

МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Кафедра 319 «Системы интеллектуального мониторинга»

**КУРСОВАЯ РАБОТА**

по дисциплине  
«Программирование»

Разработка программы для визуализации работы машины Тьюринга.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | | |  | Еремеев П.Н. |
| Группа | | | М3О-235Б-21 |  |
| Руководитель | | |  | Рубцов Е.А. |
| Оценка |  | Дата защиты « » декабря 2022 г. | | |

**Москва 2022**

МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Кафедра 319 «Системы интеллектуального мониторинга»

**З А Д А Н И Е**

на курсовую работу по дисциплине

«Программирование»

Студент М3О-235Б-21 Еремеев Павел Николаевич

(№ группы, Ф. И. О.)

Тема Разработка программы для визуализации работы машины

Тьюринга.

\_

Перечень вопросов, подлежащих разработке в курсовой работе

1. Разработка веб-приложения для игры в Flappy Bird.

2. Изучение правил игры.

3. Составление рейтинга результатов на основе лучших результатов разных игроков.

4. Кастомизация игры.

Рекомендуемая литература

1. Крокфорд Дуглас. Как устроен JavaScript / пер. с англ. — СПб.: Питер, 2019. — 304 с. — (Серия «Для профессионалов»). ISBN 978-5-4461-1260-9

.

2. Фримен Э., Робсон Э.Ф88 Изучаем программирование на JavaScript / пер. с англ. — СПб.: Питер, 2015. — 640 с.: ил. — (Серия «Head First O’Reilly») ISBN978-5-496-01257-7

.

Задание выдано « » 2022 г.

Руководитель *Рубцов Е.А.*, *старший преподаватель*

(Ф. И. О., должность, подпись)

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc121937672)

[1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc121937673)

[2. ВАРИАНТЫ ПОЛЬЗОВАНИЯ ПРОГРАММЫ 6](#_Toc121937674)

[3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ ПРОГРАММЫ 7](#_Toc121937675)

[4. РЕАЛИЗАЦИЯ 9](#_Toc121937676)

[ЗАКЛЮЧЕНИЕ 13](#_Toc121937677)

[СПИСОК ЛИТЕРАТУРЫ 14](#_Toc121937678)

[ПРИЛОЖЕНИЕ А 15](#_Toc121937679)

# **ВВЕДЕНИЕ**

Люди на протяжении нескольких десятилетий делятся информацией в интернете. Для «сёрфинга» в интернете используется сайты, которые состоят из веб-страниц (HTML-документов).

Создание языка JavaScript связано с «оживлением» веб-страниц. Программы, написанные на данном языке, могут быть написаны, как в самом HTML-файле, так и в отдельном файле, подключаемом в HTML-документ. Программный код, написанный на JavaScript, выполняется в виде обычного текста. Для своего выполнения код не проходит предкомпиляцию, в отличие от Java. Однако JavaScript и Java имеют общее: программа, написанная на этих языках, будет выполняться на всех устройствах, которые имеют специальную программу, соответствующую её языку программированию. Для Java это JVM (Java Virtual Machine), а для JavaScript – движок JavaScript engine. [1] Преимущество JavaScript заключается в том, что в легкодоступности, так как в настоящее время все браузеры, которые зачастую изначально установлены на компьютере, имеют данный движок. Чего нельзя сказать о виртуальной машине Java, которую необходимо скачать и установить. JavaScript в первую очередь был создан для работы с веб-страницы, поэтому данный язык имеет полную интеграцию с HTML, CSS – технологиями. Сейчас язык широко используется в разных средах разработки программ.

Поддержка JavaScript работы с Canvas расширяет креативность проектов программистов, благодаря ей упрощается работа с анимацией игр. С помощью языка гипертекстовой разметки HTML и языка программирования JavaScript была создана программа для игры во «Flappy Bird».

# **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

В апреле 2013 года вьетнамский разработчик Нгуен Донг решил создать мобильную игру. Игра должна была быть простой, которая бы напоминала игры Nintendo.

