Компьютерная графика

Лекция 1. Растеризация отрезков

к.ф.-м.н. Филонов Павел Владимирович
filonovpv@gmail.com

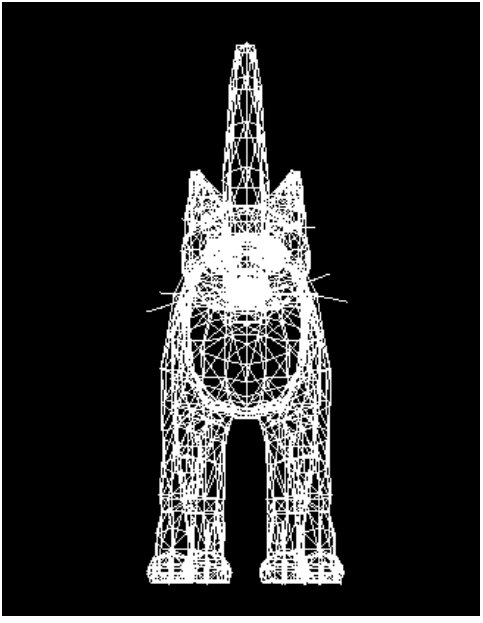
Московский Государственный Технический Университет
Гражданской Авиации

11 января 2017 г.

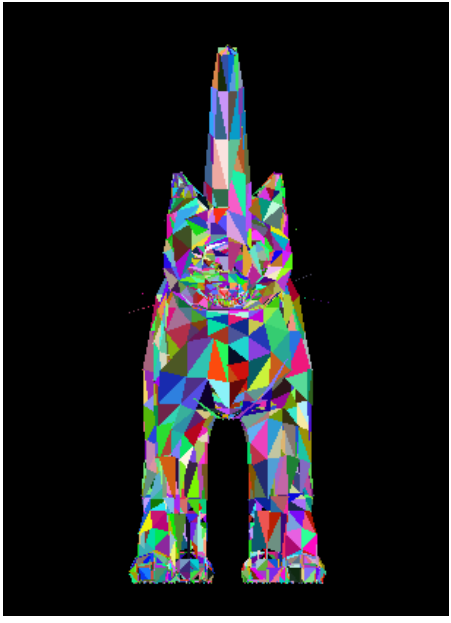
Знакомьтесь! Ребята, это Алиса!

```
...
v -0.003628 0.483028 0.926347
v -0.026661 0.485130 0.905901
v -0.025328 0.485275 0.907261
vt 0.429932 0.989021
vt 0.275879 0.197754
...
vn 0.930357 0.306925 0.200476
vn -0.588031 0.129856 0.798334
vn 0.710776 0.181555 -0.679556
f 1768/1/1 1763/2/2 1777/3/3
f 4/4/4 2/5/5 3/6/6
f 7/9/9 11/11/11 10/13/13
f 10/13/13 12/14/14 7/9/9
...
```

