

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«ИиВТ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование факультета)

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«ПОВТиАС»

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | А.Н.Карапетянц\_ |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к производственной практике на тему:

«Браузерная игра «Snake»

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.С. Сиденко\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись И.О.Ф.

Обозначение проекта (работы) \_\_09.03.04.420000.000 ПЗ Группа \_ВПР32\_

Направление \_09.03.04\_ \_\_\_\_\_\_\_\_«Программная инженерия»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(код) (наименование направления)

\_Программное обеспечение вычислительной техники и автоматизированных систем

(наименование профиля)

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_ А.М. Атаян\_\_\_\_\_\_\_\_\_

подпись (должность, И.О.Ф.)

Проект (работа) защищен (а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка подпись

Ростов-на-Дону

2018



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«ИиВТ»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование факультета)

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_«ПОВТиАС»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | А.Н.Карапетянц\_\_ |
| (подпись) | | И.О.Ф. |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2018г. | |

**ЗАДАНИЕ**

к производственной практике

Студент\_\_Сиденко Никита Сергеевич\_\_\_\_\_\_ Группа ВПР32

Обозначение проекта (работы) \_\_09.03.04.420000.000 ПЗ

Тема \_Браузерная игра «Snake»

Срок представления проекта (работы) к защите «\_\_» \_\_\_\_\_\_\_\_\_ 2018 г.

Исходные данные для производственной практики

https://habr.com/company/microsoftlumia/blog/136629

https://ru.wikipedia.org/wiki/Snake\_(%D0%B8%D0%B3%D1%80%D0%B0)

http://ru.wikipedia.nom.si/wiki/%D0%90%D1%80%D0%BA%D0%B0%D0%B4%D0%B0\_(%D0%B8%D0%B3%D1%80%D0%B0)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Содержание пояснительной записки | | | | |
| ВВЕДЕНИЕ: | | | | |
| Обзор браузерной игры «Snake» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |
| Наименования и содержание разделов: | | | | |
| \_1. Аналитический обзор. Рассматривается основные понятия, аналоги программного\_\_\_\_\_\_ средства, способы реализации и требования к программе. | | | | |
| \_2. Алгоритмическое конструирование. Рассматриваются основные и вспомогательные алгоритмы программного средства. s | | | | |
| \_3. Программное конструирование. Приводится обоснование выбора языка s программирования и описываются разработанные функции и методы.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  4. Контрольный пример. Приводится главный интерфейс программы и процесс её работы. | | | | |
| ЗАКЛЮЧЕНИЕ: | | | | |
| При выполнении данной работы, был подробно изучен принцип работы с массивами JavaScript и DOM в языке гипертекстовой разметки HTML. Был понят алгоритм многих аркадных игр. Также был изучен способ ускорения работы программы путем делегирования задач разным функциям.  В процессе написания программы не возникло особых трудностей, все функции были разделены на модули, для более понятной структуры кода. Изучены статьи о происхождении игры «Snake», особенностях этой игры, тонкостях алгоритмов,а также о ее разработчиках. | | |
|  | | | | |
|  | | | | |
| Руководитель проекта (работы) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | \_\_А.М. Атаян\_\_\_\_\_  И.О.Ф. | |
|  |  |  | |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | \_\_Н.C. Сиденко\_\_\_\_  И.О.Ф. | |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc522879558)

[1. Аналитический обзор 6](#_Toc522879559)

[1.1 Обзор игры «Snake» 7](#_Toc522879560)

[1.2 Постановка задачи 7](#_Toc522879561)

[1.3 Выводы по главе 7](#_Toc522879562)

[2. Алгоритмическое конструирование 9](#_Toc522879563)

[2.1 Алгоритм паузы и окончания игры 11](#_Toc522879564)

[2.2 Выводы по главе 12](#_Toc522879565)

[3. Программное конструирование 13](#_Toc522879566)

[3.1 Выбор среды для разработки 13](#_Toc522879567)

[3.2 Описание функций и модулей 13](#_Toc522879568)

[3.2.1 Модуль functions.js 13](#_Toc522879569)

[3.2.2 Модуль userSettings.js 15](#_Toc522879570)

[3.4 Выводы по главе 16](#_Toc522879571)

[4. Тестовый пример 17](#_Toc522879572)

[ЗАKЛЮЧЕНИЕ 20](#_Toc522879573)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 21](#_Toc522879574)

[ПРИЛОЖЕНИЕ А Исходный код 22](#_Toc522879575)

