Графические преобразования 2D

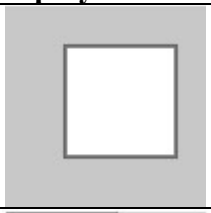
Графические преобразования 2D в библиотеке p5js:

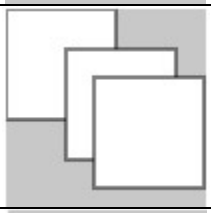
- (translate);
- (scale);
- (rotate);
- (shear);

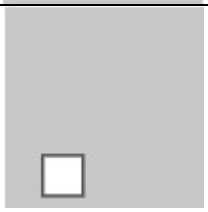
Перенос

Функция **translate()** позволяет перемещать объекты в любое место на холсте. Синтаксис переноса приведен в таблице 2.1.

Таблица 2.1. Синтаксис переноса

Код	Графический результат
translate(30, 20); rect(0, 0, 55, 55);	

rect(0, 0, 55, 55); // Draw rect at original 0,0
translate(30, 20);
rect(0, 0, 55, 55); // Draw rect at new 0,0
translate(14, 14);
rect(0, 0, 55, 55); // Draw rect at new 0,0
	

function draw() {
background(200);
rectMode(CENTER);
translate(width / 2, height / 2);
translate(p5.Vector.fromAngle(millis() / 1000, 40));
rect(0, 0, 20, 20);
}
	

В приведенном ниже примере показано выполнение переноса по коду:

```
let x = 0;
let y = 0;
let dim = 80.0;

function setup() {
  createCanvas(720, 400);
  noStroke();
}
function draw() {
  background(102);

  // Анимирует увеличение значения x
```

```
 x = x + 0.8;

 // Если фигура выходит за пределы холста, изменить начальное положение

 if (x > width + dim) {
   x = -dim;
 }

// Несмотря на то что команда ”rect” рисует фигуру с центром в начале координат, ”translate” перемещает ее в другую позицию x и y

 translate(x, height / 2 - dim / 2);
 fill(255);
 rect(-dim / 2, -dim / 2, dim, dim);

// Переносы суммируются. Обратите внимание что один прямоугольник (”rect”)
// движется в два раза быстрее чем другой, однако оба имеют одно и то же значение
// для оси x

 translate(x, dim);
 fill(0);
 rect(-dim / 2, -dim / 2, dim, dim);
}
```

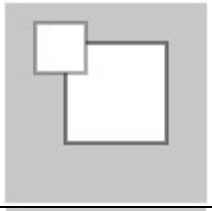
Рис. 2.1. Результат выполнения кода переноса (”translate”)

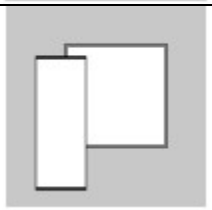
Масштабирование

Функция **scale()** преобразование, меняющее размер элемента в 2D-плоскости. Так как значение масштабирования определяется вектором, он может менять горизонтальные и вертикальные размеры в разных масштабах. В таблице 2.2 показан синтаксис масштабирования.

Таблица 2.2. Синтаксис масштабирования

Код	Графический результат

rect(30, 20, 50, 50);
scale(0.5);
rect(30, 20, 50, 50);
	

rect(30, 20, 50, 50);
scale(0.5, 1.3);
rect(30, 20, 50, 50);
	

Параметрами функции **scale()** являются значения, заданные в виде десятичных процентов. Например, метод масштабирования вызовов (2.0) увеличит размер фигуры на 200 процентов. Объекты простираются от начала координат. В приведенном ниже примере показано, как накапливаются преобразования, а также как масштабирование и перевод взаимодействуют в соответствии с их порядком.

```
let a = 0.0;
let s = 0.0;

function setup() {
  createCanvas(720, 400);
  noStroke();

// Нарисуйте прямоугольники с опорной точкой в центре

  rectMode(CENTER);
}

function draw() {
  background(102);

 // Увеличьте „a” и анимируйте „s” при помощи
 // кругового движения рассчитав косинус „a”

  a = a + 0.04;
  s = cos(a) * 2;

 // Перенесите прямоугольник из начала координат в центр
 // полотна, а потом масштабируйте на „s”

  translate(width / 2, height / 2);
  scale(s);
  fill(51);
  rect(0, 0, 50, 50);
```

```
// Перенос и масштабирование накапливаются, этот перенос
// перемещает второй прямоугольник дальше вправо, чем первый
// и масштаб удваивается. Обратите внимание, что косинус делает
// „s” как отрицательным, так и положительным, таким образом,
// прямоугольник циклически переносится слева направо.

 translate(75, 0);
 fill(255);
 scale(s);
 rect(0, 0, 50, 50);
}
```