В мае 2013 года игра вышла в магазине приложений «App Store». Игра была бесплатной, так как разработчик не гнался за большой выручкой. Изначально игра не была популярна, спустя некоторое время мобильное приложение для убивания времени заняло первые места по скачиванию в топах многих стран. «Flappy Bird» развирусилась, люди скачивали игру и соревновались в том, кто больше наберёт очков. [2]

В феврале 2014 года игра прекратила своё существование. Многочисленные отзывы о том, что приложение затягивает на несколько часов, отрывая людей от реальной жизни, побудили автора удалить её со всех площадок. [3]

Суть игры заключалась в управлении уткой, чтобы уклоняться от труб, между которыми было пространство, где утка могла пролететь. Пролетая трубу, игрок получал 1 очко к счёту. Управление уткой было очень простым – при нажатии на экран телефона утка взлетала, при отсутствии нажатия – утка падала. После прикосновения утки с трубой или землёй игра заканчивалась.

Игра решала проблему убивания времени, помогала скоротать время в ожидании чего-либо. Целевой аудиторией приложения являются люди, находящиеся в ожидании чего-либо в течение долгого времени.

# **ВАРИАНТЫ ПОЛЬЗОВАНИЯ ПРОГРАММЫ**

Действия, которые может выполнять игрок в программе для игры в Flappy Bird, показаны на рисунке 1.

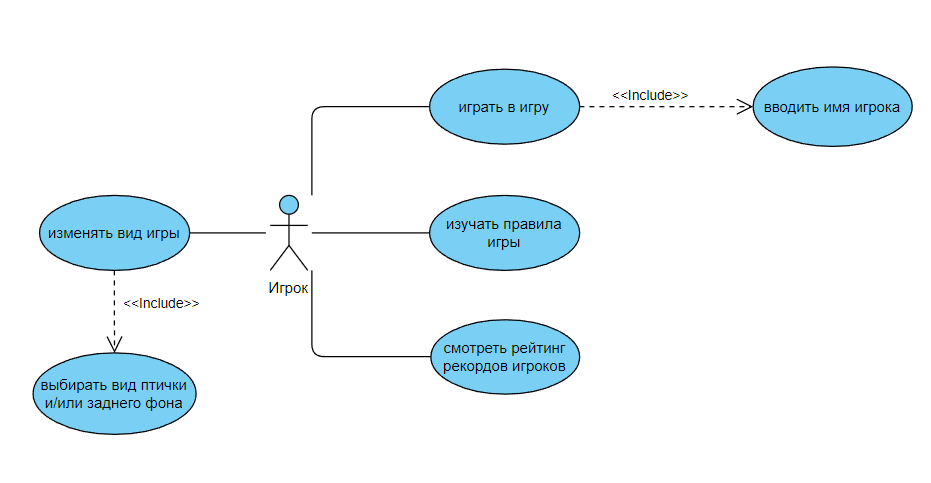


Рисунок 1 – Диаграмма вариантов пользования

# **СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ ПРОГРАММЫ**

**Играть в игру**

1. Программа просит игрока ввести имя.
2. Игрок вводит своё имя.
3. Игрок просит программу сохранить его имя.
4. Программа просит игрока начать игру, нажав клавишу «P» в английской раскладке.
5. Игрок просит программу начать игру, нажав клавишу «P» в английской раскладке.
6. Программа отрисовывает город, утку и трубы.
7. Игрок управляет уткой, нажимая клавишу «Пробел», стараясь пролететь между труб.
8. Программа мешает игроку, строя новые трубы и двигая их в направлении утки.
9. Если игрок проходит пару труб, то получает очко, иначе проигрывает.

**Изменять вид игры**

1. Игрок просит программу изменить утку и/или фон города, нажимая кнопку «Изменить фон города и/или утку».
2. Программа отображает список вариантов выбора утки и фона города.
3. Игрок просит изменить вид, нажимая кнопку «Выбрать» рядом с тем вариантом, который ему нужен.
4. Программа скрывает список вариантов.
5. При очередном запуске игры программа отображает фон города и утку с учетом пожеланий игрока.