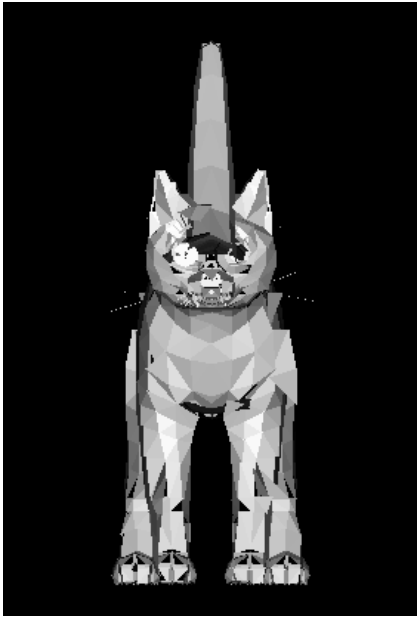
Сетчатая модель



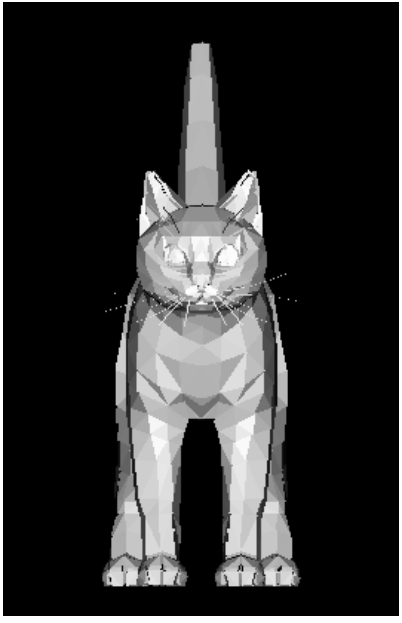
Полигональная модель



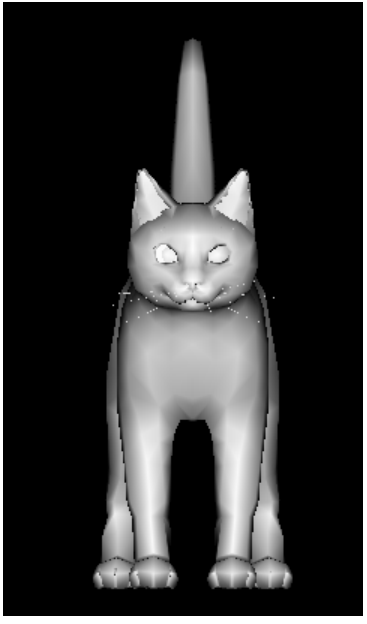
Полигональная с освещением модель



Удаление невидимых граней



Модель освещения Гуро



Наложение текстур

Преобразования моделей

Библиотека для работы с форматом TGA

https://github.com/sdukshis/ComputerGraphics

```
1  typedef struct tgaImage_t {
2      unsigned int height;
3      unsigned int width;
4      unsigned char bpp; /* bytes per pixel */
5      unsigned char *data; /* points on byte array */
6  } tgaImage;
7
8  enum tgaFormat {
9      GRAYSCALE = 1,
10     RGB = 3,
11     RGBA = 4
12 };
```

Создание и сохранение изображений

```
1  tgaImage * tgaNewImage(unsigned int height,
2                         unsigned int width,
3                         int format);
4
5  void tgaFreeImage(tgaImage *);
6
7  int tgaSaveToFile(tgaImage *, const char *filename);
8
9  tgaImage * tgaLoadFromFile(const char *filename);
```

Работа с цветом

```
1  typedef unsigned int tgaColor;
2
3  tgaColor tgaRGB(unsigned char r, unsigned char g, unsigned char b);
4
5  unsigned char Red(tgaColor);
6
7  unsigned char Blue(tgaColor);
8
9  unsigned char Green(tgaColor);
```

Работа с пикселями

```
1 int tgaSetPixel(tgaImage *, unsigned int x, unsigned int y, tgaColor);
2
3 tgaColor tgaGetPixel(tgaImage *, unsigned int x, unsigned int y);
4
5 void tgaFlipVertically(tgaImage *);
6
7 void tgaFlipHorizontally(tgaImage *);
```

Пример: множество Жюлиа

Множество точек, начиная с которых динамическая система

$$z_{n+1} = z_n^2 + c, \quad z, c \in \mathbb{C}$$

не стремится к бесконечности.