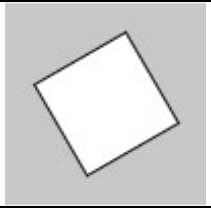
Рис. 2.2. Результат выполнения кода, комбинация между масштабированием (”scale”) и переносом (”translate”)

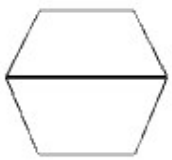
Вращение (Поворот)

Функция **rotate()** это преобразование, вращающая элемент вокруг фиксированной точки в 2D-плоскости, не искажая ее. Результатом является поворот вокруг оси.

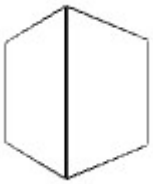
Таблица 2.3. Синтаксис поворота

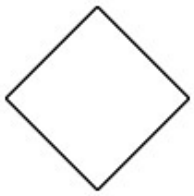
Код	Графический результат

translate(width / 2, height / 2);
rotate(PI / 3.0);
rect(-26, -26, 52, 52);
	

function setup() {
  createCanvas(100, 100, WEBGL);
}
function draw() {
  background(255);
	

rotateX(millis() / 1000);
box();
}	

function setup() {
createCanvas(100, 100, WEBGL);
}
function draw() {
background(255);
rotateY(millis() / 1000);
box();
}
	

function setup() {
createCanvas(100, 100, WEBGL);
}
function draw() {
background(255);
rotateZ(millis() / 1000);
box();
}
	

Пример. Вращение квадрата вокруг оси Z. Чтобы получить желаемый результат, параметры угла функции вращения должны быть значениями от 0 до PI * 2 (TWO_PI что составляет около 6,28). Мы также можем использовать угловые значения в градусах (0 – 360), используя метод radians() для преобразования значений. Например: rotate(radians(90)) идентичен оператору rotate(PI/2). В приведенном ниже примере каждую четную секунду выполняется вращательное движение. В нечетные секунды вращение происходит CW (по часовой стрелке) и CCW (против часовой стрелки) со скоростью, определяемой последним значением джиттера (встряхивания).

```
let angle = 0.0;
let jitter = 0.0;

function setup() {
 createCanvas(720, 400);
 noStroke();
 fill(255);

 // Нарисуйте прямоугольник в центре полотна, который будет и
 // центром вращения

 rectMode(CENTER);
}

function draw() {
 background(51);

 // для четных секунд (0, 2, 4, 6...) добавьте к вращению - джиттер (встряхивание)

 if (second() % 2 === 0) {
  jitter = random(-0.1, 0.1);
 }
```

```
  // увеличьте значение угла с помощью
  // последнего значения джиттера (встряхивания)

  angle = angle + jitter;

 // используйте косинус, чтобы получить
 // плавное движение по часовой CW и против часовой CCW стрелки
 // когда прямоугольник не вибрирует

  let c = cos(angle);

  // переместите фигуру в центр полотна

  translate(width / 2, height / 2);

  // примените окончательный поворот

  rotate(c);
  rect(0, 0, 180, 180);
}
```

