**UNIVERSITATEA TEHNICĂ A MOLDOVEI**

**FACULTATEA CALCULATOARE, INFORMATICA, MICROELECTRONICA**

# GRAFICA PE CALCULATOR

## ÎNDRUMĂR METODIC PENTRU LUCRĂRI DE LABORATOR
**(draft)**

**2022**

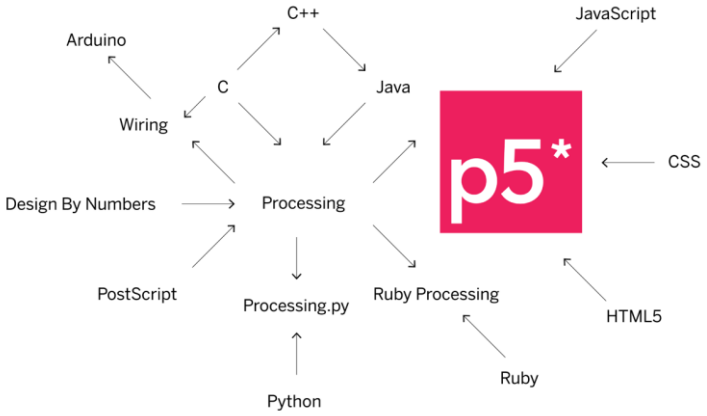
**Chişinău 2022**

## Введение в графическую библиотеку p5.js

### Что такое p5.js?

P5.js — это библиотека, написанная на языке программирования JavaScript, используемая для создания и просмотра интерактивных изображений с использованием простых графических примитивов. P5.js позволяет создавать компьютерную графику с помощью языка программирования. Это позволяет легко интегрировать письменный код в веб-страницы путем добавления письменного кода в HTML-документ.

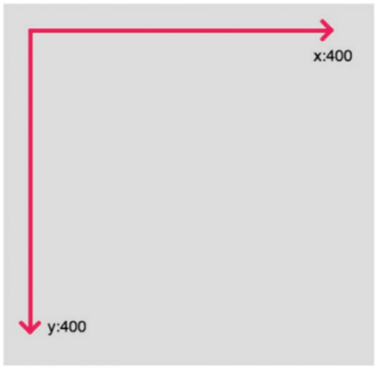
P5.js является бесплатным, и независимым от платформы, поэтому приложения могут работать в любой операционной системе, также p5.js имеет большое семейство языков программирования.




**Фигура 1. Языки программирования связанные с p5.js**

### Как начать писать код в P5.js

Перед тем, как приступить к созданию собственных программ, необходимо знать, что любая фигура, созданная в среде p5.js, связана с системой координат, начало системы координат в любой программе — это крайний левый угол экрана. Вертикальная ось называется осью Y, а горизонтальная ось называется осью X. Увеличение значений координат x и y показано на рисунке 2.

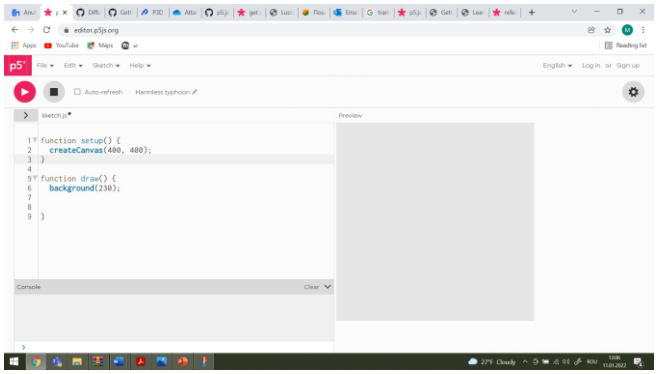



**Фигура 2. Система координат в p5.js**

Для создания простой программы на p5.js исспользуем следующий шаблон:

```
function setup(){
    createCanvas(400,400);
}
function draw(){
    background(220);}
```

Результат работы программы и пример онлайн-редактора p5.js показан на рисунке 3.



**Фигура 3. Рабочее окно в среде p5.js**

**Функция setup ()** вызывается один раз при запуске программы. Она используется для определения начальных свойств среды, таких как размер экрана и цвет фона, а также для загрузки мультимедийных файлов, таких как изображения и шрифты, при запуске программы. Для каждой программы может быть только одна функция setup (), и ее не следует вызывать после первоначального выполнения.

Примечание. Переменные, объявленные в setup (), не доступны в других функциях, включая draw ().

```
function setup() {
createCanvas();
}
```

Создает элемент canvas в документе и устанавливает его размеры в пикселях. Этот метод следует вызывать только один раз в начале установки. Вызов createCanvas более одного раза в эскизе приведет к очень непредсказуемому поведению. Если вам нужно более одного холста для рисования, вы можете использовать createGraphics (по умолчанию скрыто, но может отображаться).

Ширина и высота системных переменных задаются параметрами, передаваемыми этой функции. Если createCanvas () не используется, окну будет присвоен размер по умолчанию 100x100 пикселей.

```
createCanvas(w, h, [renderer])
```

w число: ширина холста

h число: высота холста

Константа renderer: P2D или WEBGL (необязательно, WEBGL для 3D графики)

**Функция draw ()** вызванный непосредственно после setup (), функция draw () непрерывно выполняет строки кода, содержащиеся в своем блоке, до тех пор, пока программа не будет остановлена или не будет вызвана noLoop (). Обратите внимание, что если в setup () вызывается noLoop (), draw () все равно будет выполняться один раз перед остановкой. draw () вызывается автоматически и никогда не должен вызываться явно.

Он всегда должен управляться с помощью noLoop (), redraw () и loop (). После того, как noLoop () останавливает выполнение кода в draw (), redraw () приводит к тому, что код внутри draw () выполняется один раз, а loop () заставляет код внутри draw () возобновлять непрерывное выполнение.

