**Лабораторная работа №9**

**NWjs фреймоврк. Создание декстопных приложений на JS.**

**Работа с командами npm.**

1. Для эффективной работы нужно наличие редактора кода и установленный Nodejs (<https://nodejs.org/uk/>)
2. В редакторе кода создать пустой проект и проинициализировать его.

В командной строке набрать:

npm init-y

В каталоге проинициализируется проект с файлом **package.json**

{

"name": "game\_ball\_js",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"keywords": [],

"author": "",

"license": "ISC"

}

1. Для проекта нужно установить фреймворк [NW.js](http://nwjs.io/), который позволяет вам вызывать все [модули](https://www.npmjs.org/)[Node.js](https://nodejs.org/) напрямую из DOM и обеспечивает новый способ написания приложений со всеми веб-технологиями.
2. В командной строке набрать:

npm install nw -D

npm install nw-builder -D

В результате **package.json** пропишутся две зависимости (версии скриптов по состоянию на февраль 2020)

"devDependencies": {

"nw": "^0.44.3",

"nw-builder": "^3.5.7"

}

1. Поменяем в **package.json** свойство scripts для запуска фреймворка [NW.js](http://nwjs.io/). test переименуем в dev и укажем путь для запуска фреймворка "nw src/". Также добавим свойство prod для сборки на продкашн нашего приложения "prod": "nwbuild --platforms win64 --buildDir dist/ src/"

"scripts": {

"dev": "nw src/",

"prod": "nwbuild --platforms win64 --buildDir dist/ src/"

},

1. Создаем в проекте папку src и переходи в нее (команда в терминале cd src). Далее инициализируем в этой папке новый **package.json** командой

npm inint -y

1. Отредактируем этот файл **package.json**

Сделаем корневой файл для запуска приложения

"main": "views/main.html",

Создадим новое свойство window, которое содержит в себе объект для того чтобы задать различные параметры нашего приложения. Различные свойства для window и их значение можно прочитать из документации <https://github.com/nwjs/nw.js/wiki/Manifest-format>

{

"name": "src",

"version": "1.0.0",

"description": "",

"main": "views/main.html",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"window": {

"title": "IPZ\_ZIEIT",

"icon": "assets/icon.png",

"toolbar": true,

"width": 700,

"height": 700,

"min\_width": 700,

"min\_height": 700,

"max\_width": 700,

"max\_height": 700,

"position": "mouse"

},

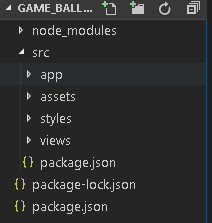
"keywords": [],

"author": "",

"license": "ISC"

}

1. В каталоге src нужно создать 4 папки app, views, assets styles.



1. В папке views создайте файл main.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<meta http-equiv="X-UA-Compatible" content="ie=edge" />

<link rel="stylesheet" href="../styles/style.css" />

<title>Разбей блоки для ИПЗ!</title>

</head>

<body>

<h1>Лабораторная работа для ИПЗ</h1>

<button id="rules-btn" class="btn rules-btn">Показать правила</button>

<div id="rules" class="rules">

<h2>Как играть:</h2>

<p>

Используй правую и левую кнопки клавиатуры для передвижения платформы

для того чтобы отбить вверх мячь и разить блоки.

