**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение   
высшего образования

«Национальный исследовательский Томский политехнический университет»

|  |  |
| --- | --- |
| Школа/Филиал | ИШИТР |
| Обеспечивающее подразделение | Отделение Информационных технологий |
| Направление подготовки / специальность | Информатика и вычислительная техника 09.03.01 |
| Образовательная программа (направленность (профиль)) | Программирование вычислительных и телекоммуникационных систем |
| Специализация | Программирование вычислительных систем |

**ОТЧЕТ О ПРАКТИКЕ**

|  |  |
| --- | --- |
| Вид практики | Учебная практика |
| Тип практики | Технологическая (Проектно-Технологическая) практика |
| Место практики | отделение информационных технологий ИШИТР ТПУ, г. Томск |

|  |  |
| --- | --- |
| Выполнил обучающийся | Назаров Матвей Владимирович |
| Группа | 8В12 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись обучающегося)

Руководитель практикиТПУ:

к.т.н., доцент ОИТ ИШИТР Попов Владимир Николаевич

Дата проверки «\_\_\_»\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

Допустить / не допустить к защите

Подпись \_\_\_\_\_\_\_\_\_\_\_\_

Итоговая оценка по практике4

(традиционная оценка, балл)

Томск 2023

**Индивидуальное задание на практику**

Основная среда программирования для учебной практики – MS Visual Studio C++ или C# в операционной системе Windows (возможны другие языки – согласовать с преподавателем!).

Минимальные требования к программам:

* Графический режим.
* Меню.
* Таблица рекордов (для игр) из 10-15 лучших – в файле.
* Помощь по организации управления процессом и описание цели/смысла происходящего в программе.

Добиться «носимости» программы. То есть должна быть возможность скопировать программу на любой неподготовленный компьютер и работать с ней без дополнительной настройки.

Вариант-8 Пасьянс «Косынка».

**Введение**

Цель практики – это написание определенной игры, исходя из варианта. Требования к игре, заранее известны. Первым шагом в реализации цели является изучение языка программирования, на котором будет написана игра. Для этого необходимо пройти соответствующий курс или выполнить самостоятельное изучение.

Затем следует определиться с инструментами разработки и выбрать наиболее подходящие для данной игры. Это может включать в себя использование определенных фреймворков, библиотек и программных пакетов.

Далее необходимо разработать дизайн и архитектуру игры, включая графику, звуковые эффекты и механику игрового процесса. После этого можно приступать к написанию кода и тестированию игры, а затем оптимизации и доработке до полной готовности.

Наконец, необходимо будет протестировать игру на различных устройствах и платформах, а затем выпустить ее на рынок.

**Содержательная часть**

**Язык программирования**

Для разработки моей игры, я выбрал язык программирования JavaScript с использованием библиотеки JQuery для облегчения процесса написания скриптов.

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который широко используется для разработки интерактивных веб-страниц и приложений. Он позволяет добавлять динамическое поведение на веб-сайты, что позволяет им реагировать на действия пользователя, изменять содержимое страницы без перезагрузки и обеспечивать взаимодействие с сервером для загрузки данных асинхронно.

**Среда разработки**

Разработки велась в текстовом редакторе Sublime Text 3.

Sublime Text предоставляет мощный и гибкий редактор кода с широкими возможностями для подсветки синтаксиса JavaScript, автодополнения, горячих клавиш и расширений для улучшения производительности. Благодаря его легковесности и отзывчивости, Sublime Text 3 стал отличным выбором для разработки на языке JavaScript, обеспечивая комфортное и продуктивное рабочее окружение.

**Проект**

**Начнем обзор моего проекта с главного меню.**

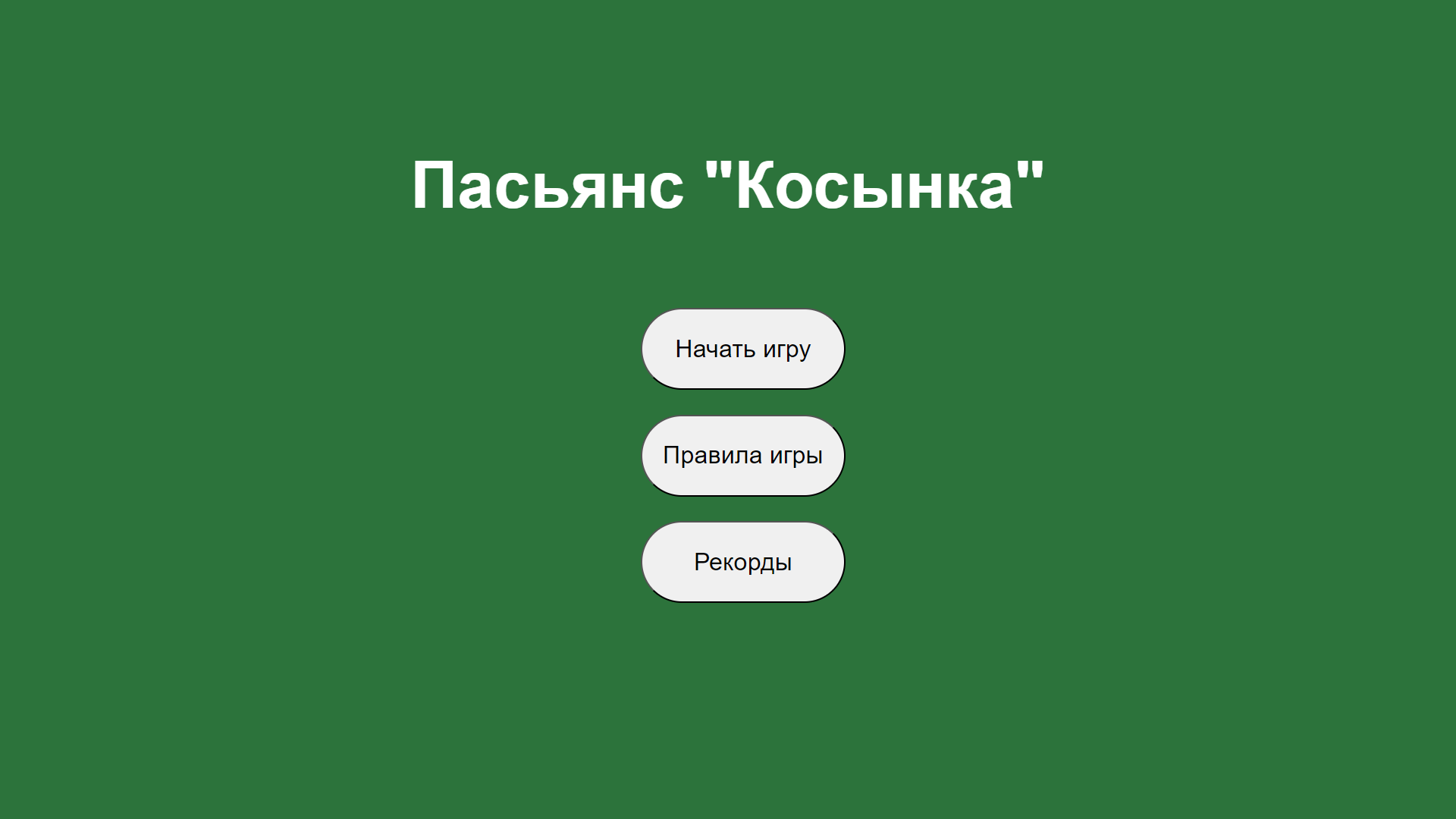


Рис.1 «Главное меню»

В верхней части главного меню находится название моего проекта – Пасьянс «Косынка». Эта игра является одной из самых известных и популярных вариаций карточной игры пасьянс.

