**Урок №11. Canvas-трансформации**

**Вводный блок (10 минут)**

**Цели и задачи урока:**

1. Рассмотреть возможные варианты трансформации объектов.
2. Использовать трансформацию для изменения положения объектов.

**Вспомним прошлые занятия**

1. Что такое объекты? Как обращаться к свойствам объекта?
2. Что такое функции? Как создавать функции?

**Познание нового (35 минут)**

При работе с объектом можно использовать функции для трансформации, которые, в основном, используются для 3D объектов, это перемещение, вращение и масштабирование.

Если на перемещение и размеры можно воздействовать при отрисовке объекта, то вращение изменить не получится.

**Вопрос для учеников**

Вы когда-нибудь видели вращающиеся объекты в 2D-игре? Если да, то в какой?

Сегодня мы попробуем реализовать алгоритм, который позволит вращать монетку вокруг своей оси.

**Задание**

Создайте монетку и разместите ее по центру экрана (11.1)

**Вращаем монетку**

Далее, в отрисовке каждого кадра будем вращать эту монетку вокруг своей оси. Для этого используется функция **rotate**, которая вращает элементы относительно выбранной точки координат. Именно поэтому очень важно сначала переместиться в центр объекта с помощью функции **translate**, а уже потом вращать объект.

function move() {

context.clearRect(0, 0, canvas.width, canvas.height);

context.translate(coin.x + coin.width / 2, coin.y + coin.height / 2);

context.rotate(Math.PI / 180 \* 2);

context.translate(-coin.x - coin.width / 2, -coin.y - coin.height / 2)

context.drawImage(coin.image, coin.x, coin.y, coin.width, coin.height);

requestAnimationFrame(move);

}

move();

**Вращение двух или более объектов**

В этом случае необходимо разобрать две полезные функции **save** и **restore.**

**Save** - сохраняет местоположение всех элементов.

**Restore** - возврат в исходному холсту.

Таким образом, перед операцией вращения, сначала происходит сохранение всех данных, затем переход к нужной точки и вращение относительно этой точки, потом возврат к левому верхнему краю экрана и возвращение к исходному холсту.

Добавим еще один объект и дополнительно к каждому объекту пропишем свойство angle:

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

const coinImage = new Image();

coinImage.src = "../coin.png";

const meteoriteImage = new Image();

meteoriteImage.src = "../meteorite.png";

const coin = {

x:300,

y:300,

width: 100,

height: 100,

image: coinImage,

angle: 0,

changeAngle: 10

};

const meteorite = {

x:500,

y:500,

width: 100,

height: 100,

image: meteoriteImage,

angle: 0,

changeAngle: 10

};

function move(){

context.drawImage(coin.image, coin.x, coin.y, coin.width, coin.height);

context.drawImage(meteorite.image, meteorite.x, meteorite.y, meteorite.width, meteorite.height);

requestAnimationFrame(move);

}

move()

Далее создадим функцию rotate для вращения и вызовем функции save и **restore:**

function rotate(object){

context.save();

context.restore();

}

Используем функцию translate для переноса координат в центр объекта и обратно:

function rotate(object) {

context.save();

**context.translate(object.x + object.width / 2, object.y + object.height / 2);**

**context.translate(-(object.x + object.width / 2), -(object.y + object.height / 2));**

context.restore();

}

Воспользуемся функцией rotate для перемещения на определенный угол (задается в радианах, поэтому здесь представлено преобразование)

function rotate(object) {

context.save();

context.translate(object.x + object.width / 2, object.y + object.height / 2);

**context.rotate(Math.PI / 180 \* object.angle);**

context.translate(-(object.x + object.width / 2), -(object.y + object.height / 2));

context.restore();

}

В конец функции, перед **context.restore** добавим отрисовку элемента и смену угла

function rotate(object) {

context.save();

context.translate(object.x + object.width / 2, object.y + object.height / 2);

context.rotate(Math.PI / 180 \* object.angle);

context.translate(-(object.x + object.width / 2), -(object.y + object.height / 2));

**context.drawImage(object.image, object.x, object.y, object.width, object.height);**

**object.angle = (object.angle + object.changeAngle) % 360;**

context.restore();

}

function move() {

context.clearRect(0, 0, canvas.width, canvas.height);

rotate(coin);

rotate(meteorite);

requestAnimationFrame(move);

}

move()

**Некомпьютерная активность (15 минут)**

Активность 1. Ученики берут лист бумаги и придумывают трансформации, такие как вращение, масштабирование или сдвиг. Например, попросите их скрутить лист в трубку, а затем развернуть его обратно и ответить на вопрос: “Какие трансформации и как они влияли на лист бумаги”?.

Активность 2. Робот. Выбирается один ученик, которому нужно добраться от своего места до выходной двери и при этом выполнять только те команды, которые ему говорят другие ученики по очереди. За раз робот может сделать только 1 команду.

**Работа над проектом (25 минут)**

Тема: Создание нового проекта и написание программы, в которой используются матрицы масштабирования, вращения и перемещения.

Далее мы попробуем вращать монетку по оси x.

Для такого вращения не нужна функция rotate. В данном случае мы должны изменять ширину объекта. Сначала создадим проект, где монетку поместим в центр

const canvas = document.querySelector('canvas');

const context = canvas.getContext('2d');

const coinImage = new Image();

coinImage.src = "../coin.png"

const coin = {

x: 300,

y: 300,

width: 100,

height: 100,

image: coinImage,

changeValue: 1,

maxHeight: 100,

};

В объекте задано свойство **changeValue. changeValue** - величина, отвечающая за изменение ширины объекта. Мы точно знаем, что максимальный размер монетки равен 100, поэтому зададим еще одно свойство **maxWidth** - максимальная ширина монетки.

Создаем функцию **rotateX** и передаем в нее монетку и сразу отрисуем ее.

function rotateY(object) {

context.drawImage(object.image, object.x, object.y, object.width, object.height);

}

Изменяем ширину объекта на величину **changeValue** и проверяем ее размер, чтобы монетка не смогла стать больше допустимого размера. Для этого напишем условие по отслеживанию ширины объекта. Изменение положение по оси x нужно для того, чтобы объект оставался на одном месте.

function rotateY(object) {

object.width -= object.changeValue;

object.x += object.changeValue \* 0.5;

if (Math.abs(object.width) == 100) {

object.changeValue = -object.changeValue;

}

context.drawImage(object.image, object.x, object.y, object.width, object.height);

}

**Задание**

1. Сделать еще один любой объект и задать вращение по оси y (11.3)

**Рефлексия (5 минут)**

"Что нового вы узнали сегодня?"

"Какие виды трансформаций бывают? Какими способами можно вращать объект?"