МИНИСТЕРСТВО ОБРАЗОВАНИЯ НОВОСИБИРСКОЙ ОБЛАСТИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ

НОВОСИБИРСКОЙ ОБЛАСТИ

«НОВОСИБИРСКИЙ КОЛЛЕДЖ ЭЛЕКТРОНИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ»

(ГБПОУ НСО «Н

КЭ и ВТ»)

ОТЧЁТ

учебной практики по теме:

«Как программисту создать картинку без Фотошопа»

По специальности:

09.02.07 «Информационные системы и программирование»

Выполнил: студент группы 9ИС-290К

Левицкий Данила Владимирович

Квалификация: программист

Проверила: Пронина Наталья Павловна

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Новосибирск. 2024 г.

Оглавление

[**Содержание** 1](#_Toc156235288)

[**Введение** 2](#_Toc156235289)

[**Теоретический обзор темы** 3](#_Toc156235290)

[**Обоснование выбранной среды разработки** 4](#_Toc156235291)

[**Постановка (формулировка) заданий** 6](#_Toc156235292)

[**Коды программ с комментариями** 7](#_Toc156235293)

[**Заключение** 12](#_Toc156235294)

## **Введение**

Как рисовать, когда не умеешь рисовать? Заставить компьютер делать это за тебя.?

Нет нужды брать в руки кисть, просто составьте алгоритм, описывающий шум Перлина или ряд Фибоначчи, а картину нарисует компьютер. Это называется процедуральным искусством (generative art, GA).

Проще говоря, GA — график некоторой функции. И если вам кажется, что график функции — это просто скучный набор линий, то как насчёт графика в виде Эйфелевой башни?

## **Теоретический обзор темы**

Шум Перлина — это градиентный шум, состоящий из набора псевдослучайных единичных векторов (направлений градиента), расположенных в определенных точках пространства и интерполированных функцией сглаживания между этими точками. Для генерации шума Перлина в одномерном пространстве необходимо для каждой точки этого пространства вычислить значение шумовой функции, используя направление градиента (или наклон) в указанной точке.

Шум Перлина широко используется в двухмерной и трёхмерной компьютерной графике для создания таких визуальных эффектов, как дым, облака, туман, огонь и т. д. Он также очень часто используется как простая текстура, покрывающая геометрическую модель. В отличие от растровых текстур, шум Перлина является процедурной текстурой, и поэтому он не занимает память, но, вместе с тем, исполнение алгоритма требует неких вычислительных ресурсов.

## **Обоснование выбранной среды разработки**

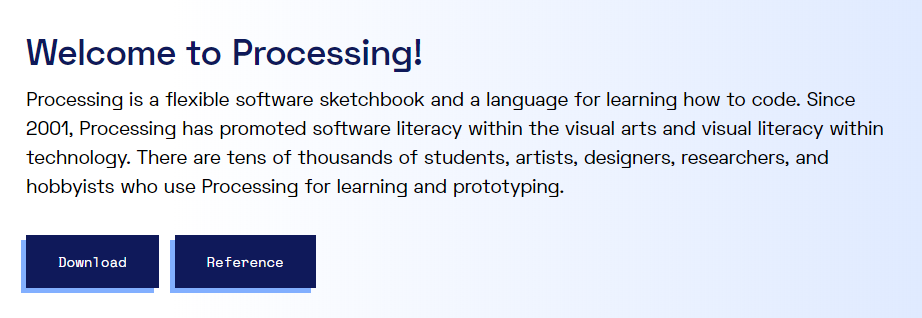
Мной были выбраны языки Java, Processing.

**Processing** — открытый язык программирования, основанный на Java. Представляет собой лёгкий и быстрый инструментарий для создания изображений, анимации и разработки интерфейсов.

**Java** представляет собой язык программирования и платформу вычислений, которая была впервые выпущена компанией Sun Microsystems в 1995 г. Технология эволюционировала из скромной разработки до инструмента, который играет серьезную роль в современном цифровом мире, предоставляя надежную платформу для множества сервисов и приложений. Инновационные продукты и цифровые услуги, разрабатываемые для будущего, также будут создаваться на основе Java.

**Краткая инструкция по установке среды разработки**

1. Установить **Processing** с официального [сайта](https://processing.org/).



1. Запустить скачанный установщик, и оставив все опции установщика стандартными, завершить установку.

## **Постановка (формулировка) заданий**

Главной моей задачей было создать несколько графических изображений, путем написания их через Processing с использование языка Java

Мне нужно написать программу, которая будет генерировать изображение, опираясь на понятие – шум Перлин.

1. Написать код для визуализации шумов Перлина.
2. Написать код для переключения графических изображений.
3. Проверить финальную модель на работоспособность и сдать работу.