В меню игры находятся 3 кнопки: Начать игру, Правила игры и Рекорды.

Первая кнопка отвечает за переход на страницу с игрой.

Вторая кнопка ведет на страницу с правилами игры. Здесь не знающий игрок может получить полную информацию о правилах игры.

Третья кнопка открывает страницу с рекордами, где игрок может увидеть таблицу с лучшими результатами.

**Страница с правилами игры**

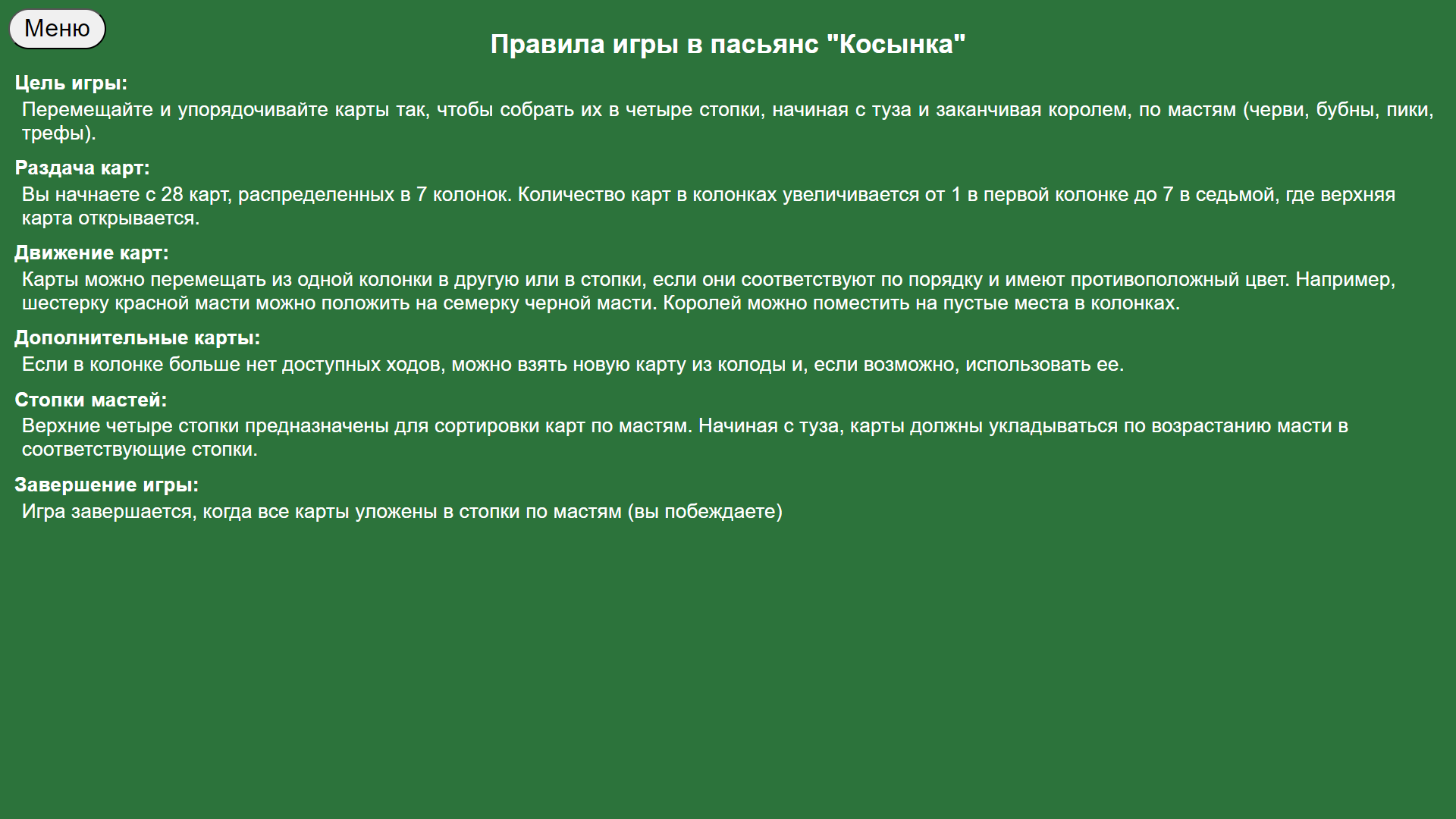


Рис.2 «Правила игры»

На странице "Правила игры" размещается приветственная надпись, которая знакомит игрока с целью игры и управлением. Для возврата в главное меню необходимо просто нажать на соответствующую кнопку, которая находится в левом верхнем углу экрана.

**Страница с рекордами**

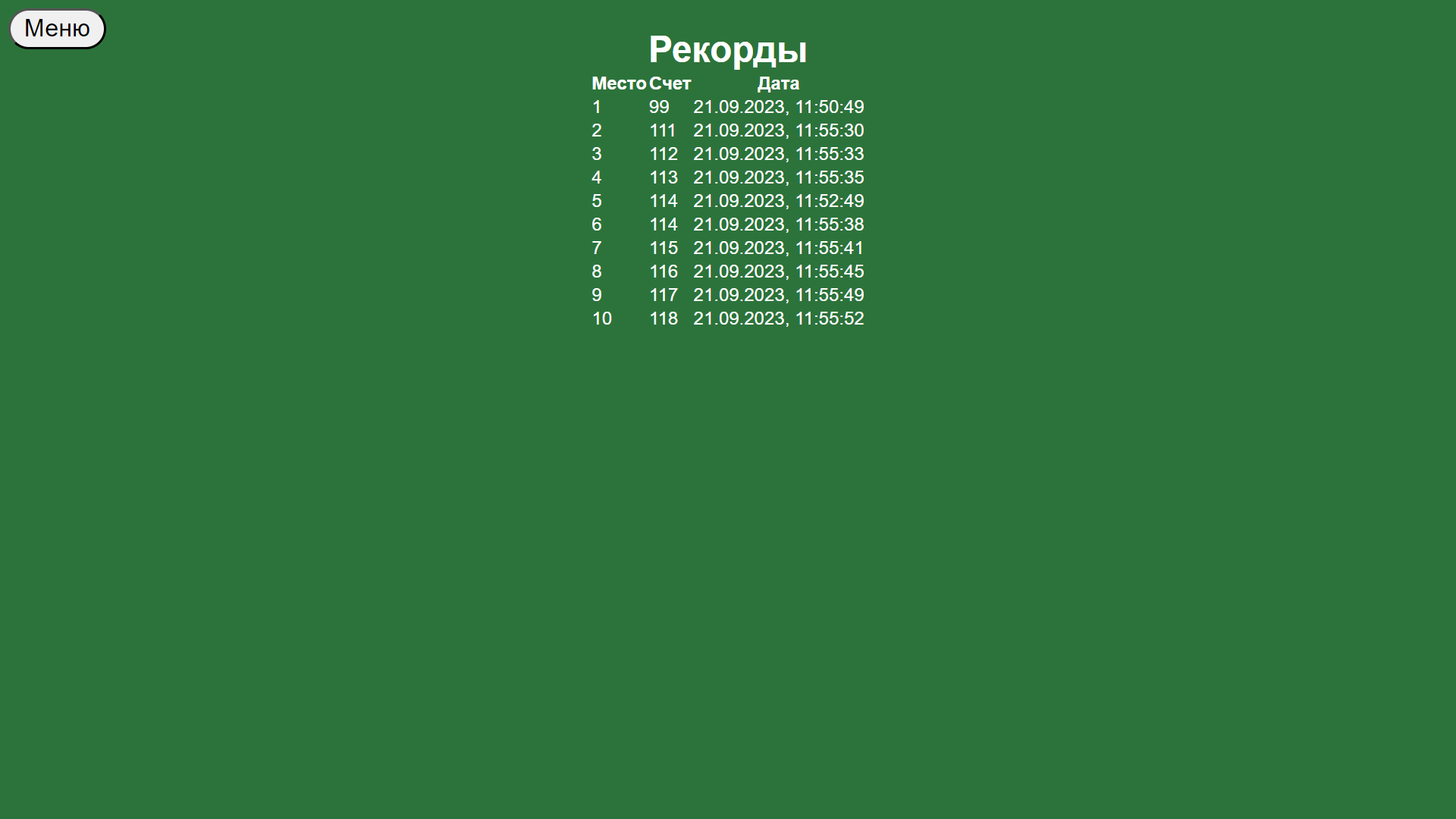


Рис.3 «Рекорды»

На странице с рекордами отображаются выигрыши с наименьшим количеством затраченных ходов. Для возврата в главное меню необходимо просто нажать на соответствующую кнопку, которая находится в левом верхнем углу экрана.