</p>

<p>Если вы уроните мячь ваш результат обнулиться а блоки восстановятся</p>

<button id="close-btn" class="btn">Закрыть</button>

</div>

<canvas id="canvas" width="800" height="600"></canvas>

<script src="../app/index.js"></script>

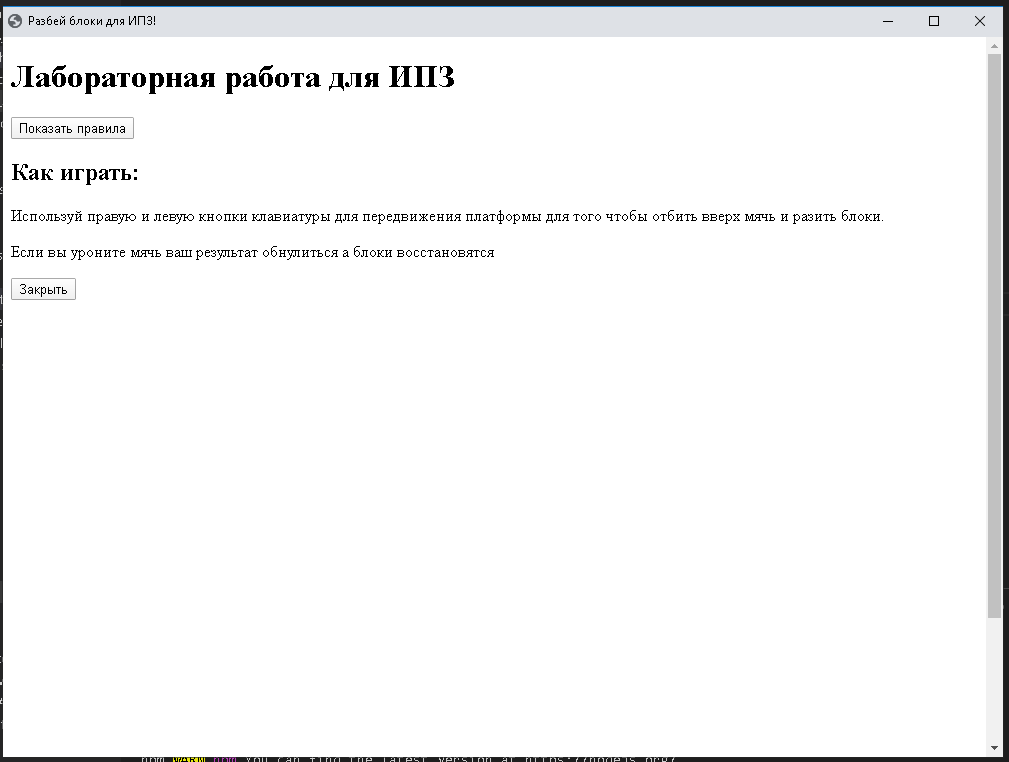
</body>

</html>

Тег **canvas** найти информацию в интернете для чего и как используется!

Для отображения текущего результата находясь в корневой папке с проектом запустить команду:

npm run dev



1. В папке styles создайте файл style.css

\* {

box-sizing: border-box;

}

body {

background-color: #0095dd;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

font-family: Arial, Helvetica, sans-serif;

min-height: 100vh;

margin: 0;

}

h1 {

font-size: 30px;

color: #fff;

}

canvas {

background: #f0d3d3;

display: block;

border-radius: 5px;

}

.btn {

cursor: pointer;

border: 0;

padding: 10px 20px;

background: rgb(25, 1, 70);

color: #fff;

border-radius: 5px;

}

.btn:focus {

outline: 0;

}

.btn:hover {

background: #222;

}

.btn:active {

transform: scale(0.98);

}

.rules-btn {

position: absolute;

top: 10px;

left: 10px;

}

.rules {

position: absolute;

top: 0;

left: 0;

background: #333;

color: #fff;

min-height: 100vh;

width: 400px;

padding: 20px;

line-height: 1.5;

transform: translateX(-400px);

transition: transform 1s ease-in-out;