Количество выполнений draw () в каждую секунду можно контролировать с помощью функции frameRate ().

Может быть только одна функция draw () для каждого эскиза, и draw () должна существовать, если вы хотите, чтобы код выполнялся непрерывно или обрабатывал события, такие как mousePressed (). Иногда в вашей программе может быть пустой вызов метода draw (), как показано в приведенном выше примере.

Важно отметить, что система координат рисования будет сбрасываться в начале каждого вызова draw (). Если какие-либо преобразования выполняются в draw () (например: масштабирование, поворот, перевод), их эффекты будут отменены в начале draw (), поэтому преобразования не будут накапливаться с течением времени.

```
function draw() {
---------
}
```

**Функция background ()** устанавливает цвет, используемый для фона холста. Фон по умолчанию прозрачный. Эта функция обычно используется в draw () для очистки окна отображения в начале каждого кадра, но ее можно использовать внутри setup (), чтобы установить фон для первого кадра анимации или если фон нужно установить только один раз.

Цвет указывается с точки зрения цвета RGB, HSB или HSL, в зависимости от текущего colorMode. (Цветовое пространство по умолчанию - RGB, каждое значение находится в диапазоне от 0 до 255). Диапазон альфа по умолчанию также составляет от 0 до 255.

Если указан один строковый аргумент, поддерживаются цветовые строки RGB, RGBA и Hex CSS и все именованные цветовые строки. В этом случае значение альфа-числа в качестве второго аргумента не поддерживается, следует использовать форму RGBA.

Объект Color также может быть предоставлен для установки цвета фона.

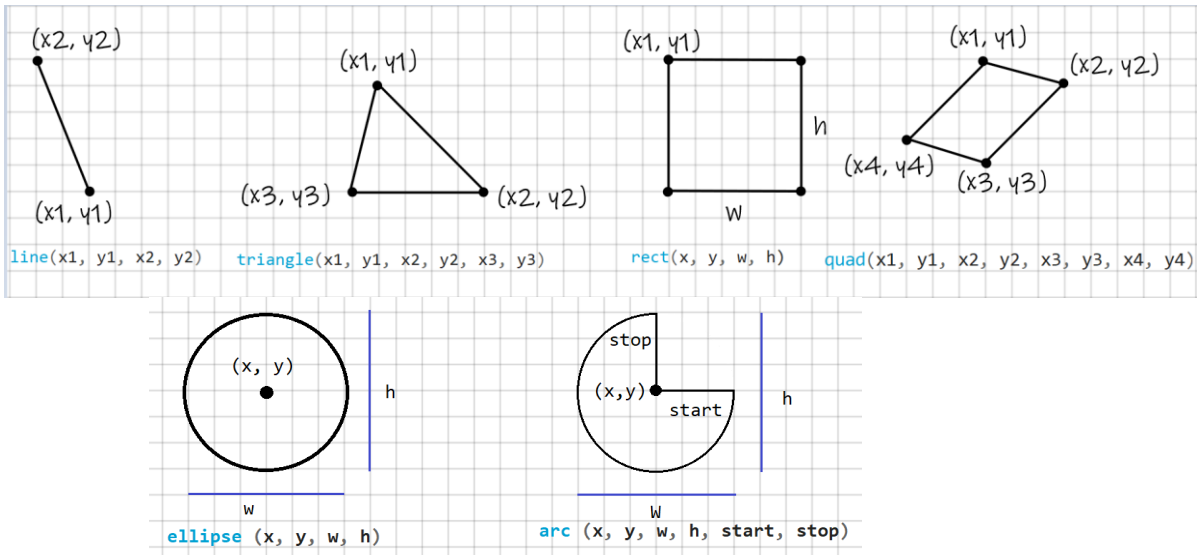
Изображение Image также может быть предоставлено для установки фонового изображения.

```
background(color)
background(colorstring, [a])
background(gray, [a])
background(v1, v2, v3, [a])
background(values)
background(image, [a])
```

| | |
|---|---|
| color: | любое значение, созданное функцией color () |
| colorstring String: | строка цвета, возможные форматы: целое число rgb () или rgba (), процентное отношение rgb () или rgba (), 3-значный шестнадцатеричный, 6-значный шестнадцатеричный |
| a число: | непрозрачность фона относительно текущего цветового диапазона (по умолчанию 0-255) (необязательно) |
| gray число: | указывает значение между белым и черным |
| v1 число: | красный или значение оттенка (в зависимости от текущего цветового режима) |
| v2 число: | значение зеленого или насыщенности (в зависимости от текущего цветового режима) |
| v3 число: | синий или значение яркости (в зависимости от текущего цветового режима) |
| values Number []: | массив, содержащий красный, зеленый, синий и альфа-компоненты цвета |
| image: | изображение, созданное с помощью loadImage () или createImage (), для установки в качестве фона (должно быть того же размера, что и окно эскиза) |

# Глава I
## 1.1 Создание простых 2D графических примитив

Простые графические примитивы — это геометрические фигуры, которые можно создавать с помощью функций в графической библиотеке P5.js. Простейшими графическими примитивами являются двумерные графические примитивы, на рис. 1.1 показано соответствие точек геометрических фигур и параметров, которые должны быть указаны в качестве аргументов функции в коде программы.




**Фигура 1.1. Связь между геометрическими точками и параметрами функции P5.js**

**Функция arc ():** рисует дугу на экране. Если вызывается только с x, y, w, h, start и stop, дуга будет нарисована и заполнена как открытый круговой сегмент. Если указан параметр режима, дуга будет заполнена как открытый полукруг (OPEN), замкнутый полукруг (CHORD) или как замкнутый круговой сегмент (PIE). Источник может быть изменен с помощью функции ellipseMode (). Разница между режимами рисования показана на рисунке 1.3.