**Смотреть рейтинг рекордов игроков**

1. После завершения очередной игры игрок просит программу показать таблицу рекордов, нажимая кнопку «Показать таблицу рекордов».
2. Программа отображает таблицу рекордов игроков.
3. После изучения таблицы игрок просит программу скрыть таблицу, повторно нажав на кнопку «Показать таблицу рекордов».
4. Программа скрывает таблицу рекордов.

**Изучать правила игры**

1. Игрок просит программу показать правила игры.
2. Программа высвечивает сообщение с описанием правил игры.
3. Игрок просит программу закрыть сообщение.
4. Программа закрывает сообщение.

# **4. РЕАЛИЗАЦИЯ**

При открытии HTML-документа игрок видит окно, изображенное на рисунке 2.

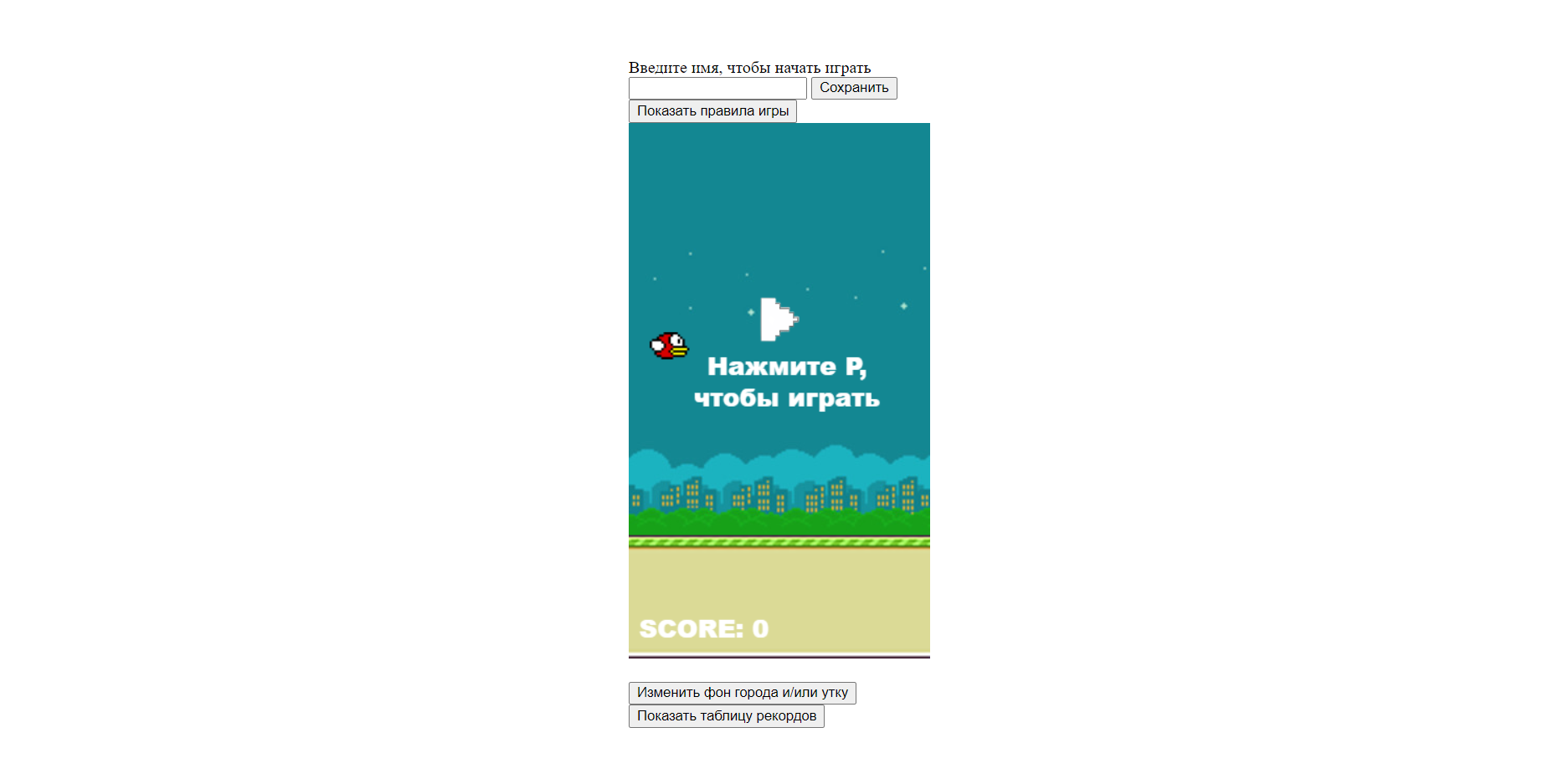


Рисунок 2 – Окно игры

Игра предлагает игроку посмотреть ознакомиться с правилами игры, для этого необходимо нажать на кнопку «Показать правила игры», которая изображена рисунке 3.

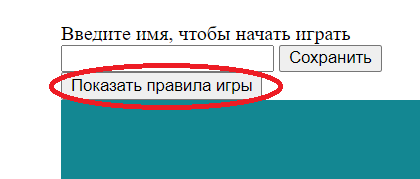


Рисунок 3 – Кнопка показа правил