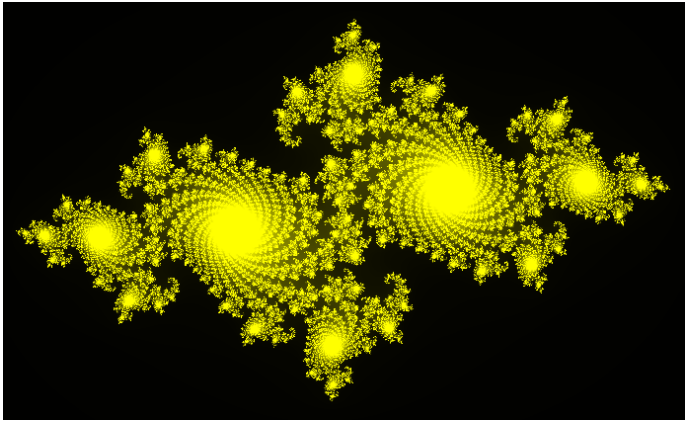


Рис.: $c = -0.7269 + 0.1889i$

```
1  void draw_julia_set(tgaImage *image,
2                      double x1, double x2, double y1, double y2,
3                      complex c, int maxiter) {
4    double x_step = (double)(x2 - x1) / image->width;
5    double y_step = (double)(y2 - y1) / image->height;
6    int i, j;
7    double x, y;
8    for (i = 0, x = x1; i < image->width; ++i, x += x_step) {
9      for (j = 0, y = y1; j < image->height; ++j, y += y_step) {
10       double intensity =
11           (1.0 -
12            (double)calculate_z(maxiter, x + y * _Complex_I, c) / maxiter);
13       tgaColor color = tgaRGB(255 * intensity, 255 * intensity, 0);
14       tgaSetPixel(image, i, j, color);
15     }
16   }
17 }
```

```
1 int calculate_z(int maxiter, complex z, complex c) {
2   while (maxiter-- && (cabs(z) < 2)) {
3     z = z * z + c;
4   }
5   return maxiter;
6 }
```

```
1  int main(int argc, char *argv[]) {
2    double x1 = -1.8, x2 = 1.8, y1 = -1.8, y2 = 1.8;
3    complex c = -0.7269 + 0.1889 * _Complex_I;
4
5    int width = 800, height = 800, max_iterations = 300_
6
7    tgaImage *image = tgaNewImage(width, height, RGB);
8
9    draw_julia_set(image, x1, x2, y1, y2, c, max_iterations);
10
11   if (-1 == tgaSaveToFile(image, argv[1])) {
12     perror("tgaSateToFile");
13     return EXIT_FAILURE;
14   }
15
16   tgaFreeImage(image);
17   return EXIT_SUCCESS;
18 }
```

Алгоритмы растеризации отрезков

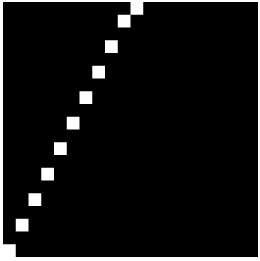
1. Цифровой дифференциальный анализатор
2. Алгоритм Брезенхема
3. Целочисленный алгоритм Брезенхема

Цифровой дифференциальный анализатор

$\mathrm{DDA}(x_0, x_1, y_0, y_1)$

1 ▷ Вычисляем тангенс угла наклона прямой
2 $k \leftarrow \frac{y_1 - y_0}{|x_1 - x_0|}$
3 $y \leftarrow y_0$
4 **for** $x \leftarrow x_0$ **to** x_1
5 **do** $\mathrm{PUTPIXEL}(x, y)$
6 $y \leftarrow y + k$

(a) $x_0 = 0, x_1 = 19, y_0 = 19, y_1 = 15$



(b) $x_0 = 0, x_1 = 19, y_0 = 10, y_1 = 0$

Цифровой дифференциальный анализатор

$\mathrm{DDA}(x_0, x_1, y_0, y_1)$

```
 1  if |y1 − y0| > |x1 − x0| ▷ Если угол наклона крутой, то меняем оси местами
 2     then steep ← TRUE
 3          SWAP(x0, y0)
 4          SWAP(x1, y1)
 5     else steep ← FALSE
 6  if x0 > x1 ▷ Если концевая точка левее начальной, то меняем их местами
 7     then SWAP(x0, x1)
 8          SWAP(y0, y1)
 9  k ← (y1 − y0) / |x1 − x0| ▷ Вычисляем тангенс угла наклона прямой
10  y ← y0
11  for x ← x0 to x1
12        do if steep
13              then PUTPIXEL(y, x)
14              else PUTPIXEL(x, y)
15           y ← y + k
```