**Страница с игрой**

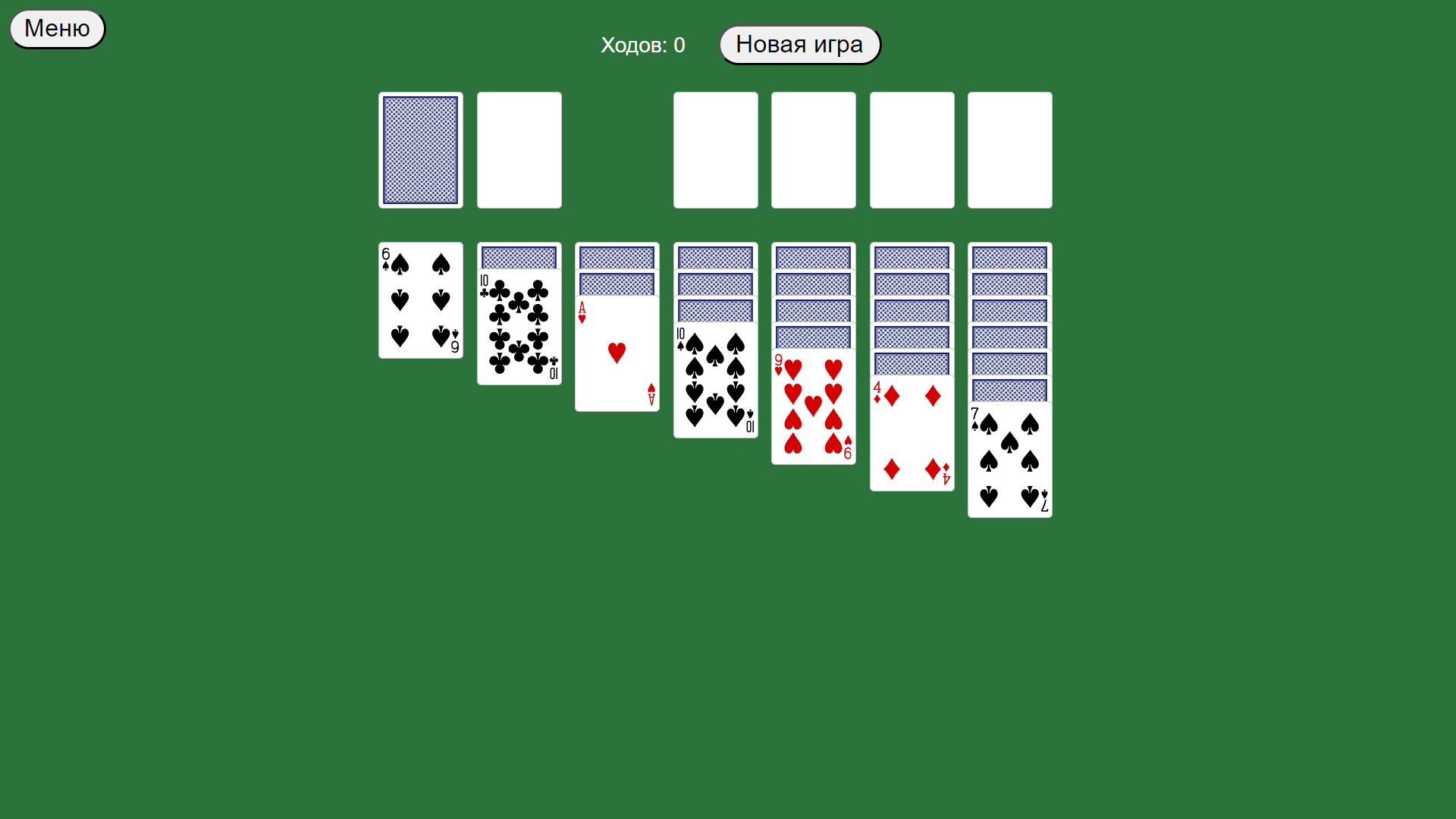


Рис.4 «Игра»

На странице находится игровое поле. На игровом поле находится три игровые зоны. Первая зона — это колода. Именно из нее игрок будет брать карты, которые сможет использовать для того, чтобы собрать пасьянс. Вторая зона — это четыре пустых основания. Здесь игрок будет складывать готовые блоки карт. Для победы они должны быть полными. Третья зона — это игровой блок. Он состоит из семи стопок карт, одна из которых лежит рубашкой вниз. В первой стопке лежит только одна эта карта, во второй под ней появляется одна карта, положенная рубашкой вверх, в третьей — две и так далее. Также отображается количество сделанных ходов и кнопка для начала новой игры. Для возврата в главное меню необходимо просто нажать на соответствующую кнопку, которая находится в левом верхнем углу экрана.

Так как движок Unity, обеспечивает кроссплатформенность проекту.

**Заключение**

В процессе выполнения практики был получен огромный опыт работы с языком программирования JavaScript и умения работы в текстовом редакторе Sublime Text 3. В итоге практики была создана игра Пасьянс «Косынка» и были выполнены все поставленные задачи:

* Графический режим.
* Меню.
* Таблица рекордов (для игр) из 10-15 лучших – в файле.
* Помощь по организации управления процессом и описание цели/смысла происходящего в программе.

Во время выполнения данных задач была написана логика игры, были сверстаны веб-страницы с меню, игровым полем, правилами и таблицей рекордов. Были написаны скрипты на языке программирования JavaScript с использованием библиотеки JQuery для работы всех объектов и их взаимодействия.

**Перечень информационных источников**

1. [https://пасьянсик.рф/pasyans-kosinka.php](https://nulltale.itch.io/white-scape) - Правила игры в пасьянс Косынка
2. [https://coderlessons.com/tutorials/raznoe/vyuchit-vozvyshennyi-tekst/sublime-text-tutorial](%20https://coderlessons.com/tutorials/raznoe/vyuchit-vozvyshennyi-tekst/sublime-text-tutorial) - Sublime Text Tutorial
3. <https://learn.javascript.ru/> - Современный учебник JavaScript
4. <https://habr.com/ru/articles/38208/> - jQuery для начинающих
5. [https://www.schoolsw3.com/css/index.php](%20https://www.schoolsw3.com/css/index.php) - Css учебник
6. <https://www.schoolsw3.com/html/index.php> - HTML учебник
7. [https://youtu.be/n\_UbAOfGFag](%20https://youtu.be/n_UbAOfGFag) - Создание карточного пасьянса "Косынка" для Javascript
8. <https://stackoverflow.com/questions/6552333/how-would-i-go-about-making-a-local-highscore-table-for-my-gamejavascript> - How would I go about making a local highscore table for my game?(Javascript)
9. <https://github.com/Siarhei-Chaptsau/Match-match-game> - Match-Match Game
10. <https://github.com/hews/card-starter> - Card Starter