После нажатия кнопки игра выведет сообщение, где будут указаны правила игры и её управление. Сообщение показано на рисунке 4.

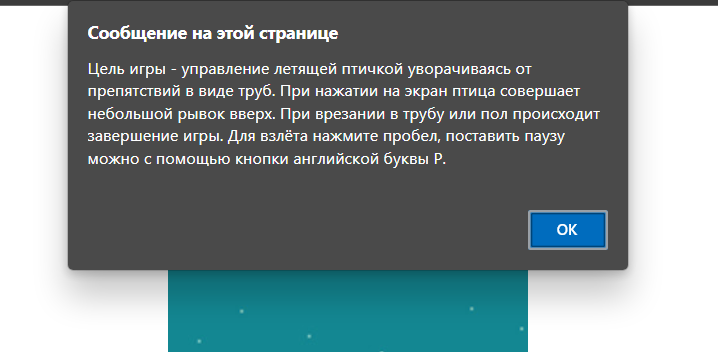


Рисунок 4 – Правила игры

Для начала игры необходимо ввести своё имя, чтобы позже программа смогла внести данные игрока. Ввод имени и его сохранение производятся в форме, обведенной на рисунке 5.

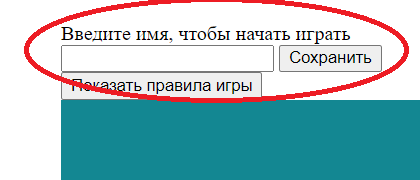


Рисунок 5 – Ввод имени игрока

После ввода имени игрок начинает игру, следуя правилам игры, нажатием кнопки английской буквы «P». Программа подсчитывает текущий счёт игрока и показывает его в левом нижнем углу игры. На рисунке 6 изображено поле счёта. Когда утка врежется в трубу или упадёт на землю или улетит за пределы окна игры, страница обновится, что означает конец игры.

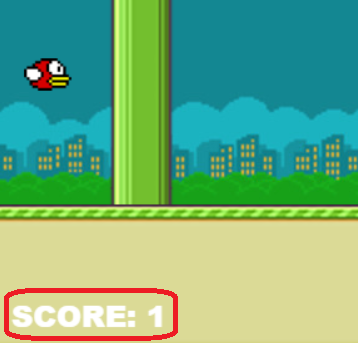


Рисунок 6 – Текущий счёт игрока

У игрока есть возможность изменить вид игры, изменив утку и/ или задний фон. Нажав кнопку «Изменить фон города и/или утку», игроку будут предложены варианты вида. Варианты вида утки изображены на рисунке 7, варианты вида фона города – на рисунке 8.

