МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Ярославский государственный университет им. П.Г. Демидова» (ЯрГУ)

Кафедра математического моделирования

«Допустить к защите»
Зав. кафедрой,
д. ф.-м. н., профессор
_____ С.А. Кащенко
«_____»_____2015

Выпускная квалификационная работа бакалавра

Исследование уравнения с распределенным запаздыванием

Научный руководитель
канд. физ.–мат. наук, доцент
_____ Д.С. Глызин
«_____»_____2015

Студент группы ПМИ-42 БО
_____А.Ф. Самсонов
«_____»_____2015

Ярославль 2015г.

Содержание

1. Введение

3D-принтер — это периферийное устройство, использующее метод послойного создания физического объекта по цифровой 3D-модели. В настоящее время это очень быстроразвивающаяся область, имеющая массу применений и позволяющая изготавливать объекты, который невозможно создать другими способами. Существует несколько технологий, с помошью которых происходит создание итогового объекта

- Лазерная стереолитография
- Лазерное сплавлени
- Ламинирование
- Застывание материала при охлаждении
- Полимеризация фотополимерного пластика под действием ультрафиолетовой лампы
- Склеивание или спекание порошкообразного материала

Все варианты печати предусматривают использование некоторой трехменой модели для создания программы печати, например на кафедре матемитического моделирования ЯРГУ, некоторые модели, печатабщиеся там представляют из себя визаулизацию некоторых математических функций. При задании модели финкцией получается модель теоретически неограниченной точности, но зачастую в такой модели образуется большое

число полигонов, которые находятся внутри модели и не имеют смысла как для печати, так и для рендеринга. В настоящей работе целью была разработка программы, предназначенной для оптимизации таких моделей т.е. отсечения геометрии, которая находится внутри модели, будет называть такую геометрию внутренней.

1.1. Функциональное название

1.2. Реализация программного комплекса

2. Входные данные

2.1. STL

STL (от англ. stereolithography) – открытый формат файла используемый для хранения 3D моделей и является основной для большинства 3D принтеров, за исключением внутренних форматов, которые рассматривать не будем, т.к. число этих форматов постоянно увеличивается и, фактически, большинство разработчиков аппаратуры 3D печати создают собственные форматы под свои нужды. Информация в STL может хранится как в текстовом, так и в двоичном виде. Вся геометрия задается в виде наборов точек, составляющих треугольник и соответствующую им нормаль, которая не используется при печати. Этот формат был выбран основным форматом для программы по причине его простоты, достаточности для целей не цветной печати и большой распространенности. Формат используется как для загрузки моделей, так и для сохранения.

2.2. Obj

Более продвинутый формат, чем STL, разработанный в Wavefront Technologies. Формат так-же является открытым и очень распространен, поддерживается большинством современных 3D редакторов, но в отличии от STL имеет более расширенные возможности, такие как текстурные координаты, нетреугольные полигоны, а так же дополнительные файлы, описывающие материа-

лы, используемые для текстурирования. Используется в проекте для загрузки моделей, выбран по причине простоты и большой распространенности.

3. Расчет досягаемости геометрии

Элементы поверхности. Первым шагом алгоритма будет упрощение данных о геометрии. Упрощение сводит каждый полигон к окружности, имеющей координаты центра, нормаль и площадь, оно позволит сильно упростить расчеты того, насколько одна из поверхностей затеняет другую. Центром окружности является усредненное значение координат всех точек полигона.

$$\sum_{i=0}^{n} x_i/n \tag{1}$$

Нормаль вычисляется по трем любым точкам (a, b, c) по следующей формуле:

$$\begin{aligned} w &= (c-a) \times (b-a) \\ n &= w \cdot 1/\sqrt{\sum_{i=0}^{n} w_i^2} \end{aligned} \tag{2}$$

Площадь треугольника вычислим по формуле Герона.

$$\begin{aligned} p &= \frac{a+b+c}{2} \\ S &= \sqrt{p(p-a)(p-b)(p-c)} \end{aligned} \tag{3}$$

Ambient occlusion – полезная техника для затенения объектов, которая используется в современной графике. Благодаря ambient occlusion получаются мягкие тени за счет затемнения поверхностей, которые частично видны

