# **Язык программирования Processing**



## Общая характеристика и история создания

«Processing — открытый язык программирования, основанный на Java. Представляет собой лёгкий и быстрый инструментарий для людей, которые хотят программировать изображения, анимацию и интерфейсы. Используется студентами, художниками, дизайнерами, исследователями и любителями, для изучения, прототипирования и производства. Он создан для изучения основ компьютерного программирования в визуальном контексте и служит альбомным программным обеспечением (имеется в виду то, что каждый \*.pde файл визуальной оболочки Processing’а представляет собой отдельное изображение или анимацию, и т. д.) и профессиональным производственным инструментом. Processing — это открытый проект инициированный Бенжамином Фраем и Кэйси Ризом. Он родился из идей, изученных в The Aesthetics and Computation Group в MIT Media Lab.» [1]

## Система типов и особенности архитектуры языка

«Переменные являются символьными именами для областей памяти компьютера, используемыми для хранения информации в программе. Есть восемь базовых типов данных в Java, и все они поддерживаются в Processing: byte, short, int, long, float, double, boolean и char.(Существующие типы данных см. в Таблица 1)» [2]

Таблица 1

|  |  |
| --- | --- |
| boolean | Тип данных для логических значений true и false |
| byte | Тип данных byte, 8 бит информации, хранящий числовые значения от -128 до 127 |
| char | Переменная типа char хранит буквы и символы в формате [Unicode](http://wikihandbk.com/ruwiki/index.php?title=Unicode&action=edit&redlink=1" \o "Unicode (страница не существует)), системе кодирования, разработанной для поддержки различных мировых языков. Каждая переменная типа char имеет длину 2 байта (16 бит) |
| color | Тип данных для хранения значений цвета. Цвета могут быть установлены с помощью get() и color() или указаны непосредственно в шестнадцатеричной нотации, например **#FFCC00** или **0xFFFFCCOO**. |
| double | Тип данных для чисел с плавающей точкой. Она обладает большим диапазоном для хранения чисел и занимает в памяти 64 бита |
| float | Тип данных для чисел с плавающей точкой. Диапазон 32 бита. |
| int | Тип данных для целых чисел без десятичной запятой в диапазоне от -2,147,483,648 до 2,147,483,647 |
| long | Тип данных для больших целых чисел. Длинное целое (long) число может принимать значения в диапазоне от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807 |

[3]

## Примеры кода и интересные факты

«Карта президентских выборов США с результатом работы:» [4]

PShape usa;

PShape state;

String [] Obama = { "HI", "RI", "CT", "MA", "ME", "NH", "VT", "NY", "NJ",

"FL", "NC", "OH", "IN", "IA", "CO", "NV", "PA", "DE", "MD", "MI",

"WA", "CA", "OR", "IL", "MN", "WI", "DC", "NM", "VA" };

String [] McCain = { "AK", "GA", "AL", "TN", "WV", "KY", "SC", "WY", "MT",

"ID", "TX", "AZ", "UT", "ND", "SD", "NE", "MS", "MO", "AR", "OK",

"KS", "LA" };

void setup() {

size(950, 600);

*// The file Blank\_US\_Map.svg can be found at Wikimedia Commons*

usa = loadShape("http://upload.wikimedia.org/wikipedia/commons/3/32/Blank\_US\_Map.svg");

smooth(); *// Improves the drawing quality of the SVG*

noLoop();

}

void draw() {

background(255);

*// Draw the full map*

shape(usa, 0, 0);

*// Blue denotes states won by Obama*

statesColoring(Obama , color(0, 0, 255));

*// Red denotes states won by McCain*

statesColoring(McCain, color(255, 0, 0));

*// Save the map as image*

saveFrame("map output.png");

}

void statesColoring(String[] states, int c){

**for** (int i = 0; i < states.length; ++i) {

PShape state = usa.getChild(states[i]);