}

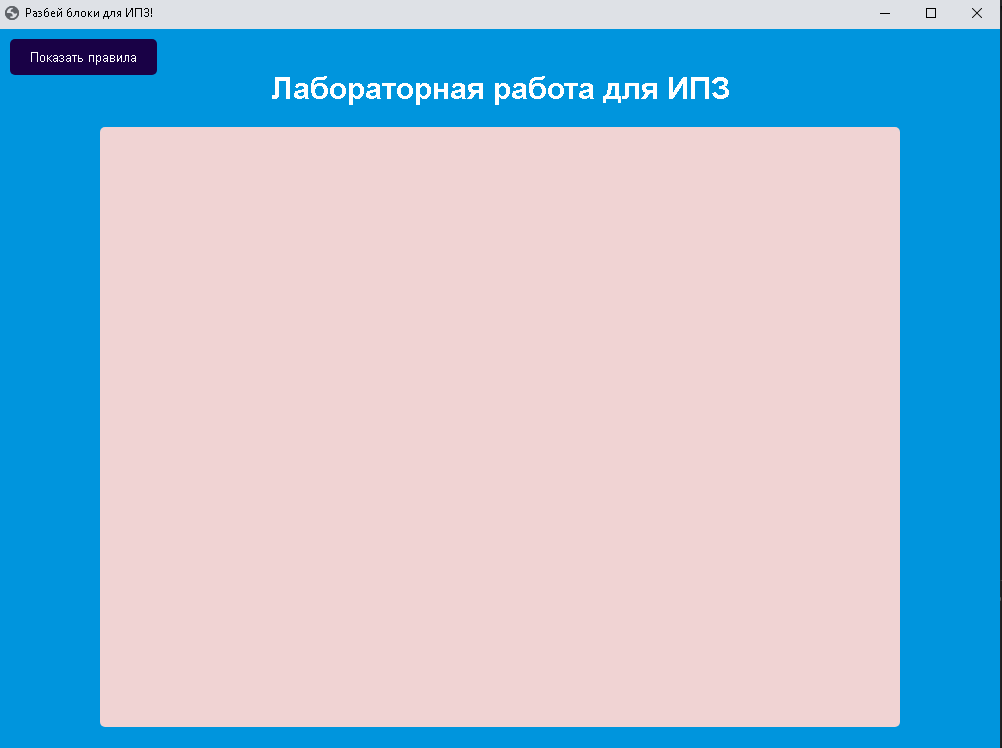
.rules.show {

transform: translateX(0);

}

После написания стилей проверьте текущий результат, находясь в корневой папке с проектом запустить команду:

npm run dev



1. В папке app создать файл index.js. Первым делом нужно навесить обработчик события на кнопки «Показать правила» и «Закрыть»

const rulesBtn = document.getElementById("rules-btn");

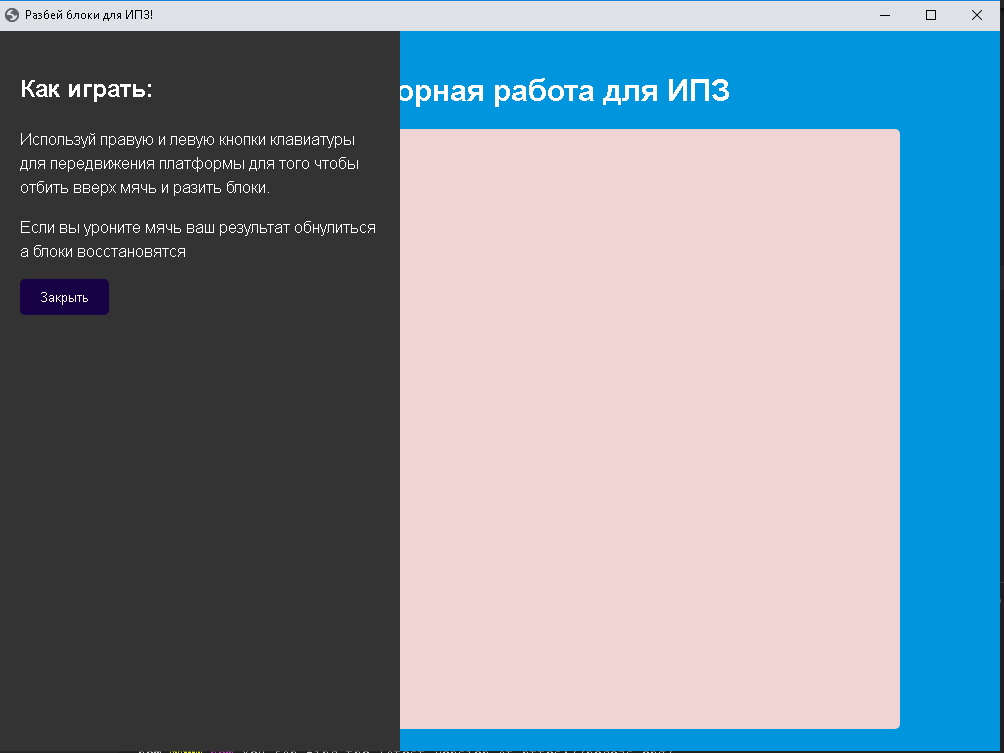
const closeBtn = document.getElementById("close-btn");