Дуга всегда рисуется по часовой стрелке от того места, где начинается падение, до того места, где остановка падает на эллипсе. Добавление или вычитание TWO_PI для любого угла не меняет места их падения. Если и старт, и остановка падают в одном и том же месте, будет нарисован полный эллипс. Имейте в виду, что ось Y увеличивается в направлении вниз, поэтому углы измеряются по часовой стрелке от положительного направления X («3 часа»).

Начальная точка и конечная точка могут использовать константы p5.js в соответствии со значениями, указанными в окружности, показанной на рисунке 1.2.

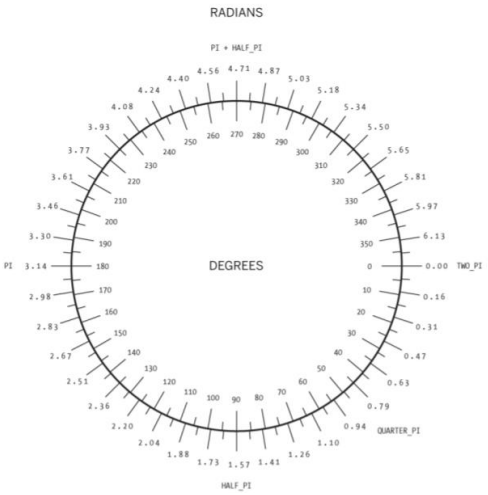
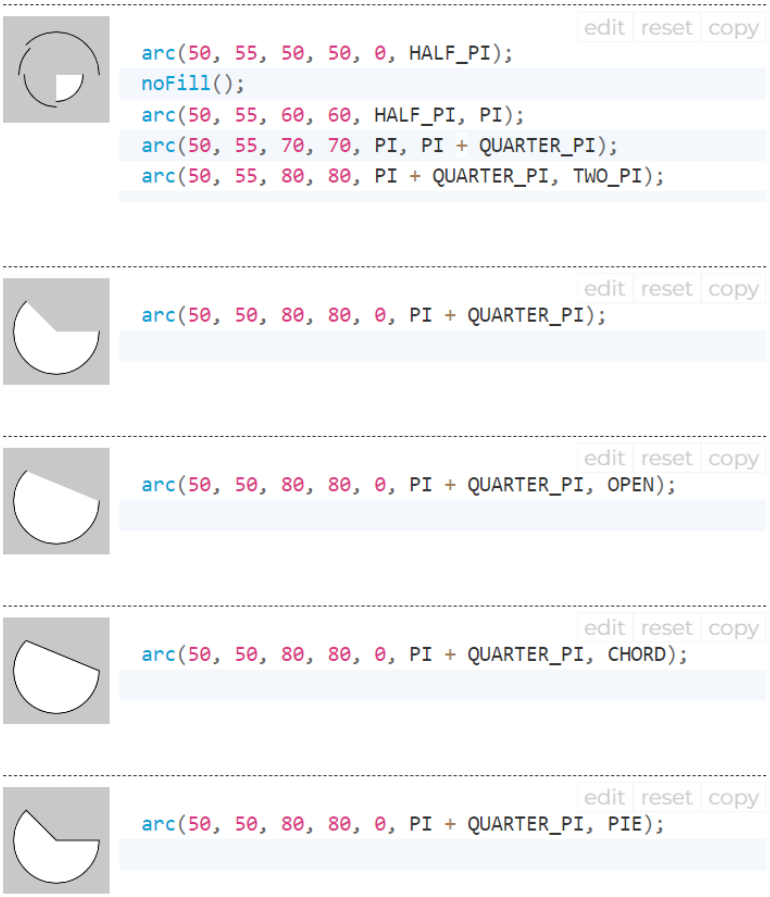



**Figura 1.2. Relația dintre constante, radiani și grade în p5.js**

```
arc (x, y, w, h, start, stop, [mode], [detail])
```

x число: x-координата эллипса дуги
y число: y-координата эллипса дуги
w число: ширина эллипса дуги по умолчанию
h число: высота эллипса дуги по умолчанию
start число: угол начала дуги, указанный в радианах
stop число: угол остановки дуги, указанный в радианах
mode константа: параметр для определения способа рисования дуги.
CHORD, PIE или OPEN (необязательно)




**Фигура 1.2. Примеры рисования дуги**

**Функция ellipse():** рисует эллипс (овал) на экране. Эллипс с равной шириной и высотой - это круг. По умолчанию первые два параметра задают местоположение, а третий и четвертый параметры определяют ширину и высоту фигуры. Если высота не указана, значение ширины используется как для ширины, так и для высоты. Если указана отрицательная высота или ширина, берется абсолютное значение. Источник может быть изменен с помощью функции ellipseMode ().

**Синтаксис**

```
ellipse(x, y, w, [h])
ellipse (x, y, w, h, detail)
```

параметры

x Число: x-координата эллипса.

y Число: y-координата эллипса.

w Число: ширина эллипса.

h Число: высота эллипса. (Необязательный)

detail Целое число: количество радиальных секторов для рисования (для режима WebGL)

**Функция circle():** рисует круг на экране. Эта функция является частным случаем функции ellipse (), где ширина и высота эллипса одинаковы. Высота и ширина эллипса соответствуют диаметру круга. По

умолчанию первые два параметра задают расположение центра круга, третий - диаметр круга.

**Синтаксис**

```
circle(x, y, d)
```

Параметры

x Число: x-координата центра круга.

y Число: y-координата центра круга.

d Число: диаметр круга.