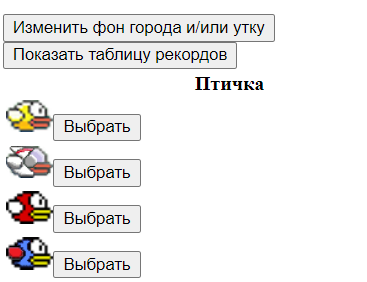


Рисунок 7 – Варианты смены вида утки

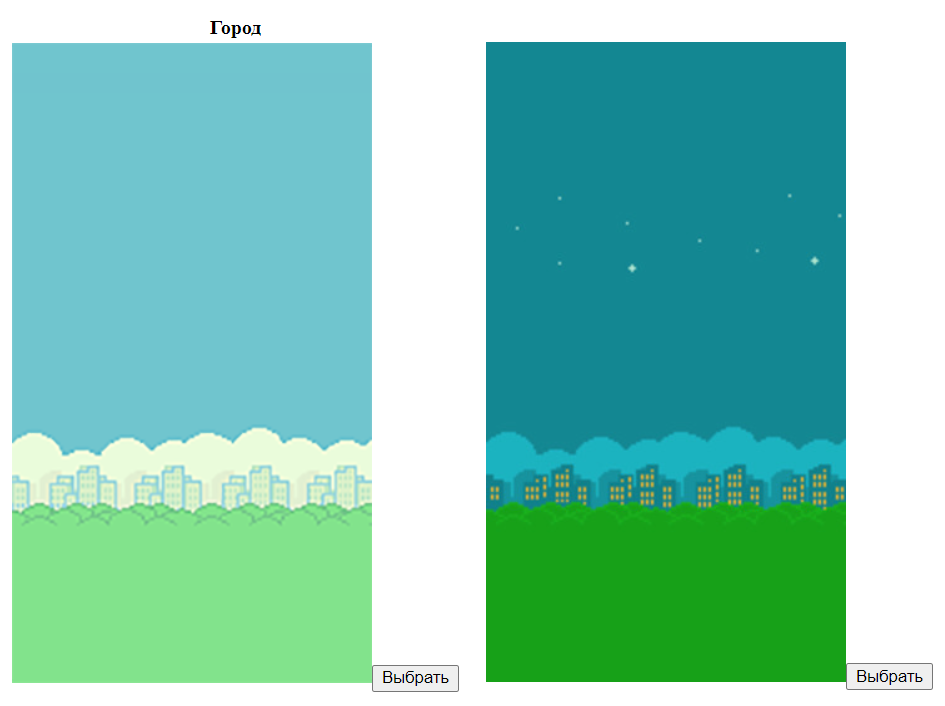


Рисунок 8 – Варианты смены заднего фона города

Для просмотра таблицы рекордов необходимо нажать кнопку «Показать таблицу рекордов», после нажатия появится таблица со списком лучших результатов игроков, сыгравших на вашем ПК. Пример таблицы рекордов представлен на рисунке 9.

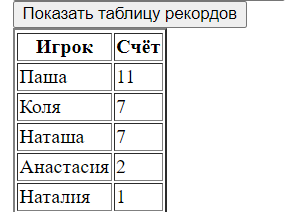


Рисунок 8 – Таблица рекордов игроков

# **ЗАКЛЮЧЕНИЕ**

В результате курсовой работы была разработана программа для игры в «Flappy Bird».

В ходе разработки была сделана диаграмма использования программы, были изучены алгоритм игры «Flappy Bird», принципы работы с веб-хранилищем, с анимацией в HTML-документе и элементом canvas, на основе которого реализована анимация игры посредством его изменения, были повышены навыки работы с языком программирования JavaScript.

# **СПИСОК ЛИТЕРАТУРЫ**

1. JavaScript.info / An Introduction to JavaScript URL: <https://javascript.info/intro> (дата обращения: 12.12.2022) – Режим доступа: свободный. – Текст: электронный.
2. App-s / Феномен Flappy Bird или история о том, как нам «нужно бооольше казуалок» URL: <http://app-s.ru/news/fenomen_flappy_bird_ili_istorija_o_tom_kak_nam_nuzhno_booolshe_kazualok/2014-03-15-5772> (дата обращения: 12.12.2022) – Режим доступа: свободный. – Текст: электронный.
3. Tjournal: журнал / Полёт птице-человека URL: <https://tjournal.ru/flood/50059-flappy-birdman> (дата обращения: 12.12.2022) – Режим доступа: свободный. – Текст: электронный.