const rules = document.getElementById("rules");

rulesBtn.addEventListener("click", () => rules.classList.add("show"));

closeBtn.addEventListener("click", () => rules.classList.remove("show"));

Проверить как работает:



1. В этом же файле index.js создать свойства прототипа мяча, блока, платформы и также создать через canvas объект представляющий двумерный контекст для будущей отрисовки.

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

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

// Create ball props

const ball = {

x: canvas.width / 2,

y: canvas.height / 2,

size: 10,

speed: 4,

dx: 4,

dy: -4

};

// Create paddle props

const paddle = {

x: canvas.width / 2 - 40,

y: canvas.height - 20,

w: 80,

h: 10,

speed: 8,

dx: 0

};

// Create brick props

const brickInfo = {

w: 70,

h: 20,

padding: 10,

offsetX: 45,

offsetY: 60,

visible: true

};

1. Сгенерировать все блоки через заданные константы brickRowCount, brickColumnCount.

const brickRowCount = 9;

const brickColumnCount = 5;

const bricks = [];

for (let i = 0; i < brickRowCount; i++) {

bricks[i] = [];

for (let j = 0; j < brickColumnCount; j++) {

const x = i \* (brickInfo.w + brickInfo.padding) + brickInfo.offsetX;

const y = j \* (brickInfo.h + brickInfo.padding) + brickInfo.offsetY;

bricks[i][j] = { x, y, ...brickInfo };

}

}

1. Ввести переменную score для генерации количества балов. Прорисовать ее на canvas. Создать функции для обновления и прорисовки текущих значений. Запустить проект и проверить.

let score = 0;

// Draw score oon canvas

function drawScore() {

ctx.font = "20px Arial";

ctx.fillText(`Score: ${score}`, canvas.width - 100, 30);

}

function draw() {

// clear canvas

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

drawScore();