**Приложения**

**menuPage.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title class="title">Меню</title>

<link rel="stylesheet" href="css/menuPage.css">

</head>

<body>

<div class="wrap">

<div class="head"><h1 align="center">Пасьянс "Косынка"</h1></div>

<form action="gamePage.html">

<button class="button">Начать игру</button>

</form>

<form action="rulesPage.html">

<button class="button">Правила игры</button>

</form>

<form action="recordsPage.html">

<button class="button">Рекорды</button>

</form>

</div>

</body>

</html>

**rulePage.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Правила игры</title>

<link rel="stylesheet" href="css/rulesPage.css">

</head>

<body>

<div class="head"><h1 align="center">Правила игры в пасьянс "Косынка"</h1></div>

<div class="head2"><h2 align="left">Цель игры:</h2></div>

<div class="text"><p align="justify">Перемещайте и упорядочивайте карты так, чтобы собрать их в четыре стопки, начиная с туза и заканчивая королем, по мастям (черви, бубны, пики, трефы).</p></div>

<div class="head2"><h2 align="left">Раздача карт:</h2></div>

<div class="text"><p align="justify"></p>Вы начнаете с 28 карт, распределенных в 7 колонок. Количество карт в колонках увеличивается от 1 в первой колонке до 7 в седьмой, где верхняя карта открывается.</div>

<div class="head2"><h2 align="left">Движение карт:</h2></div>

<div class="text"><p align="justify"></p>Карты можно перемещать из одной колонки в другую или в стопки, если они соответствуют по порядку и имеют противоположный цвет. Например, шестерку красной масти можно положить на семерку черной масти. Королей можно поместить на пустые места в колонках.</div>

<div class="head2"><h2 align="left">Дополнительные карты:</h2></div>

<div class="text"><p align="justify"></p>Если в колонке больше нет доступных ходов, можно взять новую карту из колоды и, если возможно, использовать ее.</div>

<div class="head2"><h2 align="left">Стопки мастей:</h2></div>

<div class="text"><p align="justify"></p>Верхние четыре стопки предназначены для сортировки карт по мастям. Начиная с туза, карты должны укладываться по возрастанию масти в соответствующие стопки.</div>

<div class="head2"><h2 align="left">Завершение игры:</h2></div>

<div class="text"><p align="justify"></p>Игра завершается, когда все карты уложены в стопки по мастям (вы побеждаете)</div>

<form action="menuPage.html">

<button class="button">Меню</button>

</form>

</body>

</html>

**recordsPage.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Рекорды</title>

<link rel="stylesheet" href="css/recordsPage.css">

</head>

<body>

<div class="head"><h1 align="center">Рекорды</h1></div>

<div class="recordsTable">

<table id="leaderboard">

<thead>

<tr>

<th>Место</th>

<th>Счет</th>

<th>Дата</th>

</tr>

</thead>

<tbody>

</tbody>

</table>

</div>

<script src="js/table.js"></script>

<form action="menuPage.html">

<button class="button">Меню</button>

</form>

</body>

</html>

**gamePage.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Косынка</title>

<link rel="stylesheet" type="text/css" href="css/cardstarter.css">

<link rel="stylesheet" type="text/css" href="css/gamePage.css">

</head>

<body>

<div class="wrapper">

<div class="topPanel">

<span class="moveCounter">Ходов: 0</span>

<button class="reset">Новая игра</button>

</div>

<div class="winMessage"></div>

<table class="topPiles">

<tr>

<td class="deck card"></td>

<td class="deckPile card"></td>

<td class="empty card"></td>

<td class="finalPile1 card"></td>

<td class="finalPile2 card"></td>

<td class="finalPile3 card"></td>

<td class="finalPile4 card"></td>

</tr>

</table>

<br><br><br><br><br>

<table class="boardPile">

<tr class="row1">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row2">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row3">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row4">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row5">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row6">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row7">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row8">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row9">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row10">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row11">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

<tr class="row12">

<td class="boardPile1 card"></td>

<td class="boardPile2 card"></td>

<td class="boardPile3 card"></td>

<td class="boardPile4 card"></td>

<td class="boardPile5 card"></td>

<td class="boardPile6 card"></td>

<td class="boardPile7 card"></td>

</tr>

</table>

</div>

<script src="js/jquery-3.7.0.js"></script>

<script src="js/game.js"></script>

<form action="menuPage.html">

<button class="button">Меню</button>

</form>

</body>

</html>

**menuPage.css**

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

background-color: #2c733b;

color: #ffffff;

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

}

form {

text-align: center;

}

.head {

display: block;

position: relative;

bottom: 80px;

right:20px;

font-size: 4vmin;

}

.wrap {

display: flex;

position: relative;

bottom:5px;

left:20px;

flex-direction: column;

justify-content: center;

gap: 3vmin;

}

.button {

height: 10vmin;

width: 25vmin;

border-radius: 10vmin;

font-size: 3vmin;

cursor: pointer;

}

**rulePage.css**

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

body {

justify-content: center;

align-items: center;

height: 100vh;

background-color: #2c733b;

color: #ffffff;

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

overflow-y: hidden;

}

.button {

height: 5vmin;

width: 12vmin;

border-radius: 5vmin;

font-size: 3vmin;

cursor: pointer;

position: absolute; top: 1vmin; left: 1vmin;

}

.head {

margin-top: 2%;

}

.head2 {

margin-top: 0.8%;

margin-left: 1%;

}

.text {

margin-top: 0.2%;

margin-left: 1.5%;

margin-right: 1.5%;

font-size: 18pt;

}

**recordsPage.css**

\* {

padding: 0;

margin: 0;

box-sizing: border-box;

}

body {

justify-content: center;

align-items: center;

height: 100vh;

background-color: #2c733b;

color: #ffffff;

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

overflow-y: hidden;

}

.recordsTable {

display: flex;

justify-content: center;

align-items: center;

}

.button {

height: 5vmin;

width: 12vmin;

border-radius: 5vmin;

font-size: 3vmin;

cursor: pointer;

position: absolute; top: 1vmin; left: 1vmin;

}

.head {

margin-top: 2%;

}

.head2 {

margin-top: 0.8%;

margin-left: 1%;

}

.text {

margin-top: 0.2%;

margin-left: 1.5%;

margin-right: 1.5%;

font-size: 18pt;

}

**gamePage.css**

body {

background-color: #2c733b;

font-size: 16px;

min-width: 540px;

font-family: sans-serif;

color: white;

}

span {

display: inline-block;

padding: 0px 20px;

}

.button {

height: 5vmin;

width: 12vmin;

border-radius: 5vmin;

font-size: 3vmin;

cursor: pointer;

position: absolute; top: 1vmin; left: 1vmin;

font-family: sans-serif;

}

.reset {

height: 5vmin;

width: 20vmin;

border-radius: 5vmin;

font-size: 3vmin;

cursor: pointer;

}

.card {

margin: 5px 5px -80px 5px;

}

.wrapper {

margin: 0 auto;

width: 540px;

}

.empty {

opacity: 0;

}

.winMessage {

text-align: center;

font-size: 4vmin;

}

.topPanel {

text-align: center;

padding: 10px;

}

**cardstarter.css**

.card {

box-sizing: border-box;

display: inline-block;

cursor: pointer;

width: 4.0em;

height: 5.5em;

background-color: #fff;

border-radius: 0.20em;

border: 1px solid #c3c3c3;

background-size: contain;