На рис. 1.3 показан пример функций ellipse и circle.

```
ellipse(56, 46, 55, 55);
```

```
// Draw a circle at location (30, 30) with a
diameter of 20.
circle(30, 30, 20);
```

**Фигура 1.3. Примеры функций circle и ellips**

**Функция line():** рисует линию (прямой путь между двумя точками) к экрану. Версия line () с четырьмя параметрами рисует линию в 2D. Чтобы закрасить линию, используйте функцию stroke(). Строка не может быть заполнена, поэтому функция fill () не повлияет на цвет строки. 2D линии по умолчанию рисуются с шириной в один пиксель, но это можно изменить с помощью функции strokeWeight ().

**Синтаксис**

```
line(x1, y1, x2, y2)
line(x1, y1, z1, x2, y2, z2)
```

Параметры

x1: x-координата первой точки

y1: y-координата первой точки

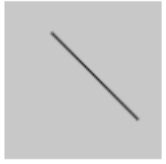
x2: x-координата второй точки

y2: y-координата второй точки

z1: координата z первой точки

z2: координата z второй точки

Пример:



```
line(30, 20, 85, 75);
```

**Функция point():** рисует точку, координату в пространстве размером один пиксель. Первый параметр - это горизонтальное значение для точки, второй - вертикальное значение для точки. Цвет точки изменяется с помощью функции stroke(). Размер точки изменяется с помощью функции strokeWeight().

**Синтаксис**

```
point(x, y, [z])
point(coordinate_vector)
```

Параметры

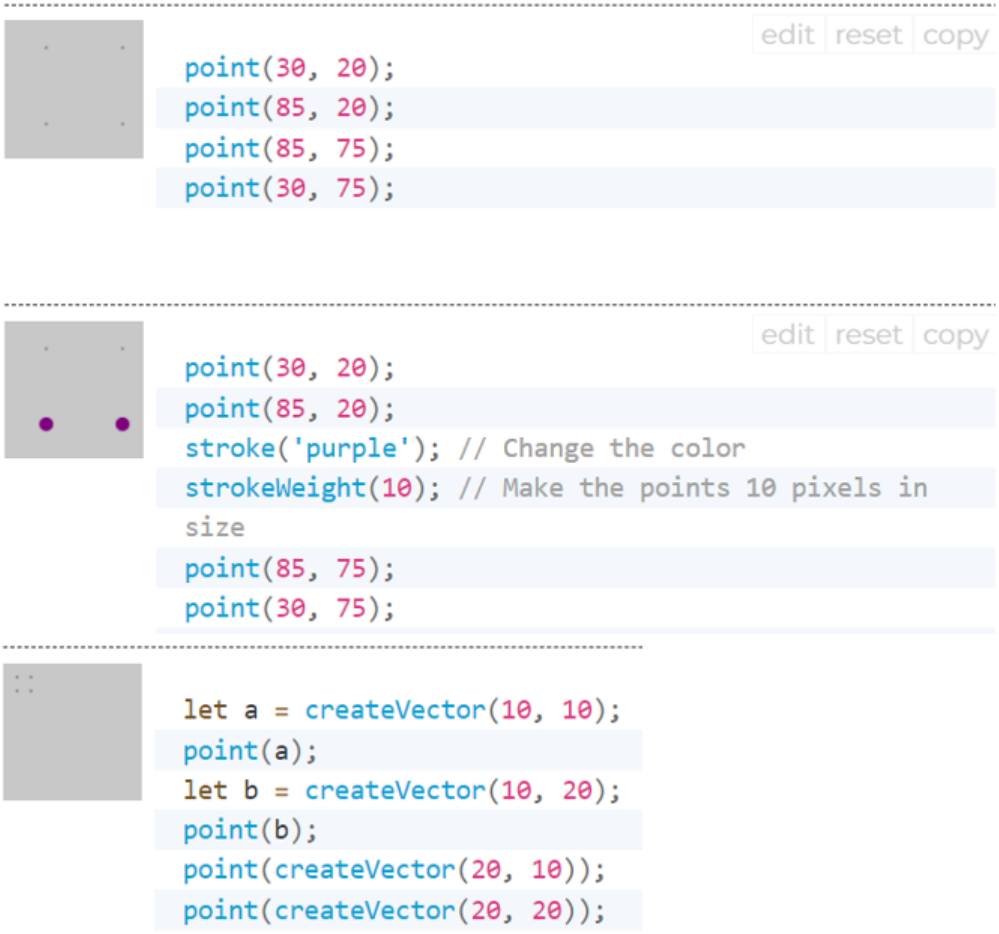
x: x-координата

y: y-координата

z: координата z (для режима WebGL) (необязательно)

coordonate_vector: вектор координат;

Примеры программы функции point показаны на рис. 1.4.




**Фигура 1.4. Примеры программы функции point**

**Функция quad():** рисует четырехугольник. Четырехугольник - четырехсторонний многоугольник. Он похож на прямоугольник, но углы между его краями не ограничены до девяноста градусов. Первая пара параметров (x1, y1) задает первую вершину, а последующие пары должны двигаться по часовой стрелке или против часовой стрелки вокруг

определенной формы. z-аргументы работают только тогда, когда quad () используется в режиме WEBGL.

**Синтаксис**

```
quad(x1, y1, x2, y2, x3, y3, x4, y4)
quad(x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4)
```

Параметры

x1: x-координата первой точки
y1: y-координата первой точки
x2: x-координата второй точки
y2: y-координата второй точки
x3: x-координата третьей точки
y3: y-координата третьей точки
x4: x-координата четвертой точки
y4: y-координата четвертой точки
z1: координата z первой точки
z2: координата z второй точки
z3: координата z третьей точки
z4: координата z четвертой точки

**Функция rect (): ** рисует прямоугольник на экране. Прямоугольник - это четырехсторонняя фигура с каждым углом в девяносто градусов. По умолчанию первые два параметра задают расположение верхнего левого угла, третий - ширину, а четвертый - высоту. Однако способ интерпретации этих параметров можно изменить с помощью функции rectMode ().

