от количества треугольников, формирующих объекты сцены, и разрешения окна. На тестовой конфигурации (процессор Intel Core i5-8250U, видеокарта NVIDIA GeForce MX130, 8 ГБ оперативной памяти) были проведены испытания с различным количеством движущихся текстурированных объектов, были получены следующие результаты (разрешение окна 1600 × 1200, вертикальная синхронизация выключена): 124.07 кадра в секунду для 361000 треугольников, 62.93 – для 722000, 33.01 – для 1444000.

Заключение

В работе рассмотрены методы и алгоритмы визуализации геометрических свойств трехмерных сферического и эллиптического пространств с видом от первого лица. Предложен алгоритм реализации управления камерой в таких пространствах, рассмотрен подход к визуализации свойств таких пространств, описана методика реализации эффекта дымки в них. Представлена реализация данных методов и алгоритмов.

В качестве дальнейшего направления исследований можно рассматривать исследование методов визуализации свойств трехмерных неоднородных пространств с помощью математического аппарата римановой (псевдоримановой) геометрии, что может использоваться, например, для демонстрации свойств пространства согласно ОТО, в частности для визуализации свойств пространства вблизи массивных объектов и черных дыр [8]. Исследовательский интерес могут представлять использование систем виртуальной реальности для визуализации трехмерных неевклидовых пространств [9] и визуализация свойств однородных трехмерных геометрий Терстона, к которым относятся не только евклидова, сферическая и гиперболическая геометрии, но и пять других [10].

Список литературы

1. **Gunn C.** Discrete groups and visualization of three-dimensional manifolds. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, 1993, p. 255–262. DOI 10.1145/166117.166150
2. **Phillips M., Gunn C.** Visualizing hyperbolic space: Unusual uses of 4x4 matrices. In: Proceedings of the Symposium on Interactive 3D Graphics, 1992, p. 209–214. DOI 10.1145/147156.147206

3. **Lamping J., Rao R.** Laying out and Visualizing Large Trees Using a Hyperbolic Space. In: Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology, UIST 4, 1994, p. 13–14. DOI 10.1145/192426.192430
4. **Munzner T., Burchard P.** Visualizing the structure of the World Wide Web in 3D hyperbolic space. In: Proceedings of the Annual Symposium on the Virtual Reality Modeling Language, VRML, 1995, p. 33–38. DOI 10.1145/217306.217311
5. **Дубровин Б. А., Новиков С. П., Фоменко А. Т.** Современная геометрия: методы и приложения. М.: Наука, 1979.
6. **Weeks J.** Real-time rendering in curved spaces. In: IEEE Computer Graphics and Applications, 2002, vol. 22, iss. 6, p. 90–99. DOI 10.1109/MCG.2002.1046633
7. **Gunn C.** Advances in Metric-neutral Visualization. In: 2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa, Workshop Proceedings, 2010, p. 17–26.
8. **Yamashita Y.** Implementing a rasterization framework for a black hole spacetime. *Journal of Information Processing*, 2016, vol. 24, no. 4, p. 690–699. DOI 10.2197/ipsjjip.24.690
9. **Weeks J.** Non-Euclidean Billiards in VR. In: Bridges 2020 Conference Proceedings, 2020, p. 1–8.
10. **Novello T., Silva V. da, Velho L.** Visualization of Nil, Sol, and SL2(R)˜ geometries. *Computers and Graphics* (Pergamon), 2020, vol. 91. DOI 10.1016/j.cag.2020.07.016

References

1. **Gunn C.** Discrete groups and visualization of three-dimensional manifolds. In: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH, 1993, p. 255–262. DOI 10.1145/166117.166150
2. **Phillips M., Gunn C.** Visualizing hyperbolic space: Unusual uses of 4x4 matrices. In: Proceedings of the Symposium on Interactive 3D Graphics, 1992, p. 209–214. DOI 10.1145/147156.147206
3. **Lamping J., Rao R.** Laying out and Visualizing Large Trees Using a Hyperbolic Space. In: Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology, UIST 4, 1994, p. 13–14. DOI 10.1145/192426.192430
4. **Munzner T., Burchard P.** Visualizing the structure of the World Wide Web in 3D hyperbolic space. In: Proceedings of the Annual Symposium on the Virtual Reality Modeling Language, VRML, 1995, p. 33–38. DOI 10.1145/217306.217311
5. **Dubrovin B. A., Novikov S P., Fomenko A. T.** Modern geometry: methods and applications. Moscow, Nauka, 1979. (in Russ.)
6. **Weeks J.** Real-time rendering in curved spaces. In: IEEE Computer Graphics and Applications, 2002, vol. 22, iss. 6, p. 90–99. DOI 10.1109/MCG.2002.1046633
7. **Gunn C.** Advances in Metric-neutral Visualization. In: 2nd International Workshop on Computer Graphics, Computer Vision and Mathematics, GraVisMa, Workshop Proceedings, 2010, p. 17–26.
8. **Yamashita Y.** Implementing a rasterization framework for a black hole spacetime. *Journal of Information Processing*, 2016, vol. 24, no. 4, p. 690–699. DOI 10.2197/ipsjjip.24.690
9. **Weeks J.** Non-Euclidean Billiards in VR. In: Bridges 2020 Conference Proceedings, 2020, p. 1–8.
10. **Novello T., Silva V. da, Velho L.** Visualization of Nil, Sol, and SL2(R)˜ geometries. *Computers and Graphics* (Pergamon), 2020, vol. 91. DOI 10.1016/j.cag.2020.07.016

Материал поступил в редколлегию
Received
16.09.2020

Сведения об авторе

Мигранов Денис Игоревич, студент, 1 курс магистратуры, факультет информационных технологий, Новосибирский государственный университет (Новосибирск, Россия)

migranov.di@gmail.com
ORCID 0000-0001-5581-0498

Information about the Author

Denis I. Migranov, first-year Master's student, Department of Information Technologies, Novosibirsk State University (Novosibirsk, Russian Federation)

migranov.di@gmail.com
ORCID 0000-0001-5581-0498