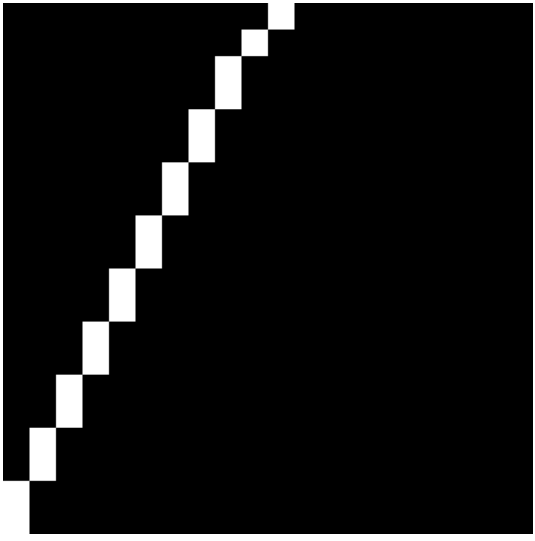
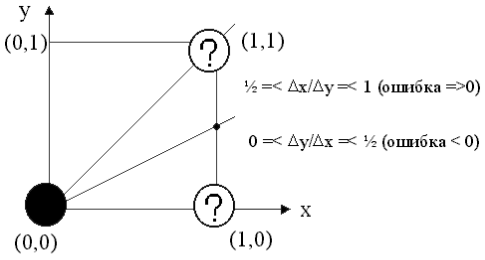
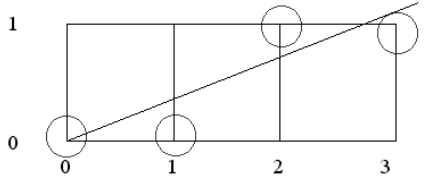


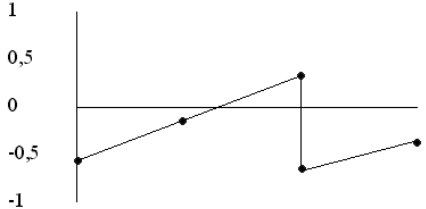
Рис.: $x_0 = 0, x_1 = 19, y_0 = 10, y_1 = 0$

Алгоритм Брезенхема



Инициализировать ошибку в −¼
ошибка = ошибка + Δy/Δx





Алгоритм Брезенхема

BREZENHEM(x_0, x_1, y_0, y_1)

```
 1  ▷ Подготовливаем точки как в предыдущей версии
 2  Δx ← x1 − x0
 3  Δy ← y1 − y0
 4  Δe ← |Δy| / Δx
 5  e ← 0
 6  y ← y0
 7  for x ← x0 to x1
 8        do if steep
 9              then PUTPIXEL(y, x)
10              else PUTPIXEL(x, y)
11           e ← e + Δe
12           if e > 0.5
13              then y ← y + sign(Δy)
14                   e ← e − 1
```

Целочисленный алгоритм Брезенхема

Произведем замену переменных

$$\tilde{e} = 2e\Delta x$$

BREZENHEM(x_0, x_1, y_0, y_1)

```
1  ▷ Подготовливаем точки как в предыдущей версии
2  Δx ← x1 − x0
3  Δy ← y1 − y0
4  Δẽ ← 2|Δy|
5  ẽ ← 0
6  y ← y0
7  for x ← x0 to x1
8        do if steep
9              then PUTPIXEL(y, x)
10             else PUTPIXEL(x, y)
11           ẽ ← ẽ + Δẽ
12           if ẽ > Δx
13             then y ← y + sign(Δy)
14                  ẽ ← ẽ − 2Δx
```

Формат Wavefront obj

Текстовый формат `obj` используется для хранения трёхмерных моделей.

```
# Это комментарий

# Список вершин с координатами (x, y, z)
v 0.123 0.234 0.345
...
# Текстурные координаты (u, v)
vt 0.500 -1.352
...
# Координаты нормалей (x, y, z). Могут быть не нормированы
vn 0.707 0.000 0.707
...
# Определения поверхности
f 6/4/1 3/5/3 7/6/5
...
```

Определение поверхностей

`f 6/4/1 3/5/3 7/6/5`

Поверхность состоит из треугольных граней. Каждая грань описывается отдельно строкой, которая начинается с символа **f**(face). В качестве параметров три вершины в формате **n**/**m**/**k**, где

- **n** — номер вершины (**v**)
- **m** — номер текстурной координаты (**vt**)
- **k** — номер нормали (**vn**)

Номера **m** и **k** необязательны.