}

.card.dA{

background-image: url("../images/diamonds/diamonds-A.svg"); }

.card.dK{

background-image: url("../images/diamonds/diamonds-K.svg"); }

.card.dQ{

background-image: url("../images/diamonds/diamonds-Q.svg"); }

.card.dJ{

background-image: url("../images/diamonds/diamonds-J.svg"); }

.card.d10{

background-image: url("../images/diamonds/diamonds-r10.svg"); }

.card.d09{

background-image: url("../images/diamonds/diamonds-r09.svg"); }

.card.d08{

background-image: url("../images/diamonds/diamonds-r08.svg"); }

.card.d07{

background-image: url("../images/diamonds/diamonds-r07.svg"); }

.card.d06{

background-image: url("../images/diamonds/diamonds-r06.svg"); }

.card.d05{

background-image: url("../images/diamonds/diamonds-r05.svg"); }

.card.d04{

background-image: url("../images/diamonds/diamonds-r04.svg"); }

.card.d03{

background-image: url("../images/diamonds/diamonds-r03.svg"); }

.card.d02{

background-image: url("../images/diamonds/diamonds-r02.svg"); }

.card.hA{

background-image: url("../images/hearts/hearts-A.svg"); }

.card.hK{

background-image: url("../images/hearts/hearts-K.svg"); }

.card.hQ{

background-image: url("../images/hearts/hearts-Q.svg"); }

.card.hJ{

background-image: url("../images/hearts/hearts-J.svg"); }

.card.h10{

background-image: url("../images/hearts/hearts-r10.svg"); }

.card.h09{

background-image: url("../images/hearts/hearts-r09.svg"); }

.card.h08{

background-image: url("../images/hearts/hearts-r08.svg"); }

.card.h07{

background-image: url("../images/hearts/hearts-r07.svg"); }

.card.h06{

background-image: url("../images/hearts/hearts-r06.svg"); }

.card.h05{

background-image: url("../images/hearts/hearts-r05.svg"); }

.card.h04{

background-image: url("../images/hearts/hearts-r04.svg"); }

.card.h03{

background-image: url("../images/hearts/hearts-r03.svg"); }

.card.h02{

background-image: url("../images/hearts/hearts-r02.svg"); }

.card.sA{

background-image: url("../images/spades/spades-A.svg"); }

.card.sK{

background-image: url("../images/spades/spades-K.svg"); }

.card.sQ{

background-image: url("../images/spades/spades-Q.svg"); }

.card.sJ{

background-image: url("../images/spades/spades-J.svg"); }

.card.s10{

background-image: url("../images/spades/spades-r10.svg"); }

.card.s09{

background-image: url("../images/spades/spades-r09.svg"); }

.card.s08{

background-image: url("../images/spades/spades-r08.svg"); }

.card.s07{

background-image: url("../images/spades/spades-r07.svg"); }

.card.s06{

background-image: url("../images/spades/spades-r06.svg"); }

.card.s05{

background-image: url("../images/spades/spades-r05.svg"); }

.card.s04{

background-image: url("../images/spades/spades-r04.svg"); }

.card.s03{

background-image: url("../images/spades/spades-r03.svg"); }

.card.s02{

background-image: url("../images/spades/spades-r02.svg"); }

.card.cA{

background-image: url("../images/clubs/clubs-A.svg"); }

.card.cK{

background-image: url("../images/clubs/clubs-K.svg"); }

.card.cQ{

background-image: url("../images/clubs/clubs-Q.svg"); }

.card.cJ{

background-image: url("../images/clubs/clubs-J.svg"); }

.card.c10{

background-image: url("../images/clubs/clubs-r10.svg"); }

.card.c09{

background-image: url("../images/clubs/clubs-r09.svg"); }

.card.c08{

background-image: url("../images/clubs/clubs-r08.svg"); }

.card.c07{

background-image: url("../images/clubs/clubs-r07.svg"); }

.card.c06{

background-image: url("../images/clubs/clubs-r06.svg"); }

.card.c05{

background-image: url("../images/clubs/clubs-r05.svg"); }

.card.c04{

background-image: url("../images/clubs/clubs-r04.svg"); }

.card.c03{

background-image: url("../images/clubs/clubs-r03.svg"); }

.card.c02{

background-image: url("../images/clubs/clubs-r02.svg"); }

.card.back {

background-image: url("../images/backs/blue.svg"); }

**table.js**

function displayLeaderboard() {

const leaderboardTable = document.getElementById('leaderboard');

const leaderboardBody = leaderboardTable.getElementsByTagName('tbody')[0];

// Получаем результаты из localStorage

const results = JSON.parse(localStorage.getItem('gameResults')) || [];

// Очищаем таблицу перед заполнением

leaderboardBody.innerHTML = '';

// Заполняем таблицу данными из результатов

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

const result = results[i];

const row = leaderboardBody.insertRow(i);

const placeCell = row.insertCell(0);

// const playerNameCell = row.insertCell(1);

const scoreCell = row.insertCell(1);

const dateCell = row.insertCell(2);

placeCell.textContent = i + 1; // Место

// playerNameCell.textContent = result.playerName;

scoreCell.textContent = result.score;

dateCell.textContent = result.timestamp;

}

}

// Вызываем функцию для отображения таблицы рекордов

displayLeaderboard();

**game.js**

// все переменные

var staticCards = [];

var deck = [];

var tempDeck = [];

var deckPile = [];

var boardPile1 = [];

var boardPile2 = [];

var boardPile3 = [];

var boardPile4 = [];

var boardPile5 = [];

var boardPile6 = [];

var boardPile7 = [];

var finalPile1 = [];

var finalPile2 = [];

var finalPile3 = [];

var finalPile4 = [];

var firstPile = [];

var boardState1 = [];

var boardState2 = [];

var boardState3 = [];

var boardState4 = [];

var boardState5 = [];

var boardState6 = [];

var boardState7 = [];

var finalState1 = [];

var finalState2 = [];

var finalState3 = [];

var finalState4 = [];

var deckPileState = [];

var moves;

// Ace: 0, king: 12

// 0-12: hearts, 13-25:diamonds, 26-38:spades, 39-52:clubs

//исходное состояние

function init() {

initializeVariables();

resetClasses();

staticCards = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]

deck = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51];

deck = makeRandom(deck);

makePile(boardPile1, 1);

makePile(boardPile2, 2);

makePile(boardPile3, 3);

makePile(boardPile4, 4);

makePile(boardPile5, 5);

makePile(boardPile6, 6);

makePile(boardPile7, 7);

$('.winMessage').text('');

render();

}

function initializeVariables() {

moves = 0;

tempDeck.length = 0;

deckPile.length = 0;

boardPile1.length = 0;

boardPile2.length = 0;

boardPile3.length = 0;

boardPile4.length = 0;

boardPile5.length = 0;

boardPile6.length = 0;

boardPile7.length = 0;

finalPile1.length = 0;

finalPile2.length = 0;

finalPile3.length = 0;

finalPile4.length = 0;

firstPile.length = 0;

boardState1.length = 0;

boardState2.length = 0;

boardState3.length = 0;

boardState4.length = 0;

boardState5.length = 0;

boardState6.length = 0;

boardState7.length = 0;

finalState1.length = 0;

finalState2.length = 0;

finalState3.length = 0;

finalState4.length = 0;

deckPileState.length = 0;

}

//сброс классов

function resetClasses() {

var rows = $('td');

for (var i = 0; i < rows.length; i++) {

$(rows[i]).removeClass('hA h01 h02 h03 h04 h05 h06 h07 h08 h09 h10 hJ hQ hK dA d01 d02 d03 d04 d05 d06 d07 d08 d09 d10 dJ dQ dK sA s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 sJ sQ sK cA c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 cJ cQ cK faceUp')

}

}

//проверка событий

$('.deck').on('click', function(evt) {

if (deck.length > 0) {

deckPile.unshift(deck.shift())

moves++

render();

}

else reloadDeck();

})