**ПРИЛОЖЕНИЕ А**

Код игры «FlappyBird»

Const doc = document;

let canvas = doc.getElementById("canvas");

let context = canvas.getContext("2d");

let br = doc.createElement("br");

let input = doc.getElementById("username");

input.value = localStorage.getItem("user");

let saveName = doc.getElementById("saveButton");

let showRules = doc.getElementById("showRules");

let changeView = doc.createElement("button");

changeView.innerHTML = "Изменить фон города и/или утку";

let showRecords = doc.createElement("button");

showRecords.innerHTML = "Показать таблицу рекордов";

doc.body.append(br, changeView);

br = doc.createElement("br");

doc.body.append(br, showRecords);

let arrBg = ["Image/bg.png", "Image/bg\_night.png"];

let arrBirds = ["Image/bird.png", "Image/roboduck.png", "Image/red\_bird.png", "Image/blue\_bird.png"];

saveName.onclick = function() {

let label = doc.getElementById("enterName");

user.name = input.value;

localStorage.setItem("user", user.name);

if (user.name !== "") {

input.remove();

saveName.remove();

label.remove();

isNameEntered = true;

}

}

showRules.onclick = function() {

alert("Цель игры - управлять уткой, уворачиваясь от препятствий в виде труб и стараясь не задеть верхнюю и нижнюю границу игрового поля. При врезании в трубу, верхнюю или нижнюю границу игрового поля происходит завершение игры. При нажатии клавиши \"Пробел\" утка совершает небольшой рывок вверх, поставить паузу можно с помощью клавиши \"P\" в английской раскладке. ");

}

changeView.onclick = function() {

if (doc.getElementById("view")) {

doc.getElementById("view").remove();

} else {

createTableOfSkins(arrBirds, arrBg);

}

}

showRecords.onclick = function() {

if (doc.getElementById("records")) {

doc.getElementById("records").remove();

} else {

createTableOfRecords(scores);

}

}

function createTableOfSkins(arrBirds, arrBg) {

if (!doc.getElementById("view")) {

let table = doc.createElement("table");

table.setAttribute("id", "view");

let tr = doc.createElement("tr");

let th = doc.createElement("th");

th.innerHTML = "Утка";

tr.append(th);

table.append(tr);

for (let i = 0; i < arrBirds.length; i++) {

tr = doc.createElement("tr");

let td = doc.createElement("td");

let img = doc.createElement("img");

img.src = arrBirds[i];

td.append(img);

let changeBird = doc.createElement("button");

changeBird.innerHTML = "Выбрать";

changeBird.setAttribute("id", i + "");

changeBird.onclick = function() {

bird.src = arrBirds[changeBird.id];

localStorage.setItem('bird', JSON.stringify(bird.src));

table.remove();

fg.onload = draw;

}

td.append(changeBird);

tr.append(td);

table.append(tr);

}

tr = doc.createElement("tr");

th = doc.createElement("th");

th.innerHTML = "Город";

tr.append(th);

table.append(tr);

for (let i = 0; i < arrBg.length; i++) {

tr = doc.createElement("tr");

let td = doc.createElement("td");

let img = doc.createElement("img");

img.src = arrBg[i];

td.append(img);

let changeBg = doc.createElement("button");

changeBg.innerHTML = "Выбрать";

changeBg.setAttribute("id", i + "");

changeBg.onclick = function() {

bg.src = arrBg[changeBg.id];

localStorage.setItem('bg', JSON.stringify(bg.src));

table.remove();

fg.onload = draw;

}

td.append(changeBg);

tr.append(td);

table.append(tr);

}

doc.body.append(table);

}

}

function createTableOfRecords(scores) {

if (scores.length === 0) {

showRecords.remove();

let label = doc.createElement("label");

label.innerHTML = "Нет рекордов. Сыграйте, чтобы быть первым!";

doc.body.append(label);

}

if (!doc.getElementById("records")

&& scores.length !== 0) {

let table = doc.createElement("table");

table.setAttribute("id", "records");

table.setAttribute("border", "2");

let tr = doc.createElement("tr");

let th = doc.createElement("th");

th.innerHTML = "Игрок";

tr.append(th);

th = doc.createElement("th");

th.innerHTML = "Счёт";

tr.append(th);

table.append(tr);

for (let i = 0; i < scores.length; i++) {

tr = doc.createElement("tr");

let td = doc.createElement("td");

td.innerHTML = scores[i].name;

tr.append(td);

td = doc.createElement("td");

td.innerHTML = scores[i].score;

tr.append(td);

table.append(tr);