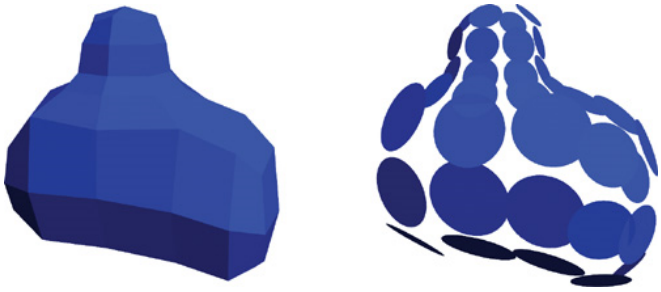

Рис. 1. Перевод полигонов в диски той же площади

с некоторой внешней точки. Эта техника также включает в себя расчет коэффициента досягаемости полигона к окружению, или, другими словами, число обратное числу пересечений его с другими полигонами в полусфере перед первым.

Будем называть элемент, на который падает тень ресивером, а элемент бросающий тень – эммитером. Для расчетов количества тени, которая бросается эммитером на ресивер используется формула, основанная на телесном угле для ориентированного диска

$$\frac{r \cos \Theta_E \max(1, 4 \cos \Theta_R)}{\sqrt{\frac{A}{\pi} + r^2}} \tag{4}$$

Где – площадь эммитера
E – эммитер
R – ресивер
RE – отрезок, соединяющий центры дисков
r – длина отрезка RE, расстояние между дисками
Θ_R – угол между нормалью ресивера и отрезком ER
Θ_E – угол между нормалью эммитера и отрезком ER

Если вычислить значение затенения каждого полиго-

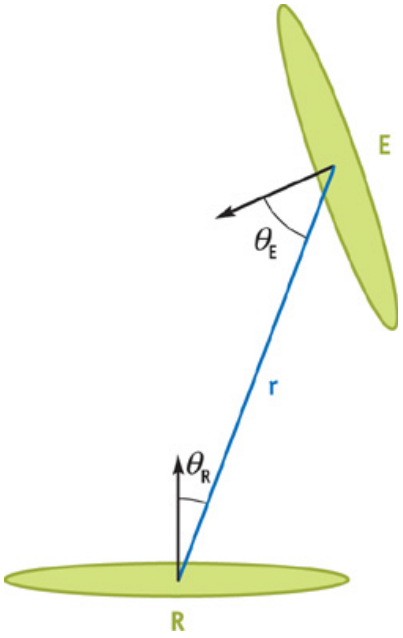

Рис. 2. визуализация расположения эммитера и ресивера и эелементов уравнения 4

на с каждым другим, то суммарная величина и будет являться коэффициентом самозатенения этого полигона. Полигоны, имеющие относительно низкие значения этого коэффициента более вероятно являются оболочкой, нежели находятся внутри и наоборот. Для полигонов, находящихся внутри модели коэффициент будет близок в единице. Полигоны, имеющие некоторые средние значения коэффициента могут как находится внутри модели, так и являться оболочкой. Для корректного удаления такой геометрии при отсечении используется вводимая пользователем граница отсечения. В случае слишком агрессивного, или наоборот, отсечения можно отменить сделанные изменения и повторить процедуру с другим коэффициентом. Пересчета самозатенения для этого не требуется, поэтому процедура выполняется в реальном времени.

4. Заключение

Список литературы

[1] GPU Gems 2: Programming Techniques For High-Performance Graphics And General-Purpose Computation. Matt Pharr, Randima Fernando. Pearson Addison Wesley Prof, 2005 -Всего страниц: 814

[2] Christensen, P. H. 2002. Note #35: Ambient Occlusion, Image-Based Illumination, and Global Illumination.

[3] Chunhui, M., Jiaoying, S., and Fuli, W. 2004. Rendering with Spherical Radiance Transport Maps. Computer Graphics.

[4] Deering, M., Winner, S., Schediwy, B., Duffy, C., and Hunt, N. 1988. The Triangle Processor and Normal Vector. Shader: A VLSI System for High Performance Graphics.

[5] Iones, A., Krupkin, A., Sbert, M., and Zhukov, S. 2003. Fast, Realistic Lighting for Video Games.