$('.deckPile').on('click', function(evt) {

if (deckPile.length > 0) {

firstPile.length = 0;

firstPile.unshift(deckPile[0]);

$(this).css('opacity', '.8')

}

})

$('.finalPile1, .finalPile2, .finalPile3, .finalPile4').on('click', function(evt) {

if (firstPile.length === 1) {

if ($(this).hasClass('finalPile1')) {

checkFinalMove(firstPile, finalPile1);

}

else if ($(this).hasClass('finalPile2')) {

checkFinalMove(firstPile, finalPile2);

}

else if ($(this).hasClass('finalPile3')) {

checkFinalMove(firstPile, finalPile3);

}

else if ($(this).hasClass('finalPile4')) {

checkFinalMove(firstPile, finalPile4);

}

else;

}

resetFirstPile();

})

$('button').on('click', function(evt) {

init();

})

//рандомайзер для игры

function makeRandom(arr) {

for (var i = arr.length - 1; i > 0; i--) {

var j = Math.floor(Math.random() \* (i + 1));

var temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

return arr;

}

//создание начальных стопок карт

function makePile(pile, num) {

for (var i = 0; i < num; i++)

pile.unshift(deck.shift());

}

//клик на пустые ячейки

function addEmptyClick() {

$('.empty').off();

$('.empty').on('click', function(evt) {

if (firstPile.length === 0) {

if ($(this).hasClass('boardPile1')) {

firstPile.unshift(boardPile1[0])

$(boardState1[boardPile1.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile2')) {

firstPile.unshift(boardPile2[0])

$(boardState2[boardPile2.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile3')) {

firstPile.unshift(boardPile3[0])

$(boardState3[boardPile3.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile4')) {

firstPile.unshift(boardPile4[0])

$(boardState4[boardPile4.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile5')) {

firstPile.unshift(boardPile5[0])

$(boardState5[boardPile5.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile6')) {

firstPile.unshift(boardPile6[0])

$(boardState6[boardPile6.length-1]).css('opacity', '.8')

}

else if ($(this).hasClass('boardPile7')) {

firstPile.unshift(boardPile7[0])

$(boardState7[boardPile7.length-1]).css('opacity', '.8')

}

}

else {

if ($(this).hasClass('boardPile1')) {

checkBoardMove(firstPile, boardPile1)

}

else if ($(this).hasClass('boardPile2')) {

checkBoardMove(firstPile, boardPile2)

}

else if ($(this).hasClass('boardPile3')) {

checkBoardMove(firstPile, boardPile3)

}

else if ($(this).hasClass('boardPile4')) {

checkBoardMove(firstPile, boardPile4)

}

else if ($(this).hasClass('boardPile5')) {

checkBoardMove(firstPile, boardPile5)

}

else if ($(this).hasClass('boardPile6')) {

checkBoardMove(firstPile, boardPile6)

}

else if ($(this).hasClass('boardPile7')) {

checkBoardMove(firstPile, boardPile7)

}

}

})

}

// Ace: 0, king: 12

// 0-12: hearts, 13-25:diamonds, 26-38:spades, 39-52:clubs

//проверка ходов на игровом блоке

function checkBoardMove(firstPile, secondPile) {

if (secondPile.length === 0) {

legal(firstPile, secondPile);

}

else if (firstPile[firstPile.length-1] === 12 || firstPile[firstPile.length-1] === 25 || firstPile[firstPile.length-1] === 38 || firstPile[firstPile.length-1] === 51 ) {

illegal();

}

else if (firstPile[firstPile.length-1] < 13) {

if (secondPile[0] === firstPile[firstPile.length-1] + 27 || secondPile[0] === firstPile[firstPile.length-1] + 40) {

legal(firstPile, secondPile);

}

else illegal()}

else if (firstPile[firstPile.length-1] < 26) {

if (secondPile[0] === firstPile[firstPile.length-1] + 14 || secondPile[0] === firstPile[firstPile.length-1] + 27) {

legal(firstPile, secondPile);

}

else illegal()}

else if (firstPile[firstPile.length-1] < 39) {

if (secondPile[0] === firstPile[firstPile.length-1] - 12 || secondPile[0] === firstPile[firstPile.length-1] - 25) {

legal(firstPile, secondPile);

}

else illegal()}

else if (firstPile[firstPile.length-1] < 52) {

if (secondPile[0] === firstPile[firstPile.length-1] - 25 || secondPile[0] === firstPile[firstPile.length-1] - 38) {

legal(firstPile, secondPile);

}

else illegal()}

}

//проверка ходов на основаниях

function checkFinalMove(firstPile, secondPile) {

if (firstPile.length === 1) {

if (firstPile[0] === staticCards[0] || firstPile[0] === staticCards[13] || firstPile[0] === staticCards[26] || firstPile[0] === staticCards[39]) {

if (secondPile.length === 0) {

legal(firstPile, secondPile);

}

else illegal();

}

else if (firstPile[0] === secondPile[0] + 1) {

legal(firstPile, secondPile)

}

else illegal();

}

else illegal();

}

//обработка доступных ходов

function legal(firstPile, secondPile) {

for (var i = firstPile.length-1; i >= 0; i--) {

secondPile.unshift(firstPile[i])

};

secondPile.unshift('placeholder');

switch (firstPile[0]) {

case deckPile[0]:

for (var i = firstPile.length; i > 0; i--)

{deckPile.shift()}

break;

case boardPile1[0]:

for (var i = firstPile.length; i > 0; i--)

{boardPile1.shift()}

break;

case boardPile2[0]:

for (var i = firstPile.length; i > 0; i--)

{boardPile2.shift()}

break;

case boardPile3[0]:

for (var i = firstPile.length; i > 0; i--)

{boardPile3.shift()}

break;

case boardPile4[0]:

for (var i = firstPile.length; i > 0; i--) {

boardPile4.shift()}

break;

case boardPile5[0]:

for (var i = firstPile.length; i > 0; i--) {

boardPile5.shift()}

break;

case boardPile6[0]:

for (var i = firstPile.length; i > 0; i--)

{boardPile6.shift()}

break;

case boardPile7[0]:

for (var i = firstPile.length; i > 0; i--)

{boardPile7.shift()}

break;

case finalPile1[0]:

for (var i = firstPile.length; i > 0; i--) {

finalPile1.shift()}

break;

case finalPile2[0]:

for (var i = firstPile.length; i > 0; i--) {

finalPile2.shift()}

break;

case finalPile3[0]:

for (var i = firstPile.length; i > 0; i--)

{finalPile3.shift()}

break;

case finalPile4[0]:

for (var i = firstPile.length; i > 0; i--)

{finalPile4.shift()}

break;

}

resetFirstPile();

secondPile.shift();

moves++;

render();

}

//возвращение поля в нормальное состояние после недоступного хода

function illegal() {

resetFirstPile();

render();

}

//переворачивает колоду на доску когда просмотрены все карты

function reloadDeck() {

if (deck.length === 0) {

for (var i = deckPile.length; i > 0; i--) {

deck.unshift(deckPile.shift());

}

}

}

//проверка победы

function checkWin() {

if (finalPile1[0] === staticCards[12] || finalPile1[0] === staticCards[25] || finalPile1[0] === staticCards[38] || finalPile1[0] === staticCards[51]) {

if (finalPile2[0] === staticCards[12] || finalPile2[0] === staticCards[25] || finalPile2[0] === staticCards[38] || finalPile2[0] === staticCards[51]) {

if (finalPile3[0] === staticCards[12] || finalPile3[0] === staticCards[25] || finalPile3[0] === staticCards[38] || finalPile3[0] === staticCards[51]) {

if (finalPile4[0] === staticCards[12] || finalPile4[0] === staticCards[25] || finalPile4[0] === staticCards[38] || finalPile4[0] === staticCards[51]) {

alert("Вы выиграли!\nВам потребовалось: " + moves + " ходов");

saveGameResult(moves);