}

doc.body.append(table);

}

}

let bg = new Image();

let bird = new Image();

let down = new Image();

let up = new Image();

let fg = new Image();

let play = new Image();

let birdId = localStorage.getItem("bird");

let bgId = localStorage.getItem("bg");

bg.src = JSON.parse(bgId) || "Image/bg.png";

fg.src = "Image/fg.png";

bird.src = JSON.parse(birdId) || "Image/bird.png";

up.src = "Image/up.png";

down.src = "Image/down.png";

play.src = "Image/play.png";

let diff = 75;

let xBird = 20;

let yBird = 200;

let gravity = 1;

let user = {

name: 0,

score: 0

}

let score = 0;

let scoresId = localStorage.getItem("scores");

let scores = JSON.parse(scoresId) || [];

let isPause = true;

let isNameEntered = false;

let animation = 0;

let barriers = [];

barriers[0] = {

xBar : canvas.width,

yBar : Math.floor(Math.random() \* up.height) - up.height

};

function drawBarriers() {

for (let i = 0; i < barriers.length; i++) {

context.drawImage(up,barriers[i].xBar, barriers[i].yBar);

context.drawImage(down,barriers[i].xBar, barriers[i].yBar + up.height + diff);

barriers[i].xBar--;

if (barriers[i].xBar == 60) {

barriers.push({

xBar : canvas.width,

yBar : Math.floor(Math.random() \* up.height) - up.height

});

}

if (barriers[i].xBar == 2) {

user.score++;

}

if ((xBird + bird.width >= barriers[i].xBar && xBird <= barriers[i].xBar + up.width)

&& (yBird <= barriers[i].yBar + up.height || yBird + bird.height >= barriers[i].yBar + up.height + diff)

|| (yBird + bird.height >= canvas.height - fg.height || yBird + bird.height <= 0)) {

scores.push(user);

for (let i = 0; i < scores.length - 1; i++) {

if (scores.at(-1).name === scores[i].name) {

if (scores.at(-1).score > scores[i].score) {

scores[i].score = scores.at(-1).score;

scores.pop();

break;

} else if (scores.at(-1).score <= scores[i].score){

scores.pop();

break;

}

}

}

for (let i = 0; i < scores.length - 1; i++) {

if (scores[i].score <= scores[i + 1].score) {

let temp = scores[i];

scores[i] = scores[i + 1];

scores[i + 1] = temp;

}

}

localStorage.setItem("scores", JSON.stringify(scores));

location.reload();

}

}

}

function draw() {

context.drawImage(bg, 0, 0);

drawBarriers();

context.drawImage(fg, 0, canvas.height - fg.height);

context.drawImage(bird, xBird, yBird);

context.fillStyle = "white";

context.font = "24px Arial Black";

context.fillText("SCORE: " + user.score, 10, canvas.height - 20);

yBird += gravity;

if (isPause === false) {

animation = requestAnimationFrame(draw);

} else {

context.drawImage(play, 100, 150);

context.fillText("Нажмите P,", 75, 150 + play.height + 15);

context.fillText("чтобы играть", 62, 150 + play.height + 45);

}

}

fg.onload = draw;

function fly() {

if (isPause === false) {

yBird -= 25;

}

}

function toPause() {

if (isNameEntered) {

if (isPause == false) {

cancelAnimationFrame(animation);

context.drawImage(play, 100, 150);

context.fillText("Нажмите P,", 75, 150 + play.height + 15);

context.fillText("чтобы продолжить", 20, 150 + play.height + 45);

context.fillText("играть", 100, 150 + play.height + 75);

isPause = true;

} else if(isPause == true) {

animation = requestAnimationFrame(draw);

isPause = false;

}

}

}

doc.addEventListener("keydown",

function (e) {

if (e.code == "Space") {

fly();

} else if (e.code == "KeyP") {

toPause();

}

});