}

// Update canvas drawing and animation

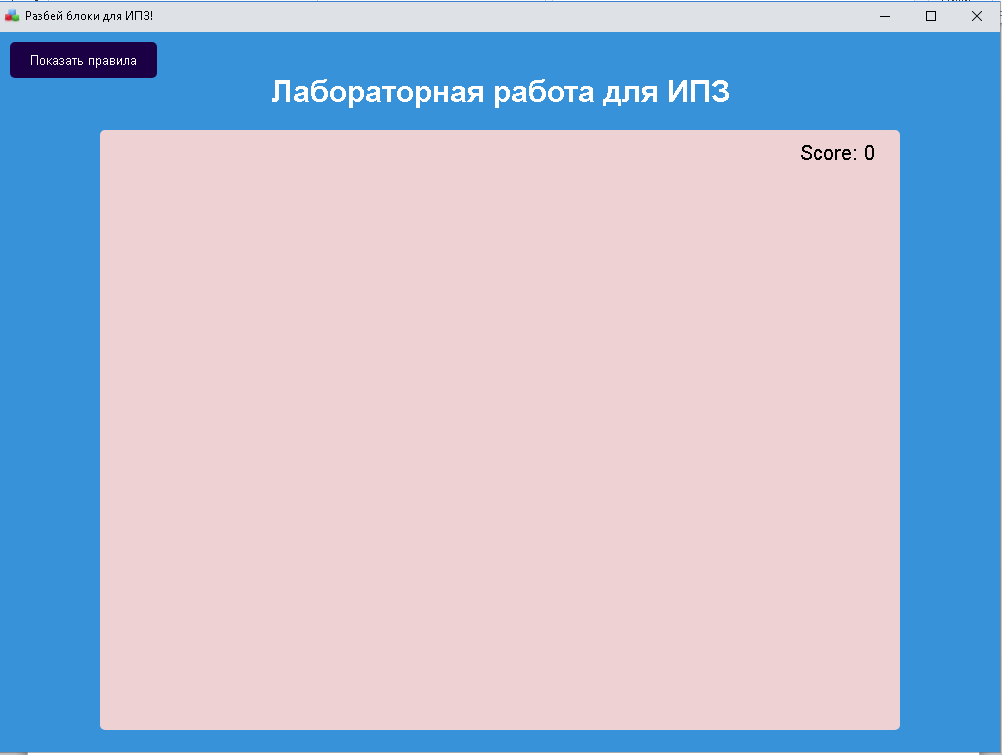
function update() {

// Draw everything

draw();

}

update();



1. Прорисовать на площадке платформу и мяч.

// Draw ball on canvas

function drawBall() {

ctx.beginPath();

ctx.arc(ball.x, ball.y, ball.size, 0, Math.PI \* 2);

ctx.fillStyle = '#0095dd';

ctx.fill();

ctx.closePath();

}

// Draw paddle on canvas

function drawPaddle() {

ctx.beginPath();

ctx.rect(paddle.x, paddle.y, paddle.w, paddle.h);

ctx.fillStyle = '#0095dd';

ctx.fill();

ctx.closePath();

}

И добавить эти функции в функцию draw()

function draw() {

// clear canvas

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

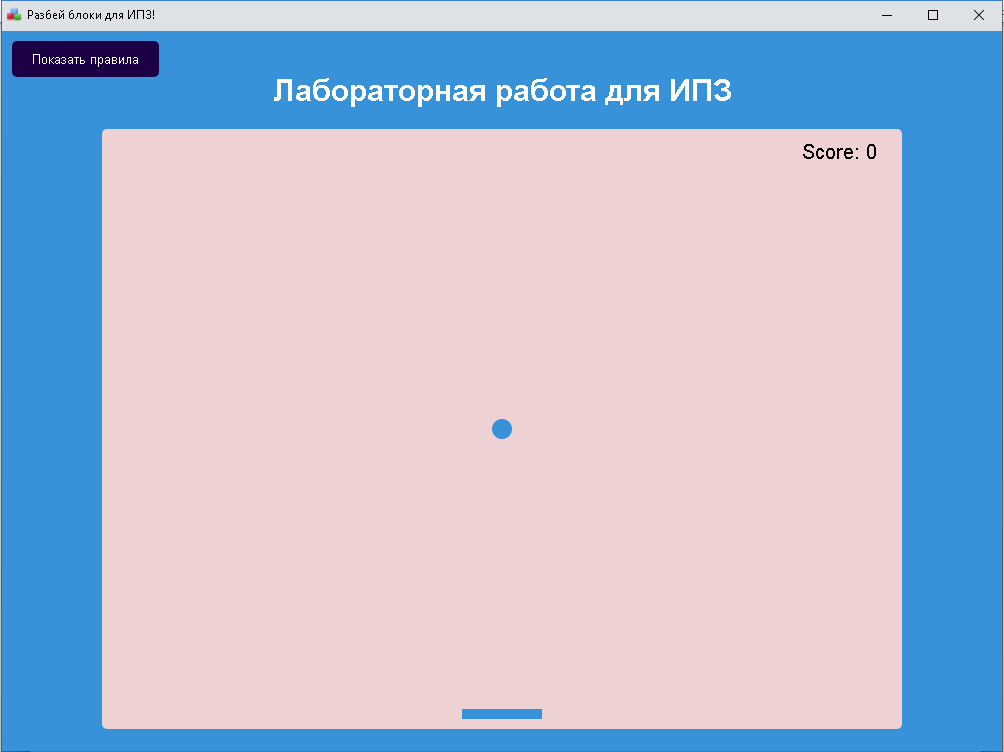
drawScore();

drawBall();

drawPaddle();