*// Disable the colors found in the SVG file*

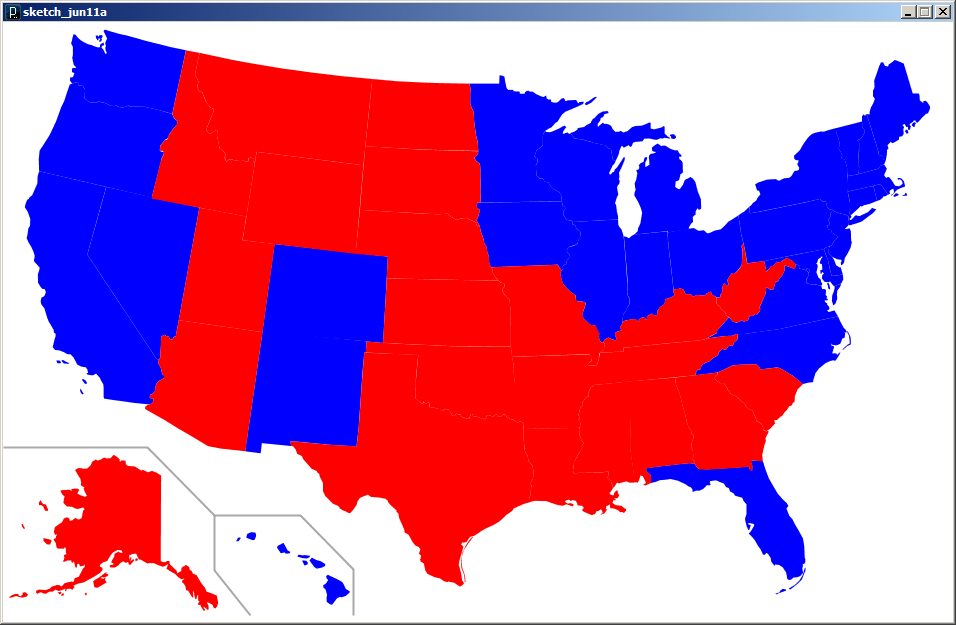
state.disableStyle();

*// Set our own coloring*

fill(c);

noStroke();

*// Draw a single state*

shape(state, 0, 0);}

# **Язык программирования Fantom**



## Общая характеристика и история создания

«Fantom объектно ориентированный язык программирования общего назначения созданный Брайаном и Энди Фрэнком (Brian Frank, Andy Frank) который компилируется в Java Runtime Environment (JRE), JavaScript, и .NET Common Language Runtime (CLR). Главной целью является предоставление стандартного API который абстрагирует от вопроса в каком окружении будет выполняться код: в JRE или CLR. Язык поддерживает функциональное программирование через замыкания и многопоточность через акторы.» [5]

## Система типов и особенности архитектуры языка

«Весь мир раскололся на сторонников статической и динамической типизации. Создатели Fantom считают, что обе стороны крайне критично относятся друг к другу и выбирают середину между ними — умеренный подход к системе типизации. Со стороны статической типизации, Fantom требует описание полей и сигнатур методов с указанием типов. Это хорошая практика, зафиксировать формат общения между компонентами. К примеру, если я хочу написать метод, который работает со строкой и числом, то это должно быть зафиксировано прямо в коде. В отличие от статической типизации сигнатур методов и полей, в коде она часто только мешает, заставляя писать ненужный код. Вывод типов в Fantom позволяет избежать этой проблемы. Но иногда вам действительно нужна динамическая типизация, поэтому одной из ключевых особенностей Fantom является возможность вызвать метод, используя статическую или динамическую типизацию. Если вы вызываете метод с помощью оператора ".", вызов будет проверен компилятором и скомпилирован в эффективный машинный код. С другой стороны, вы можете использовать оператор "->", для указания динамического вызова. На самом деле, "->" будет перенаправлен на вызов Obj.trap. По умолчанию trap работает как ".", но только во время выполнения. Вы можете изменить это поведение, определив свой динамический дизайн.» [6]