**Урок №9. Реализация стрельбы с использованием функций**

**Вводный блок (10 минут)**

**Цели и задачи урока:**

1. Рассмотреть синтаксис создания функций.
2. Рассмотреть механику стрельбы.
3. Реализовать алгоритм стрельбы.

Мы уже разобрали многие механики игр (перемещение, сбор, прыжок и т.д.). Сегодня на практике разберем механику стрельбы, но сначала мы разберем как работают функции.

**Познание нового (30 минут)**

Функция - блок кода, который можно вызывать по одному названию. Такие приемы удобно использовать тогда, когда необходимо сделать несколько однотипных действий, например, перемещение объектов на фиксированную величину, обработку однотипных данных (найти дискриминант квадратного уравнения и т.д.)

Предположим, нам нужно создать функцию для создания пули. В таком случае синтаксис создания будет следующий:

function createBullet(object) {

}

Для создания функций используется ключевое слово function, далее идет название функции, по которому она будет вызываться, ну и в скобках находится переменная, которая передается в функцию. Таких переменных может и не быть, а может быть и несколько в зависимости от назначения функции.

**Вопрос для учеников**

1. Какие параметры мы точно должны знать при создании пули? (координаты вылета пули)
2. Сколько тогда будет параметров у функции? (два)

Тогда давайте изменим функцию **createBullet** и будем работать с ней дальше

function createBullet(x, y) {

let bullet\_x = x;

let bullet\_y = y;

return [bullet\_x, bullet\_y];

}

Оператор **return** - оператор, который возвращает результат работы функции. В данном случае результатом работы функции является массив с заданными координатами.

Теперь создадим массив bullets и добавим в него несколько пуль

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

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

**let bullets = []**

function createBullet(x, y) {

let bullet\_x = x;

let bullet\_y = y;

return [bullet\_x, bullet\_y];

}

**bullets.push(createBullet(100, 100));**

**bullets.push(createBullet(100, 200));**

**bullets.push(createBullet(100, 300));**

function move() {

requestAnimationFrame(move);

}

move()

**Задание**

1. Создать функцию drawBullets, в которую передается массив bullets и отрисовываются все элементы массива и протестировать ее (9.2)
2. Создать функцию moveBullets, в которую передается массив bullets и происходит смещение координаты **y** на 10 у каждой пули и протестировать ее (9.3)

**Некомпьютерная активность (15 минут)**

Игра «Программист-помощник»

Цель игры: закрепить знания о функциях в программировании и их применении, а также о работе с canvas на JavaScript.

Ход игры: участники делятся на пары. Один участник из каждой пары играет роль программиста, который должен написать программу с использованием функций и canvas на JavaScript. Второй участник играет роль помощника, который может задавать вопросы о программе, но не может видеть код. Задача помощника — помочь программисту написать программу. Затем участники меняются ролями.

**Например**:

1. Первый участник говорит: «Мне нужно написать программу, которая рисует прямоугольник на холсте».
2. Второй участник спрашивает: «Какие данные для этого нужно использовать?»
3. Первый участник отвечает: «Координаты построения прямоугольника, его размеры и цвет».
4. Второй участник предлагает: «Откуда ты будешь брать эти данные?».
5. Первый участник соглашается: «Создам переменные w,h,x,y и color и задам для них нужные значения».
6. Второй участник продолжает: «А как ты будешь рисовать график?»
7. Первый участник решает: «Создам функцию drawRectangle, в которую буду передавать все созданные переменные. В функции, с помощью функции context.fillRect построю прямоугольник».
8. Второй участник одобряет: «Это хороший план».

Задача игроков — придумать интересную и логичную программу, соблюдая условия задачи. Побеждает тот, кто сможет наиболее увлекательно написать программу и объяснить её помощнику.

**Работа над проектом (30 минут)**

Тема: Создание нового проекта и написание программы, в которой реализован полет космического корабля и механика стрельбы для космического корабля.

Перейдем к созданию персонажа. У нас это будет космический корабль.

**Задание**

1. Создать персонаж «Корабль» и отобразить его в нижней части экрана по центру (9.4)
2. Для созданного персонажа задать управление по горизонтали (**A** (влево) и **D** (вправо) (9.4)

**Создание пули у носа корабля**

Для создания пули у носа корабля нужно знать ее точные размеры, а также координаты самого корабля.

Координаты верхнего левого угла корабля хранятся в координатах **sheep\_x** и **sheep\_y**, а значит к координате **sheep\_x** нужно добавить половину ширины корабля и отнять половину размера пули. От координаты **sheep\_y** нужно отнять размер пули.

Стрелять мы будем на пробел, тогда вся конструкция выглядит следующим образом (9.4)

window.addEventListener('keydown', (event) => {

if (event.key.toLowerCase() == "a") {

dx = -5;

}

if (event.key.toLowerCase() == "d") {

dx = 5;

}

**if (event.key.toLowerCase() == " ") {**

**bullets.push(createBullet(sheep\_x + 100 - 20, sheep\_y - 20));**

**}**

})

Отлично, теперь пули создаются у носа корабля и летят в правильном направлении.

**Задание**

1. Дополнить игру, создав метеориты, которые летят в сторону корабля
2. Добавить коллизию для всех элементов.
3. Добавить проигрыш, если в корабль попадет больше трех метеоритов.

**Рефлексия (5 минут):**

"Что нового вы узнали сегодня?"

"Что такое функция? Как передавать данные в функцию? Какое слово используется для окончания работы функции и возвращении результата?"