}

Проверить результат (должна появиться платформа и мяч)



1. Прорисовать блоки на поле.

function drawBricks() {

bricks.forEach(column => {

column.forEach(brick => {

ctx.beginPath();

ctx.rect(brick.x, brick.y, brick.w, brick.h);

ctx.fillStyle = brick.visible ? '#0095dd' : 'transparent';

ctx.fill();

ctx.closePath();

});

});

}

И добавить эту функции в функцию draw()

function draw() {

// clear canvas

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

drawScore();

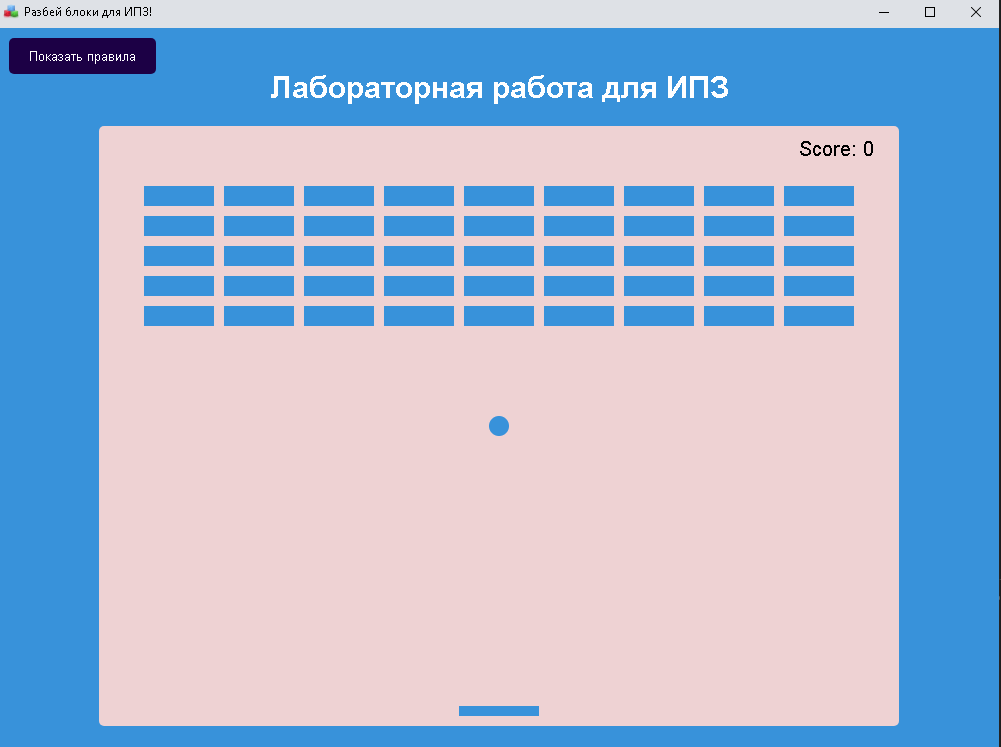
drawBall();

drawPaddle();

drawBricks();

}

Проверить результат (должны появиться блоки)



1. Создать функции по перемещению платформы по оси х.

function movePaddle() {

paddle.x += paddle.dx;

// Wall detection

if (paddle.x + paddle.w > canvas.width) {

paddle.x = canvas.width - paddle.w;

}

if (paddle.x < 0) {

paddle.x = 0;

}

}

Добавить ее в функцию update()

function update() {

// Draw everything

movePaddle();

draw();

}

1. Создать перехватчики событий по нажатию на клавиши влево и вправо и прописать действия над платформой.