Пятый, шестой, седьмой и восьмой параметры, если указаны, определяют радиус угла для верхнего левого, верхнего правого, нижнего правого и нижнего левого углов соответственно. Опущенный параметр радиуса угла устанавливается равным значению ранее указанного значения радиуса в списке параметров.

**Синтаксис**

```
rect(x, y, w, h, [tl], [tr], [br], [bl])
rect(x, y, w, h, [detailX], [detailY])
```

Параметры

x: x-координата прямоугольника.
y: y-координата прямоугольника.
w r: ширина прямоугольника.
h: высота прямоугольника.
tl: радиус верхнего левого угла. (Необязательный)
tr: радиус верхнего правого угла. (Необязательный)
br: радиус нижнего правого угла. (Необязательный)
bl: радиус нижнего левого угла. (Необязательный)

detailX Integer: количество сегментов в направлении x (для режима WebGL) (необязательно)

detailY Integer: количество сегментов в направлении y (для режима WebGL) (необязательно)

Примеры программы и их результаты показаны на рис. 1.5.

```
quad(38, 31, 86, 20, 69, 63, 30, 76);
```

```
// Draw a rectangle at location (30, 20) with a
width and height of 55.
rect(30, 20, 55, 55);
```

```
// Draw a rectangle with rounded corners, each
having a radius of 20.
rect(30, 20, 55, 55, 20);
```

```
// Draw a rectangle with rounded corners having the
following radii:
// top-left = 20, top-right = 15, bottom-right =
10, bottom-left = 5.
rect(30, 20, 55, 55, 20, 15, 10, 5);
```

**Фигура 1.5. Примеры программ функций rect și quad**

**Функция square():** рисует квадрат на экране. Квадрат - это четырехсторонняя форма с каждым углом в девяносто градусов и равным размером стороны. Эта функция является частным случаем функции rect (), где ширина и высота одинаковы, а параметр называется «s» для размера стороны. По умолчанию первые два параметра задают расположение верхнего левого угла, третий - размер стороны квадрата. Однако способ интерпретации этих параметров можно изменить с помощью функции rectMode ().

Четвертый, пятый, шестой и седьмой параметры, если они указаны, определяют радиус угла для левого верхнего, правого верхнего, правого нижнего и левого нижнего углов соответственно. Опущенный параметр радиуса угла устанавливается равным значению ранее указанного значения радиуса в списке параметров.

**Синтаксис**

```
square(x, y, s, [tl], [tr], [br], [bl])
```

Параметры

x: x-координата квадрата.

y: y-координата квадрата.

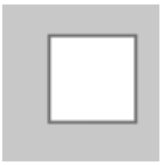
s: размер стороны квадрата.

tl: радиус верхнего левого угла. (Необязательный)

tr: радиус верхнего правого угла. (Необязательный)

br: радиус нижнего правого угла. (Необязательный)

bl: радиус нижнего левого угла. (Необязательный)



```
// Draw a square at location (30, 20) with a side
size of 55.
square(30, 20, 55);
```

```
// Draw a square with rounded corners, each having
a radius of 20.
square(30, 20, 55, 20);
```

```
// Draw a square with rounded corners having the
following radii:
// top-left = 20, top-right = 15, bottom-right =
10, bottom-left = 5.
square(30, 20, 55, 20, 15, 10, 5);
```

**Фигура 1.6. Примеры программ функции square**

**Функция triangle():** Треугольник - это плоскость, созданная путем соединения трех точек. Первые два аргумента указывают первую точку, средние два аргумента указывают вторую точку, а последние два аргумента указывают третью точку.

**Синтаксис**

```
triangle(x1, y1, x2, y2, x3, y3)
```

Параметры

x1: x-координата первой точки
y1: y-координата первой точки
x2: x-координата второй точки
y2: y-координата второй точки
x3: x-координата третьей точки
y3: y-координата третьей точки

```
triangle(30, 75, 58, 20, 86, 75);
```

**Фигура 1.7. Примеры программ функции triangle**

## 1.2 Простые графические атрибуты

Особенности геометрических фигур, такие как цвет фигуры, толщина и цвет линии, тип штриховки, способ соединения линий, являются простыми графическими атрибутами. В библиотеке p5.js есть список функций, позволяющих устанавливать или изменять эти атрибуты.

Основные графические атрибуты описаны ниже:

**Функция fill():** Устанавливает цвет, используемый для заливки фигур. Например, если вы запустите заливку (204, 102, 0), все фигуры, нарисованные после команды заливки, будут заполнены оранжевым цветом. Этот цвет указывается с точки зрения цвета RGB или HSB в зависимости от текущего colorMode (). (Цветовое пространство по умолчанию - RGB, каждое значение находится в диапазоне от 0 до 255). Диапазон альфа по умолчанию также составляет от 0 до 255.

Если указан один строковый аргумент, поддерживаются цветовые строки RGB, RGBA и Hex CSS и все именованные цветовые строки. В этом случае значение альфа-числа в качестве второго аргумента не поддерживается, следует использовать форму RGBA.

```
fill(v1, v2, v3, [alpha])
fill(value)
fill(gray, [alpha])
fill(values)
fill(color)
```

v1 число: красный или значение оттенка относительно текущего цветового диапазона

v2 число: значение зеленого или насыщенности относительно текущего цветового диапазона

v3 число: синий или значение яркости относительно текущего цветового диапазона

альфа число: (необязательно)

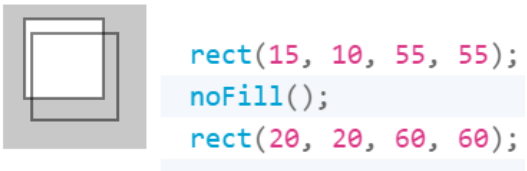
String: string цветная строка

серый номер: значение серого

значения Number []: массив, содержащий красный, зеленый, синий и альфа-компоненты цвета


**Фигура 1.8.а) Примеры использования функции fill**

Чтобы фигура была прозрачной, используем функцию **nofill**.