}

}

}

}

else return;

}

// Функция для сохранения результатов игры

function saveGameResult(score) {

// Получаем текущий список результатов из localStorage (если он есть)

let results = JSON.parse(localStorage.getItem('gameResults')) || [];

// Создаем объект с результатами текущей игры

const gameResult = {

score: score,

timestamp: new Date().toLocaleString(), // Добавляем временную метку

};

// Добавляем результат текущей игры в список результатов

results.push(gameResult);

// Сортируем результаты по счету (от наименьшего к наибольшему)

results.sort((a, b) => a.score - b.score);

// Ограничиваем список результатов, чтобы оставить только, например, 10 лучших результатов

const maxResults = 10;

if (results.length > maxResults) {

results = results.slice(0, maxResults);

}

// Сохраняем обновленный список результатов в localStorage

localStorage.setItem('gameResults', JSON.stringify(results));

}

//отображение карт в основании и колоде

function renderTop() {

if (deck.length > 0) {

$('.deck').addClass('card').addClass('back');

}

else $(".deck").removeClass('back');

if (deckPile.length > 0) {

$('.deckPile').removeClass().addClass('deckPile').addClass('card')

deckPileState = $('deckPile')

renderCard('.deckPile', deckPile[0]);

}

else $('.deckPile').removeClass().addClass('deckPile').addClass('card');

if (finalPile1.length > 0) {

finalState1 = $('.finalPile1');

removeExtraClasses('.finalPile1');

renderCard(finalState1[0], finalPile1[0])

}

if (finalPile2.length > 0) {

finalState2 = $('.finalPile2');

removeExtraClasses('.finalPile2');

renderCard(finalState2[0], finalPile2[0])

}

if (finalPile3.length > 0) {

finalState3 = $('.finalPile3');

removeExtraClasses('.finalPile3');

renderCard(finalState3[0], finalPile3[0])

}

if (finalPile4.length > 0) {

finalState4 = $('.finalPile4');

removeExtraClasses('.finalPile4');

renderCard(finalState4[0], finalPile4[0])

}

}

//удаляет лишние классы из конечных стопок

function removeExtraClasses(pile) {

$(pile).removeClass('hA h01 h02 h03 h04 h05 h06 h07 h08 h09 h10 hJ hQ hK dA d01 d02 d03 d04 d05 d06 d07 d08 d09 d10 dJ dQ dK sA s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 sJ sQ sK cA c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 cJ cQ cK faceUp');

}

//отображение карт в игровом блоке

function renderPiles() {

if (boardPile1.length > 0) {

boardState1 = $('.boardPile1');

var difference = (1 + (boardState1.length - boardPile1.length));

for (var i = 0; i < boardState1.length; i++)

if (boardPile1[boardPile1.length-1] !== undefined) {

if (i >= boardPile1.length) {

renderCard(boardState1[i], boardPile1[i])

}

else {renderCard(boardState1[boardState1.length - difference - i], boardPile1[i])

}}}

else {$('.boardPile1').removeClass().addClass('boardPile1').addClass('card').addClass('faceUp');

boardState1 = $('.boardPile1');

for (var i = 1; i < boardState1.length; i++) {

renderCard(boardState1[i], boardPile1[i])

}}

if (boardPile2.length > 0) {

boardState2 = $('.boardPile2');

var difference = (1 + (boardState2.length - boardPile2.length));

for (var i = 0; i < boardState2.length; i++)

if (boardPile2[boardPile2.length-1] !== undefined) {

if (i >= boardPile2.length) {

renderCard(boardState2[i], boardPile2[i])

}

else {renderCard(boardState2[boardState2.length - difference - i], boardPile2[i])

}}}

else {$('.boardPile2').removeClass().addClass('boardPile2').addClass('card').addClass('faceUp');

boardState2 = $('.boardPile2');

for (var i = 1; i < boardState2.length; i++) {

renderCard(boardState2[i], boardPile2[i])

}}

if (boardPile3.length > 0) {

boardState3 = $('.boardPile3');

var difference = (1 + (boardState3.length - boardPile3.length));

for (var i = 0; i < boardState3.length; i++)

if (boardPile3[boardPile3.length-1] !== undefined) {

if (i >= boardPile3.length) {

renderCard(boardState3[i], boardPile3[i])

}

else {renderCard(boardState3[boardState3.length - difference - i], boardPile3[i])

}}}

else {$('.boardPile3').removeClass().addClass('boardPile3').addClass('card').addClass('faceUp');

boardState3 = $('.boardPile3');

for (var i = 1; i < boardState3.length; i++) {

renderCard(boardState3[i], boardPile3[i])

}}

if (boardPile4.length > 0) {

boardState4 = $('.boardPile4');

var difference = (1 + (boardState4.length - boardPile4.length));

for (var i = 0; i < boardState4.length; i++)

if (boardPile4[boardPile4.length-1] !== undefined) {

if (i >= boardPile4.length) {

renderCard(boardState4[i], boardPile4[i])

}

else {renderCard(boardState4[boardState4.length - difference - i], boardPile4[i])

}}}

else {$('.boardPile4').removeClass().addClass('boardPile4').addClass('card').addClass('faceUp');

boardState4 = $('.boardPile4');

for (var i = 1; i < boardState4.length; i++) {

renderCard(boardState4[i], boardPile4[i])

}}

if (boardPile5.length > 0) {

boardState5 = $('.boardPile5');

var difference = (1 + (boardState5.length - boardPile5.length));

for (var i = 0; i < boardState5.length; i++)

if (boardPile5[boardPile5.length-1] !== undefined) {

if (i >= boardPile5.length) {

renderCard(boardState5[i], boardPile5[i])

}

else {renderCard(boardState5[boardState5.length - difference - i], boardPile5[i])

}}}

else {$('.boardPile5').removeClass().addClass('boardPile5').addClass('card').addClass('faceUp');

boardState5 = $('.boardPile5');

for (var i = 1; i < boardState5.length; i++) {

renderCard(boardState5[i], boardPile5[i])

}}

if (boardPile6.length > 0) {

boardState6 = $('.boardPile6');

var difference = (1 + (boardState6.length - boardPile6.length));

for (var i = 0; i < boardState6.length; i++)

if (boardPile6[boardPile6.length-1] !== undefined) {

if (i >= boardPile6.length) {

renderCard(boardState6[i], boardPile6[i])

}

else {renderCard(boardState6[boardState6.length - difference - i], boardPile6[i])

}}}

else {$('.boardPile6').removeClass().addClass('boardPile6').addClass('card').addClass('faceUp');

boardState6 = $('.boardPile6');

for (var i = 1; i < boardState6.length; i++) {

renderCard(boardState6[i], boardPile6[i])

}}

if (boardPile7.length > 0) {

boardState7 = $('.boardPile7');

var difference = (1 + (boardState7.length - boardPile7.length));

for (var i = 0; i < boardState7.length; i++)

if (boardPile7[boardPile7.length-1] !== undefined) {

if (i >= boardPile7.length) {

renderCard(boardState7[i], boardPile7[i])

}

else {renderCard(boardState7[boardState7.length - difference - i], boardPile7[i])

}}}

else {$('.boardPile7').removeClass().addClass('boardPile7').addClass('card').addClass('faceUp');

boardState7 = $('.boardPile7');

for (var i = 1; i < boardState7.length; i++) {

renderCard(boardState7[i], boardPile7[i])

}}

};

//Переворачивает первую карту в каждой стопке

function flipFirstCard() {

if (boardPile1.length > 0); {

$(boardState1[boardPile1.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile2.length > 0); {