// Keyboard event handlers

document.addEventListener("keydown", keyDown);

document.addEventListener("keyup", keyUp);

// Keydown event

function keyDown(e) {

if (e.key === "Right" || e.key === "ArrowRight") {

paddle.dx = paddle.speed;

} else if (e.key === "Left" || e.key === "ArrowLeft") {

paddle.dx = -paddle.speed;

}

}

// Keyup event

function keyUp(e) {

if (

e.key === "Right" ||

e.key === "ArrowRight" ||

e.key === "Left" ||

e.key === "ArrowLeft"

) {

paddle.dx = 0;

}

}

1. Для того чтобы указывать браузеру на то, что вы хотите произвести анимацию, и просит его запланировать перерисовку на следующем кадре анимации использую встроенный метод глобального объекта window - requestAnimationFrame(). В качестве параметра метод получает функцию, которая будет вызвана перед перерисовкой.

function update() {

// Draw everything

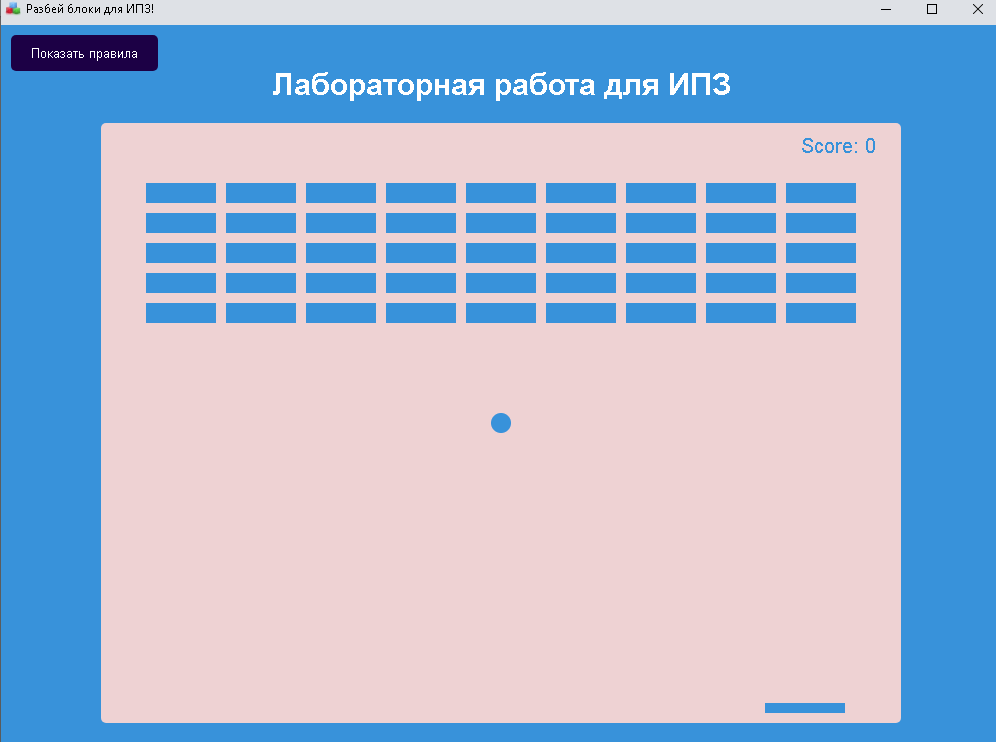
movePaddle();

draw();

requestAnimationFrame(update);

}

Запустите программу и проверьте что платформа двигается



1. Создать функцию перемещения мяча.

function moveBall() {

ball.x += ball.dx;

ball.y += ball.dy;

// Wall collision (right/left)

if (ball.x + ball.size > canvas.width || ball.x - ball.size < 0) {

ball.dx \*= -1; // ball.dx = ball.dx \* -1

}

// Wall collision (top/bottom)

if (ball.y + ball.size > canvas.height || ball.y - ball.size < 0) {

ball.dy \*= -1;