**Фигура 1.8.б) Примеры использования функции nofill**

**Функция stroke ():**

Задает цвет, используемый для рисования линий и границ фигур. Этот цвет указывается в терминах цвета RGB или HSB в зависимости от текущего colorMode () (цветовое пространство по умолчанию - RGB, каждое значение находится в диапазоне от 0 до 255). Альфа-диапазон по умолчанию также составляет от 0 до 255. Если указан единственный строковый аргумент, поддерживаются цветовые строки RGB, RGBA и Hex CSS и все именованные цветовые строки. В этом случае значение альфа-

числа в качестве второго аргумента не поддерживается, следует использовать форму RGBA.

**Синтаксис:**

```
stroke(v1, v2, v3, [alpha])
stroke(value)
stroke(gray, [alpha])
stroke(values)
stroke(color)
```

Параметры

v1 - значение красного или оттенка относительно текущего цветового диапазона

v2 - зеленый цвет или значение насыщенности относительно текущего цветового диапазона

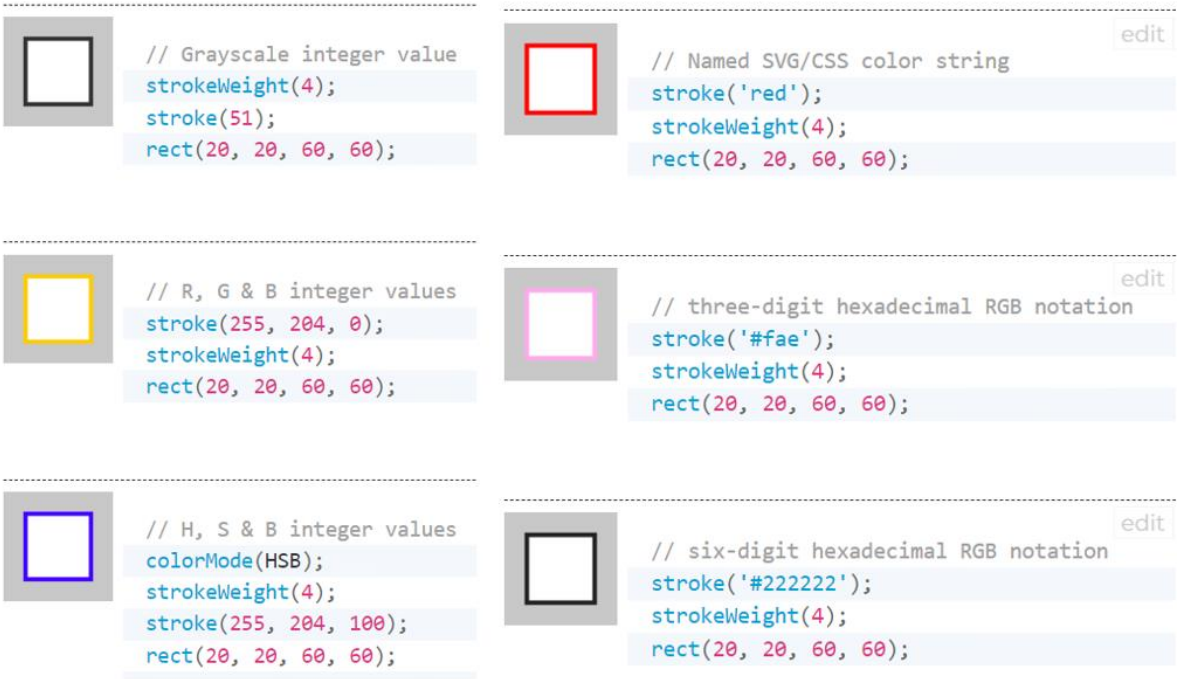
v3 - синий или значение яркости относительно текущего цветового диапазона

alpha -число: (необязательно)

value - цветовая строка

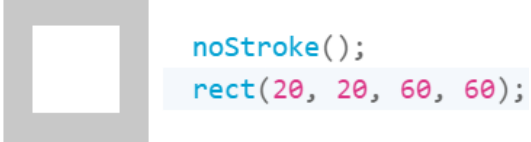
grey – число, значение серого

values число []: массив, содержащий красный, зеленый, синий и альфа-компоненты цвета




**Фигура 1.9.а) Примеры использования функции stroke**

Чтобы фигура рисовалась без рамки, используем функцию noStroke.




**Фигура 1.9.б) Примеры использования функции noStroke**

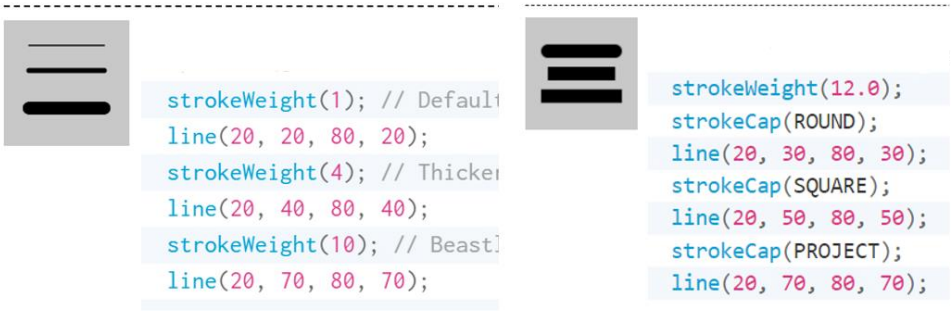
Помимо цвета, функция stroke может указывать следующие параметры:

strokeCap()
strokeJoin()
strokeWeight()

**Функция strokeWeight(number):** Устанавливает ширину линии. Все значения указаны в пикселях.

**Функция strokeCap(cap):** Устанавливает стиль воспроизведения конца строки. Эти концы либо закруглены, либо квадратны, либо расширены, каждый из которых указан с соответствующими параметрами: ROUND, SQUARE şi PROIECT.