$(boardState2[boardPile2.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile3.length > 0); {

$(boardState3[boardPile3.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile4.length > 0); {

$(boardState4[boardPile4.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile5.length > 0); {

$(boardState5[boardPile5.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile6.length > 0); {

$(boardState6[boardPile6.length-1]).addClass('faceUp').removeClass('back')}

if (boardPile7.length > 0); {

$(boardState7[boardPile7.length-1]).addClass('faceUp').removeClass('back')}};

//переворачивает карты которые находятся под верхней в игровом блоке

function flipAllOtherCards() {

flipOtherCards(boardState1);

flipOtherCards(boardState2);

flipOtherCards(boardState3);

flipOtherCards(boardState4);

flipOtherCards(boardState5);

flipOtherCards(boardState6);

flipOtherCards(boardState7);

}

function flipOtherCards(arr) {

for (var i = 0; i < arr.length; i++) {

if ($(arr[i]).hasClass('faceUp')) {

$(arr[i+1]).addClass('faceUp')

}

}

}

//Если что-то на плате не имеет другого класса, то карта отображается как перевернутая карта

function renderFaceDowns() {

for (var i = 0; i < boardPile1.length; i++)

if ($(boardState1[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState1[i]).addClass('back')}

for (var i = 0; i < boardPile2.length; i++)

if ($(boardState2[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState2[i]).addClass('back')}

for (var i = 0; i < boardPile3.length; i++)

if ($(boardState3[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState3[i]).addClass('back')}

for (var i = 0; i < boardPile4.length; i++)

if ($(boardState4[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState4[i]).addClass('back')}

for (var i = 0; i < boardPile5.length; i++)

if ($(boardState5[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState5[i]).addClass('back')}

for (var i = 0; i < boardPile6.length; i++)

if ($(boardState6[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState6[i]).addClass('back')}

for (var i = 0; i < boardPile7.length; i++)

if ($(boardState7[i]).hasClass('faceUp' || 'empty' || 'back')) {}

else {$(boardState7[i]).addClass('back')}

}

// Присваивает класс и связанное с ним изображение на основе данных массива

function renderCard(pile, num) {

switch(num) {

case 0: $(pile).addClass('hA');

break;

case 1: $(pile).addClass('h02');

break;

case 2: $(pile).addClass('h03');

break;

case 3: $(pile).addClass('h04');

break;

case 4: $(pile).addClass('h05');

break;

case 5: $(pile).addClass('h06');

break;

case 6: $(pile).addClass('h07');

break;

case 7: $(pile).addClass('h08');

break;

case 8: $(pile).addClass('h09');

break;

case 9: $(pile).addClass('h10');

break;

case 10: $(pile).addClass('hJ');

break;

case 11: $(pile).addClass('hQ');

break;

case 12: $(pile).addClass('hK');

break;

case 13: $(pile).addClass('dA');

break;

case 14: $(pile).addClass('d02');

break;

case 15: $(pile).addClass('d03');

break;

case 16: $(pile).addClass('d04');

break;

case 17: $(pile).addClass('d05');

break;

case 18: $(pile).addClass('d06');

break;

case 19: $(pile).addClass('d07');

break;

case 20: $(pile).addClass('d08');

break;

case 21: $(pile).addClass('d09');

break;

case 22: $(pile).addClass('d10');

break;

case 23: $(pile).addClass('dJ');

break;

case 24: $(pile).addClass('dQ');

break;

case 25: $(pile).addClass('dK');

break;

case 26: $(pile).addClass('sA');

break;

case 27: $(pile).addClass('s02');

break;

case 28: $(pile).addClass('s03');

break;

case 29: $(pile).addClass('s04');

break;

case 30: $(pile).addClass('s05');

break;

case 31: $(pile).addClass('s06');

break;

case 32: $(pile).addClass('s07');

break;

case 33: $(pile).addClass('s08');

break;

case 34: $(pile).addClass('s09');

break;

case 35: $(pile).addClass('s10');

break;

case 36: $(pile).addClass('sJ');

break;

case 37: $(pile).addClass('sQ');

break;

case 38: $(pile).addClass('sK');

break;

case 39: $(pile).addClass('cA');

break;

case 40: $(pile).addClass('c02');

break;

case 41: $(pile).addClass('c03');

break;

case 42: $(pile).addClass('c04');

break;

case 43: $(pile).addClass('c05');

break;

case 44: $(pile).addClass('c06');

break;

case 45: $(pile).addClass('c07');

break;

case 46: $(pile).addClass('c08');

break;

case 47: $(pile).addClass('c09');

break;

case 48: $(pile).addClass('c10');

break;

case 49: $(pile).addClass('cJ');

break;

case 50: $(pile).addClass('cQ');

break;

case 51: $(pile).addClass('cK');

break;

}

if (num === undefined) {

$(pile).removeClass('hA h01 h02 h03 h04 h05 h06 h07 h08 h09 h10 hJ hQ hK dA d01 d02 d03 d04 d05 d06 d07 d08 d09 d10 dJ dQ dK sA s01 s02 s03 s04 s05 s06 s07 s08 s09 s10 sJ sQ sK cA c01 c02 c03 c04 c05 c06 c07 c08 c09 c10 cJ cQ cK faceUp').addClass('empty');

}

else {

$(pile).removeClass('empty')

}

}

//сбрасывает переменную firstPile

function resetFirstPile() {

firstPile.length = 0;

render();

}

//обновляет счетчик ходов

function updateMoves() {

$('.moveCounter').text('Ходов: ' + moves)

}

//нормализация непрозрачности

function makeFaceUpSolid() {

$('.faceUp').not('.empty').css('opacity', '1');

$('.empty').css('opacity', '0');

$('.deckPile').css('opacity', '1');

$('.back').css('opacity', '1');

}

function render() {

checkWin();

renderPiles();

renderFaceDowns();

flipFirstCard();

flipAllOtherCards();

renderTop();

addFaceUpClick();

addEmptyClick();

updateMoves();

makeFaceUpSolid();

}

function addFaceUpClick() {

$('.faceUp').off();

var $faceUps = $('td');

for (var i = 7; i < $faceUps.length; i++) {

if (i % 7 === 0) {

selectBoardPile($faceUps, i, boardPile1, boardPile1.length - (i/7));

}

else if (i % 7 === 1 ) {

selectBoardPile($faceUps, i, boardPile2, boardPile2.length - ((i-1)/7));

}

else if (i % 7 === 2 ) {

selectBoardPile($faceUps, i, boardPile3, boardPile3.length - ((i-2)/7));

}

else if (i % 7 === 3 ) {

selectBoardPile($faceUps, i, boardPile4, boardPile4.length - ((i-3)/7));

}

else if (i % 7 === 4 ) {

selectBoardPile($faceUps, i, boardPile5, boardPile5.length - ((i-4)/7));

}

else if (i % 7 === 5 ) {

selectBoardPile($faceUps, i, boardPile6, boardPile6.length - ((i-5)/7));

}

else if (i % 7 === 6 ) {

selectBoardPile($faceUps, i, boardPile7, boardPile7.length - ((i-6)/7));

}

}

};

function selectBoardPile($faceUps, i, pile, num) {

$($faceUps[i]).on('click', function(evt) {

if (firstPile.length === 0) {

if ($(this).hasClass('empty')) {

addEmptyToFirstPile(pile[0])

}

else {

addToFirstPile(pile, num);

$(this).css('opacity', '.8')

}

}

else {

checkBoardMove(firstPile, pile)

}

})

}

function addToFirstPile(pile, num) {

for (var i = num; i >= 0; i--) {

firstPile.unshift(pile[i])

}

}

function addEmptyToFirstPile(card) {

firstPile.unshift(card);

}

init();