}

// console.log(ball.x, ball.y);

// Paddle collision

if (

ball.x - ball.size > paddle.x &&

ball.x + ball.size < paddle.x + paddle.w &&

ball.y + ball.size > paddle.y

) {

ball.dy = -ball.speed;

}

// Brick collision

bricks.forEach(column => {

column.forEach(brick => {

if (brick.visible) {

if (

ball.x - ball.size > brick.x && // left brick side check

ball.x + ball.size < brick.x + brick.w && // right brick side check

ball.y + ball.size > brick.y && // top brick side check

ball.y - ball.size < brick.y + brick.h // bottom brick side check

) {

ball.dy \*= -1;

brick.visible = false;

increaseScore();

}

}

});

});

// Hit bottom wall - Lose

if (ball.y + ball.size > canvas.height) {

showAllBricks();

score = 0;

}

}

Добавить эту функцию в функцию update()

function update() {

// Draw everything

movePaddle();

moveBall();

draw();

requestAnimationFrame(update);

}

1. В случае потери мяча добавить все блоки на экран, поэтому создать функцию для прорисовки всех блоков

// Make all bricks appear

function showAllBricks() {

bricks.forEach(column => {

column.forEach(brick => (brick.visible = true));

});

}

1. В случае попадания по блоку увеличить счет в случае потери мяча вызывать функцию прорисовки всех блоков

// Increase score

function increaseScore() {

score++;

if (score % (brickRowCount \* brickRowCount) === 0) {

showAllBricks();

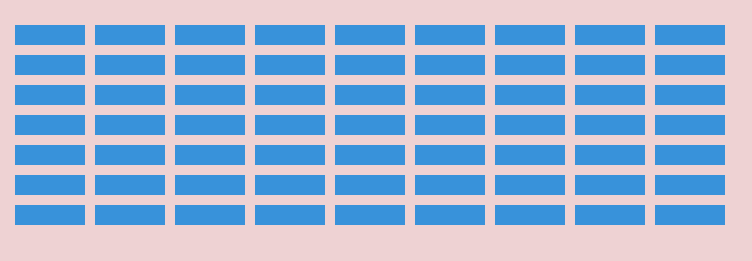
}

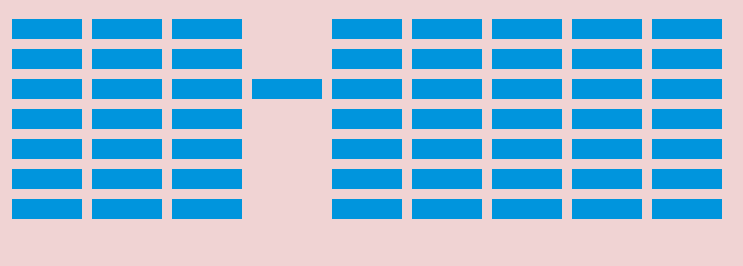
}

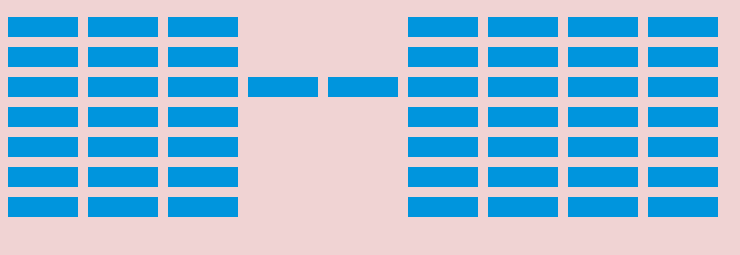
Проверить готовый вариант программы.

Самостоятельно:

1. Увеличивать скорость мяча при попадании в каждые 10 блоков на 1 и т.д.
2. Поменять форму отображения блоков (любой ваш вариант)







1. Если набрали 25 очков, вывести на экране вместо счета, сообщение «ИПЗ лучшие!!!»

Сбилдить проект под win64 платформу:

npm run build