В итоге я получил готовый код, который содержит в себе интерфейс приложений и несколько графически изображений.

## **Коды программ с комментариями**

Для начала рассмотрим Java код:

int selectedOption = 0;

boolean drawingCompleted = false;

void setup() {

size(400, 400);

background(255);

drawButtons();

}

void draw() {

if (drawingCompleted) {

drawReturnButton();

}

}

void drawButtons() {

fill(200);

rect(50, 50, 160, 40);

rect(220, 50, 160, 40);

fill(0);

textAlign(CENTER, CENTER);

textSize(15);

text("Нарисовать дом", 135, 70);

text("Нарисовать машину", 300, 70);

}

void drawReturnButton() {

fill(200);

rect(300, 330, 80, 40);

fill(0);

textAlign(CENTER, CENTER);

textSize(16);

text("Вернуться", 340, 350);

}

void mousePressed() {

if (!drawingCompleted) {

if (mouseX > 50 && mouseX < 150 && mouseY > 50 && mouseY < 90) {

selectedOption = 1;

clearCanvas();

drawHouse();

drawingCompleted = true;

} else if (mouseX > 200 && mouseX < 300 && mouseY > 50 && mouseY < 90) {

selectedOption = 2;

clearCanvas();

drawSomethingElse();

drawingCompleted = true;

}

} else {

if (mouseX > 300 && mouseX < 380 && mouseY > 330 && mouseY < 370) {

drawingCompleted = false;

clearCanvas();

drawButtons();

}

}

}

void clearCanvas() {

background(255);

}

void drawHouse() {

fill(255, 0, 0);

rect(100, 200, 200, 150);

fill(100);// Труба

rect(250, 120, 20, 50);

fill(0, 0, 255);

triangle(100, 200, 200, 100, 300, 200);

fill(255);

rect(120, 220, 60, 60);

fill(120, 80, 19);

rect(200, 220, 40, 130);

}

void drawSomethingElse() {

fill(0, 150, 255);

rect(50, 100, 300, 80, 10);

fill(173, 216, 230);

rect(260, 110, 70, 30);

fill(0);

ellipse(120, 180, 40, 40);

ellipse(320, 180, 40, 40);

fill(0, 150, 255);

rect(190, 100, 60, 80);

fill(255);

ellipse(280, 160, 10, 10);

}

Выше представлен код, с помощью которого мы можем нарисовать два графических изображения, Автомобиль и Дом, а также интерфейс приложения.

А ниже представлен код, с помощью которого мы можем нарисовать два изображения, но в этот раз 3D-изображения, между которыми мы можем переключаться с помощью клавиш 1 и 2.

|  |
| --- |
| int currentDrawing = 1;  void setup() {  size(500, 500, P3D);  }  void draw() {  background(255);  if (currentDrawing == 1) {  drawDrawing1();  } else if (currentDrawing == 2) {  drawDrawing2();  }  }  void drawDrawing1() {  float changerate = 0.005;  float timeslice = 0.005;  float noiseScale = 0.005;  translate(width/2, height/2);  background(150);  rotateX((mouseY - height/2)\*2\*PI/width);  rotateZ((mouseX - width/2)\*2\*PI/height);  for (int i = -250; i < 250; i+=10) {  for (int j = -250; j < 250; j+=10) {  point(i, j, noise(i \* noiseScale, j \* noiseScale, timeslice) \* 200 - 50);  }  }  timeslice += changerate;  }  float[] x = new float[0];  float[] y = new float[0];  float[] xSpeed = new float[0];  float[] ySpeed = new float[0];  float[] size = new float[0];  float[] r = new float[0];  float[] g = new float[0];  float[] b = new float[0];  void drawDrawing2() {  size(500, 500);  background(200);  for (int i = 0; i < x.length; i++) {  x[i] += xSpeed[i];  if (x[i] < 0 || x[i] > width) {  xSpeed[i] \*= -1;  }  y[i] += ySpeed[i];  if (y[i] < 0 || y[i] > height) {  ySpeed[i] \*= -1;  }  fill(r[i], g[i], b[i]);  ellipse(x[i], y[i], size[i], size[i]);  }  }  void mousePressed() {  x = append(x, mouseX);  y = append(y, mouseY);  xSpeed = append(xSpeed, random(-5, 5));  ySpeed = append(ySpeed, random(-5, 5));  size = append(size, random(5, 20));  r = append(r, random(256));  g = append(g, random(256));  b = append(b, random(256));  }  void keyPressed() {  if (key == '1') {  currentDrawing = 1;  } else if (key == '2') {  currentDrawing = 2;  }  } |

## **Заключение**

По завершению данной практической работы, были нарисованы 4 графических изображения с помощью языка Java, с использованием инструмента Processin