Примеры использования этих аргументов показаны на рис. 1. 10.




**Фигура 1.10. Примеры использования функции stroke**

**Функция strokeJoin(join):** Задает стиль соединения (стыковки) отрезков. Они могут быть указаны с соответствующими параметрами MITRE, BEVEL и ROUND. По умолчанию установлено значение MITRE.

Разница между этими режимами показана на рисунке 1.11.


**Фигура 1.11. Примеры использования функции strokeJoin**

Если необходимо добавить текст, можно использовать текстовую функцию, которая может иметь разные параметры.

**Функция text():** Рисует текст на экране, выводит информацию, указанную в первом параметре, на экран в позиции, заданной дополнительными параметрами. Будет использоваться шрифт по

умолчанию, если шрифт не установлен с помощью функции textFont(), и размер по умолчанию будет использоваться, если шрифт с textSize() не установлен. Цвет текста можно изменить с помощью функции fill(). Изменение контура текста осуществляется с помощью функций stroke() и strokeWeight().

Текст отображается в соответствии с функцией textAlign(), которая предоставляет возможность рисовать левые, правые и центральные координаты.

**Синтаксис :**

text(str, x, y, [x2], [y2])

Параметры:

str : указывает строку для отображения на экране;

x: координата x начальной точки текста;

y: координата y начальной точки текста;

x2 и y2: определяют прямоугольную область для отображения и могут использоваться только со строками. Когда эти параметры указаны, они интерпретируются на основе текущей настройки rectMode(). Текст, который не полностью помещается в указанный прямоугольник, не будет отображаться на экране. Если x2 и y2 не указаны, базовое выравнивание используется по умолчанию, что означает, что текст будет составлен из x и y.

Основные атрибуты текста приведены на рисунок. 1.12


**Фигура 1.12. Примеры использования функций работы с текстом**

**Библиографические источники:**


- https://p5js.org/reference/
- https://github.com/processing/p5.js/wiki/p5.js-overview
- ”Make: Getting started with p5.js” Lauren McCarthy, Casey Reas, Ben Fry;
- ”Learn JavaScript with p5.js” Engin Arslan

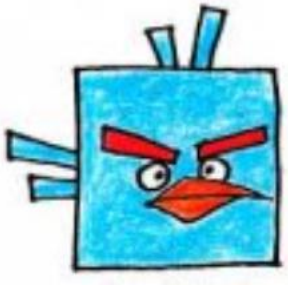
## Lucrarea de laborator 1
## Tema: Studierea primitivelor grafice simple 2D

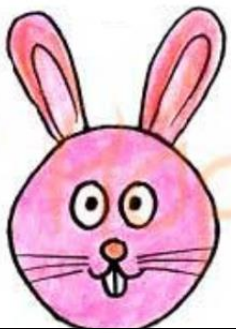
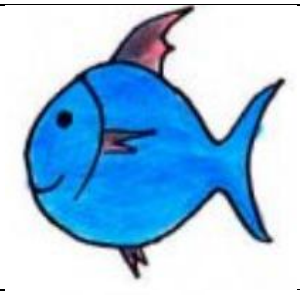
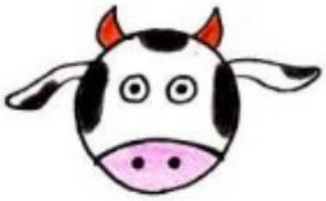
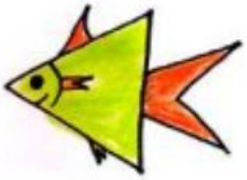
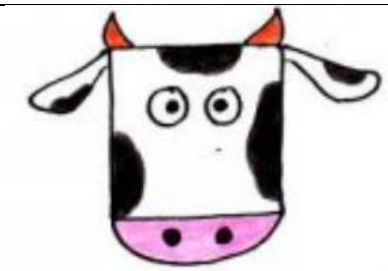
***Scopul lucrării:*** Obţinerea cunoştinţelor practice în sinteza scenelor grafice 2D statice, utilizînd primitivele grafice simple a bibliotecii p5.js.

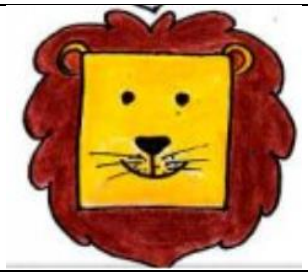
***Sarcina lucrării:***

1. Elaborați un program pentru sinteza unei scene 2D statice utilizîng cel puțin 6 primitive grafice de diferite cum ar fi - *arc()*, *ellipse()*, *circle()*, *line()*, *point()*, *quad()*, *rect()*, *square()*, *triangle()*, primitivele trebuie să fie cu diferite atribute, lucrarea trebuie semnată (numele prenumele grupa) în colțul dreapta jos a ecranului.
2. Elaborați unprogram care crează personajul conform variantei indicate de profesor. Variantele sunt indicate în tabelul 1.1. Pentru crearea acestei imagini pe pași puteți consulta pagina https://ja-rastu.ru/raskraski/uroki/1079-poshagovoe-risovanie-zhivotnyh-iz-geometricheskih-figur.html

