# Компьютерная графика 2017

# Формулировка задания №2

Сравнение алгоритмов аппроксимации полутонового изображения двухуровневым

---

6 сентября 2017 г.

# Формулировка задачи

Написать программу, на любом интересном Вам языке программирования (C, C++, C#, Java, JavaScript, Python, Julia, Rust, Octave,....). Программа должна прочесть заданный растровый файл, при необходимости преобразовать в полутоновое изображение (изображение в градациях серого), применить заданный пользователем алгоритм преобразования в двухуровневое изображение и сохранить в файл с новым именем.

# Алгоритмы аппроксимации полутонов, которые требуется сравнить

1. Пороговое усечение (округление до ближайшего / thresholding).
2. Случайный дизеринг (random dithering). В этом алгоритме в качестве порога для каждого пиксела используется своё случайное значение.
3. Упорядоченный дизеринг (ordered dithering). Порог выбирается не случайно, а из специально подобранных матриц порогов.
4. Диффузия ошибки (error diffusion) только вперёд по строке
5. Диффузия ошибки вперёд по строке для чётных и назад для нечётных
6. Диффузия ошибки по алгоритму Флойда-Штейнберга (с/без чередованием строк)

# Тестовые изображения

- одна градация серого на всё изображение (автоматическое тестирование? , чистый белый/черный, разные размеры растра)
- вертикальные или горизонтальные полосы
- `http://www.lagom.nl/lcd-test/img/gradient-h.png`
- `https://en.wikipedia.org/wiki/Standard_test_image`
- что-нибудь своё

# Округление до ближайшего

# Случайное возбуждение (случайный дизеринг)

# Матрицы порогов для упорядоченного возбуждения

$$D_2 = \begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}$$

$$D_n = \begin{pmatrix} 4D_{n/2} & 4D_{n/2} + 2U_{n/2} \\ 4D_{n/2} + 3U_{n/2} & 4D_{n/2} + U_{n/2} \end{pmatrix}$$

$$U_n = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

# Диффузия ошибки по алгоритму Флойда-Штейнберга

Четные строки обходим слева направо, распространяя ошибку округления вперёд и на следующую строку, со следующими весами:

| | X | 7/16 |
|---|---|---|
| 3/16 | 5/16 | 1/16 |

Нечетные строки обходим в обратном порядке, распространяя ошибку округления вперёд и на следующую строку со следующими весами:

| 7/16 | X | |
|---|---|---|
| 1/16 | 5/16 | 3/16 |

# Простейшая работа с растрами в Python

```
1  from PIL import Image
2  def roundToNearestDithering2(inputFileName,
       outputFileName, threshold = None):
3    if threshold is None:
4      threshold = 128
5    im = Image.open(inputFileName)
6    im = im.convert("L")
7    xSize, ySize = im.size
8    pixmap = im.load()
9    for y in range(0, ySize):
10     for x in range(0, xSize):
11       pixmap[x,y] = 255 if pixmap[x,y] > threshold
             else 0
12   im.save(outputFileName,"PNG")
```

# Julia?

```
1  using Images, TestImages, ImageView, Colors
2  function process_image(in_file_name, out_file_name,
       image_func)
3    img = convert(Image{Gray}, load(in_file_name))
4    image_func(img)
5    save(out_file_name,img)
6  end
7  function random_dithering!{T,N}(
8    img::AbstractImageDirect{T,N}
9  )
10   for iter in eachindex(img)
11     if img[iter] > rand(T)
12       img[iter] = one(T)
13     else
14       img[iter] = zero(T)
15     end
16   end
17 end
18 process_image("in.png","out.png",random_dithering!)
```

# C/C++???

Возможно, что самостоятельное чтение/запись файлов в формате PGM — это самое простое решение.

# Javascript

Если Вы знакомы с технологией HTML5/Canvas и Javascript или хотите с ней познакомиться, то Вам должно быть легко выполнить это задание с использованием Canvas+Image+Javascript. Объект Image представляет прямой доступ к массиву пикселов.

https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
http://www.html5canvastutorials.com/
http://www.w3schools.com/canvas/default.asp
http://www.html5canvastutorials.com/advanced/html5-canvas-save-drawing-as-an-image/

# Бонусные задания

- Применить к изображению варианты гамма преобразования с разными значениями gamma `http://www.cambridgeincolour.com/tutorials/gamma-correction.htm`
- загрузка цветного изображения с преобразованием в серое
- дизеринг с уменьшением числа градаций не до двух, а до большего числа
- алгоритм диффузии ошибки по кривой Гильберта
- какие-нибудь усложненные варианты диффузии ошибки с более сложным шаблоном, чем у Флойда-Штейнберга
- дизеринг цветных изображений (независимо каждую компоненту и другие варианты)