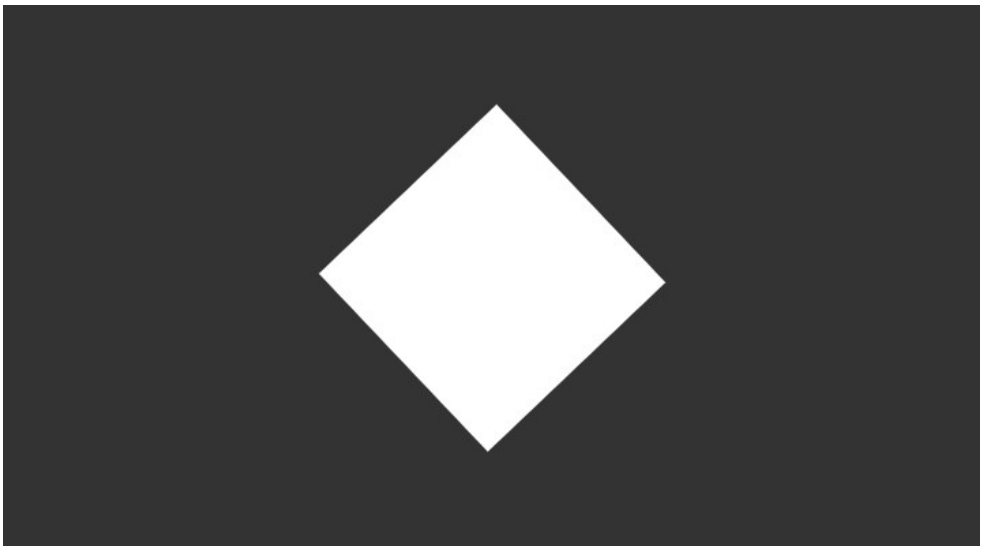


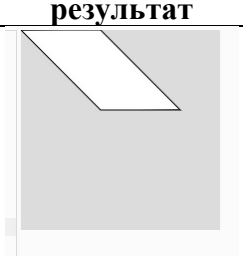
Рис. 2.3. Результат выполнения кода, вращение (″rotate″) с добавлением джиттера (встряхивания)

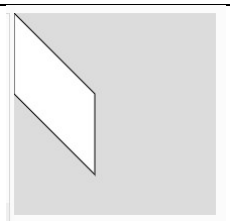
Наклон (сдвиг – Shear)

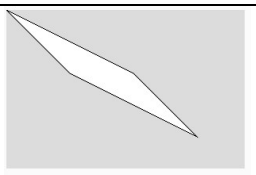
Функция **shear()** - это преобразование, вызывающее наклон преобразованного объекта.

Таблица 2.4. Синтаксис наклона

Код	Графический результат

function setup() {
  createCanvas(200, 200);
}
function draw() {
  background(220);
  shearY(PI / 4.0);
  rect(0, 0, 80, 80);
}
	

function setup() {
  createCanvas(200, 200);
}
function draw() {
  background(220);
  shearY(PI / 4.0);
  rect(0, 0, 80, 80);
}
	

function setup() {
  createCanvas(300, 200);
}
function draw() {
  background(220);
  shearX(PI / 4.0);
  shearY(PI / 4.0);
  rect(0, 0, 80, 80);
}
	

Наклон - это форма преобразования объекта вокруг оси *x* или *y* с заданным значением параметра угла. Угол должен быть указан в angleMode current. Объекты всегда наклоняются вокруг начала координат, положительные числа наклоняют объекты по часовой стрелке.

Преобразования применяются ко всему, что происходит после, а последующие вызовы функции суммируют эффект. Например, вызов shearX(PI/2), а затем еще раз shearX(PI/2) - это то же самое, что shearX(PI). Если shearX() вызывается внутри draw(), преобразование сбрасывается при повторном запуске цикла.

Технически shearX() умножает матрицу преобразования наклона на матрицу вращения. Эта функция также может управляться функциями push() и pop().

Лабораторная работа № 2
Тема: 2D графические преобразования

Цель: Выполнение графических преобразований в 2D-сцене с помощью набора функций библиотеки (JavaScript) p5.js.

Задача: Разработать программу для выполнения 2D графических преобразований с помощью **translate()**, **scale()**, **rotate()** и **shear()**.

Преобразования будут применены к 2D-сцене, созданной на лабораторной работе № 1.

Примеры программы для каждого преобразования можно найти выше в таблицах 2.1, 2.2, 2.3 и 2.4.

Таблица 2.4. Варианты проведения лабораторных работ

Вариант	Угол поворота	Коэффициент масштабирования	Перенос X и Y	Наклон
1	320	0.5	10,10	QUARTER_PI
2	70	1.2	20,10	HALF_PI
3	30	2.1	15,10	HALF_PI+ QUARTER_PI
4	25	0.8	100,100	PI+ QUARTER_PI
5	100	1.3	90,85	PI+ HALF_PI
6	110	1.4	15,15	PI+ HALF_PI +QUARTER_PI
7	20	0.6	80,80	PI / 3.0
8	49	1.1	45,40	PI / 4.0
9	95	0.8	20,25	PI / 5.0
10	64	1.9	70,70	PI / 6.0
11	38	0.2	35,35	PI / 7.0
12	128	2.3	40,45	PI / 8.0
13	300	1.47	100,110	0.31
14	256	0.3	90,95	0.47
15	49	1.3	20,10	0.79
16	60	1.29	10,20	1.26
17	83	1.7	40,20	1.73
18	39	0.4	85,100	2.20
19	26	1.23	30,35	2.51
20	74	1.97	20,30	2.83
21	91	2.3	105,125	3.30
22	90	1.32	110,130	3.61
23	95	1.90	100,90	4.08
24	44	1.5	120,115	4.56
25	79	0.9	20,40	5.03