**Tabelul 1.1 Variantele pentru realizarea lucrarii de laborator**

| Varianta | Personajul | Varianta | Personajul |
|---|---|---|---|
| 1 |  | 11 |  |
| 2 |  | 12 |  |
| 3 |  | 13 |  |

| | | | |
|---|---|---|---|
| 4 |  | 14 |  |
| 5 |  | 15 |  |
| 6 |  | 16 |  |
| 7 |  | 17 |  |
| 8 |  | 18 |  |
| 9 |  | 19 |  |

| | | | |
|---|---|---|---|
| 10 |  | 20 |  |

## Exemplu de program realizat în p5.js:

```
var Width;
var Height;
var CurrentY;

function setup() {
  Width = 400;
  Height = 734;
  createCanvas(Width, Height);
}

function draw() {
  background(220);
  CurrentY = Height - 20;

  //Realizam baza (1 parte)
  stroke(6, 21, 131);
  for (let i = 0; i < 20; i++) {
    line(0 + 20, CurrentY, Width - 20, CurrentY);
    CurrentY--;
  }

  // Realizam baza (2 parte)
  stroke(15, 23, 71);
  for (let i = 0; i < 20; i++) {
    line(0 + 20 + i, CurrentY, Width - 20 - i, CurrentY);
    CurrentY--;
  }

  //Основание двери
  CurrentY += 10;
  stroke(6, 21, 121);
  for (let i = 0; i < 550; i++) {
    line(0 + 40, CurrentY, Width - 40, CurrentY);
    CurrentY--;
  }

  //Колонки
  fill(6, 21, 121);
  stroke(15, 21, 71);
  rect(40, CurrentY, 40, Height - 190);
  rect(80, CurrentY, 3, Height - 190);
  rect(83, CurrentY, 6, Height - 190);

  rect(Width - 40, CurrentY, -40, Height - 190);
  rect(Width - 80, CurrentY, -3, Height - 190);
  rect(Width - 83, CurrentY, -6, Height - 190);

  //Дверные ямы
  for (let i = 0; i < 4; i++) {
    for (let j = 0; j < 2; j++) {
```

```
    stroke(129, 153, 193);
    line(110 + j * 100, Height - 80 - i * 130, 110 + j * 100, Height - 180 - i * 130);
    line(110 + j * 100, Height - 80 - i * 130, 190 + j * 100, Height - 80 - i * 130);
    stroke(10, 14, 45);
    line(110 + j * 100, Height - 180 - i * 130, 190 + j * 100, Height - 180 - i * 130);
    line(190 + j * 100, Height - 180 - i * 130, 190 + j * 100, Height - 80 - i * 130);
  }
}

//Средняя колонка
stroke(10, 14, 45);
rect(200, CurrentY, -3, Height - 190);
stroke(16, 31, 138);
rect(203, CurrentY, -3, Height - 190);
stroke(49, 65, 157);
rect(205, CurrentY, -1, Height - 190);

//Дверь
fill(240, 240, 240);
ellipse(210, 350, 8, 30);
fill(147, 127, 68);
circle(210, 410, 10);

stroke(10, 14, 45);
line(110, Height - 80 - 2 * 130, 110, Height - 180 - 2 * 130);
line(110, Height - 80 - 2 * 130, 190, Height - 80 - 2 * 130);
line(110, Height - 180 - 2 * 130, 190, Height - 180 - 2 * 130);
line(190, Height - 180 - 2 * 130, 190, Height - 80 - 2 * 130);
fill(240, 240, 240);
stroke(240, 240, 240);
rect(120, Height - 430, 60, 80);
fill(0, 0, 0);
noStroke();
textSize(5);
text('POLICE TELEPHONE', 125, Height - 420);
textSize(10);
text('FREE', 135, Height - 405);
textSize(5);
text('FOR USE OR', 132, Height - 395);
textSize(10);
text('PUBLIC', 130, Height - 380);
textSize(5);
text('ADVICE & ASSIS', 128, Height - 370);
textSize(7);
text('PULL TO OPEN', 125, Height - 355);

//Окна
for(let j = 0; j < 2; j++) {
  stroke(129, 153, 193);
  fill(240, 240, 240);
  rect(113 + j * 100, Height-565, 75,93);

  stroke(18, 34, 129);
  line(138 + j * 100, Height-565, 138 + j * 100,Height-473);
  line(163 + j * 100, Height-565, 163 + j * 100,Height-473);
  line(113 + j * 100, Height-519, 188 + j * 100,Height-519);
}

//Верхушка
CurrentY-=40
fill(6, 21, 121);
stroke(15, 21, 71);
```

```
rect(35, CurrentY, 330,50);
stroke(3, 11, 101);
strokeWeight(10);
fill(22, 27, 46);
rect(65, CurrentY, 270,50);
strokeWeight(1);

fill(255,255,255);
noStroke();
textSize(26);
text('POLICE', 90, CurrentY+35);
text('BOX',260, CurrentY+35);
textSize(12);
text('PUBLIC', 200, CurrentY+25);
text('CALL', 209, CurrentY+39);

CurrentY-=30
fill(6, 21, 121);
stroke(15, 21, 71);
rect(65, CurrentY, 270,30);

CurrentY-=20
rect(85, CurrentY, 230,20);  }
```

Rezultatul realizării prgramului:

***Întrebări de control:***

1. Numiți primitive grafice simple.
2. Cum poate fi realizată modificarea atributelor de afişare ale primitivelor grafice?
3. Cum poate fi scris textul în mod grafic?
4. Numiți formate standard pentru imagini.

## Capitolul II