Пример: поверхность из одного треугольника

```
1 # Координаты трёх вершин треугольника
2 v 0.0 0.0 0.0
3 v 1.0 0.0 0.0
4 v 0.0 1.0 0.0
5 # Описание грани
6 f 1 2 3
```

Функции для работы с obj файлами

```
1  // Грань состоит из 9 целочисленных номеров вершин, текстур и нормалей
2  typedef unsigned int Face[9];
3  // Каждая вершина описывается трехмерным вектором
4  typedef double Vec3[3];
5  // Основная структура для работы с моделью
6  typedef struct Model {
7      unsigned int nvert; // number of vertices
8      unsigned int ntext; // number of texture coords
9      unsigned int nnorm; // number of normals
10     unsigned int nface; // number of faces
11     Vec3 *vertices;
12     Vec3 *textures;
13     Vec3 *normals;
14     Face *faces;
15     tgaImage *diffuse_map;
16     tgaImage *normal_map;
17     tgaImage *specular_map;
```

Загрузка модели

```
1  // Загрузить модель из файла
2  Model * loadFromObj(const char *filename);
3
4  // Загрузка текстур
5  int loadDiffuseMap(Model *model, const char *filename);
6  int loadNormalMap(Model *model, const char *filename);
7  int loadSpecularMap(Model *model, const char *filename);
8
9  // Освободить занимаемую память
10 void freeModel(Model *);
```

Получение данных

```
1  // Вернуть координаты врешины с номером nvert(0, 1, 2) для конкретной грани
2  Vec3 * getVertex(Model *model, unsigned int nface, unsigned int nvert);
3  // Аналогично для текстурных координат
4  Vec3 * getDiffuseUV(Model *model, unsigned int nface, unsigned int nvert);
5  // Аналогично для координат нормали
6  Vec3 * getNorm(Model *model, unsigned int nface, unsigned int nvert);
7
8  // Вернуть цвет диффузной текстуры по координатам uv
9  tgaColor getDiffuseColor(Model *model, Vec3 *uv);
10
11 // Вернуть нормаль (n) для координаты uv карты нормалей
12 int getNormal(Model *model, Vec3 *n, Vec3 *uv);
```

Пример: сетчатая модель

```
1  int main(int argc, char const *argv[]) {
2      // Загружаем модель из файла
3      Model *model = loadFromObj(argv[1]);
4
5      // Создаем область для рисования
6      int height = 800;
7      int width = 800;
8      tgaImage * image = tgaNewImage(height, width, RGB);
9
10     // Рисуем сетчатую модель
11     meshgrid(image, model);
12
13     // Освобождаем ресурсы
14     tgaFreeImage(image);
15     freeModel(model);
16     return rv;
17 }
```

Сетчатая модель

```
1  void meshgrid(tgaImage *image, Model *model) {
2      int i, j; tgaColor white = tgaRGB(255, 255, 255);
3      for (i = 0; i < model->nface; ++i) {
4          int screen_coords[3][2]; // Переводим в экранные координаты
5          for (j = 0; j < 3; ++j) {
6              Vec3 *v = &(model->vertices[model->faces[i][3*j]])
7              screen_coords[j][0] = ((*v)[0] + 1) * image->width / 2;
8              screen_coords[j][1] = (1 - (*v)[1]) * image->height / 2;
9          }
10         // Рисуем 3 ребра
11         for (j = 0; j < 3; ++j) {
12             line(screen_coords[j][0],       screen_coords[j][1],
13                  screen_coords[(j+1)%3][0], screen_coords[(j+1)%3][1],
14                  white);
15         }
16     }
17 }
```

Закрепление

- Задача компьютерной графики
- Функции для работы с TGA форматом
- Целочисленный алгоритм Брезенхема
- Формат obj
- Перевод в экранные координаты
- Построение сетчатой модели