[6] Kautz, J., Lehtinen, J., and Aila, T. 2004. Hemispherical Rasterization for Self-Shadowing of Dynamic Objects.

[7] Sattler, M., Sarlette, R., Zachmann, G., and Klein, R., 2004. Hardware-accelerated ambient occlusion computation. To appear in the proceedings of International Fall Workshop on Vision, Modeling, and Visualization 2004.

[8] http://www.cadcamcae.lv/hot/STL_n23_p64.pdf

[9] http://www.fabbers.com/tech/STL_Format

[10] http://rodrigo-silveira.com/
opengl-tutorial-parsing-obj-file-blender

Приложение А

```
public void RecalcNormals()
{
    for (int i = 0; i < Verteces.Count; i += 3)
    {
        var a = Verteces[i].Position;
        var b = Verteces[i + 1].Position;
        var c = Verteces[i + 2].Position;
        var normal = Vector3.Normalize(
              Vector3.Cross(c - a, b - a));
        var i0 = Verteces[i];
        var i1 = Verteces[i + 1];
        var i2 = Verteces[i + 2];

        i0.Normal = i1.Normal = i2.Normal =
            normal;
        Verteces[i] = i0;
        Verteces[i + 1] = i1;
        Verteces[i + 2] = i2;
    }
    FlipNormals();
}
public void FlipNormals() {
    for (int i = 0; i < Verteces.Count; i++) {
        var vertex = Verteces[i];
        vertex.Normal *= -1;
        Verteces[i] = vertex;
    }
```

}

Приложение Б

```
__kernel void floatSquareDivPi(
    __global float * vx,
    __global float * vy,
    __global float * vz,
    __global float * nx,
    __global float * ny,
    __global float * nz,
    __global float * a,
    __global float * s)
{
    int i = get_global_id(0)*3;
    float3 vec1 = (float3)(vx[i],
              vy[i], vz[i]);
    float3 vec2 = (float3)(vx[i+1],
               vy[i+1], vz[i+1]);
    float3 vec3 = (float3)(vx[i+2],
              vy[i+2], vz[i+2]);
    float A = distance(vec1, vec2);
    float B = distance(vec2, vec3);
    float C = distance(vec1, vec3);
    float S = (A + B + C) / 2.0f;
    float sq = sqrt(S * (S - A) *
            (S - B) * (S - C)) / M_PI;
    s[i] = s[i+1] = s[i+2] = sq;
}

float ElementShadow(float3 v,
```

```
        float rSquared,
        float3 receiverNormal,
        float3 emitterNormal,
        float emitterArea) {
    return (1.0f - rsqrt(emitterArea/
            rSquared + 1.0f)) *
    clamp(dot(emitterNormal, v),
            0.0f, 1.0f) *
    clamp(4.0f * dot(receiverNormal,
            v), 0.0f, 1.0f);
}

__kernel void floatAO(
            __global float * vx,
            __global float * vy,
            __global float * vz,
          __global float * nx,
            __global float * ny,
            __global float * nz,
          __global float * a,
            __global float * s,
            long from)
{
    int i = from + get_global_id(0)*3;
    float res = 0.0f;
    float3 v = (float3)(0, 0, 0);
    float d2 = 0;
    float value = 0;
```

```
        for(int j = 0; j<count;j+=3){
            v = (float3)(vx[j], vy[j], vz[j]) -
                    (float3)(vx[i], vy[i], vz[i]);
            d2 = dot(v, v) + 1e-16;
            v *= rsqrt(d2);
            value = ElementShadow(v, d2,
            (float3)(nx[i], ny[i], nz[i]),
            (float3)(nx[j], ny[j], nz[j]),
            s[j]);
            res += value;
        }
    a[i] = a[i+1] = a[i+2] = 1.0f - res;
}
```

Приложение В

00:01:43.419	
00:01:44.241	
00:01:32.125	Среднее: 00:01:40.2732
00:01:44.158	
00:01:37.423	

00:00:30.205	
00:00:29.944	
00:00:30.072	Среднее: 00:00:30.0788
00:00:29.832	
00:00:30.341	

00:00:02.527	
00:00:02.502	
00:00:02.509	Среднее: 00:00:02.5196
00:00:02.549	
00:00:02.511	