# ВВЕДЕНИЕ

Одной из первых аркадных игр является [Computer Space](http://en.wikipedia.nom.si/wiki/Computer_Space) (англ.)[русск.](http://ru.wikipedia.nom.si/w/index.php?title=Computer_Space&action=edit&redlink=1), которая была создана в [1971](http://ru.wikipedia.nom.si/wiki/1971_%D0%B3%D0%BE%D0%B4_%D0%B2_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D1%85_%D0%B8%D0%B3%D1%80%D0%B0%D1%85) году. Она является первой промышленной [электронной игрой](http://en.wikipedia.nom.si/wiki/Electronic_game) (англ.)[русск.](http://ru.wikipedia.nom.si/w/index.php?title=%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0&action=edit&redlink=1)данного жанра. Несмотря на своё название, которое начало ассоциироваться с играми даного типа только в конце 1970-х.

Первые аркадные игры, такие как [Asteroids](http://ru.wikipedia.nom.si/wiki/Asteroids), [Battlezone](http://ru.wikipedia.nom.si/wiki/Battlezone_(%D0%B8%D0%B3%D1%80%D0%B0,_1980)), Star Castle использовали [векторную графику](http://ru.wikipedia.nom.si/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) для [рендеринга](http://ru.wikipedia.nom.si/wiki/%D0%A0%D0%B5%D0%BD%D0%B4%D0%B5%D1%80%D0%B8%D0%BD%D0%B3), когда изображение строилось из линий. После появления цветной [растровой графики](http://ru.wikipedia.nom.si/wiki/%D0%A0%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%BA%D0%B0) появились мультяшные видеоигры и такие персонажи, как [Pac-Man](http://ru.wikipedia.nom.si/wiki/Pac-Man) и [Donkey Kong](http://ru.wikipedia.nom.si/wiki/Donkey_Kong), которые стали культовыми и проявились как явление культуры.

В середине 1980-х аркадные игры начали своё широкое распространение, становясь более разнообразными, как относительно интерфейса управления (аркадного автомата), так и в плане тематики, графики и жанров. Использовались более реалистичные контроллеры, такие как двуместные космические корабли [Tail Gunner](http://en.wikipedia.nom.si/wiki/Tail_Gunner) (англ.)[русск.](http://ru.wikipedia.nom.si/w/index.php?title=Tail_Gunner&action=edit&redlink=1)или управление [Феррари](http://ru.wikipedia.nom.si/wiki/Ferrari) в [Out Run](http://ru.wikipedia.nom.si/wiki/Out_Run). С течением времени аркадные автоматы вытеснялись домашними системами, так как их графика начала обходить большинство аркадных игр, и таким образом к концу 1990-х аркадные автоматы покинули индустрию.

1. **Аналитический обзор**

История игры «Snake» началась за несколько лет до появления первых мобильных телефонов. В 1977 году компания Gremlin Industries выпустила игровой автомат Hustle, рассчитанный на одного или двух игроков, в которой нужно было управлять «змейками», направляя их на бессистемно появляющиеся цели. Для победы нужно было заполучить больше очков, чем у оппонента, преграждая по ходу игры ему путь к новым целям (в случае многопользовательской игры), или просто побить установленный на игровом автомате рекорд. В 1984 году Gremlin Industries была вынуждена закрыться, но игра Hustle начала набирать обороты: сначала появился порт для компьютеров TRS-80, затем для Commodore PET и Apple II.

Оригинальная «Змейка» (Snake) от Nokia появилась в 1997 году благодаря стараниями разработчика Танели Орманто. В том же году компания выпустила первый телефон с этой игрой — Nokia 6110. Уже тогда игра была многопользовательской: телефоны общались через ИК-порты, ведь ни Bluetooth, ни тем более Wi-Fi в телефонах в то время не было. Сама змейка состояла из чёрных квадратов и могла двигаться в четырёх направлениях. Игровая зона, по которой передвигалось пресмыкающееся, была ограничена размерами экрана телефона: при ударе головы змейки о край телефона игра завершалась. «Змейка» приобрела невероятную популярность, сравнимую разве что с популярностью современных хитов «Angry Birds» и «Cut the Rope» [1].

**1.1 Обзор игры «Snake»**

Игрок управляет длинным, тонким существом, напоминающим змею, которое ползает по плоскости (как правило, ограниченной стенками), собирая еду (или другие предметы), избегая столкновения с собственным хвостом и краями игрового поля. В некоторых вариантах на поле присутствуют дополнительные препятствия. Каждый раз, когда змея съедает кусок пищи, она становится длиннее, что постепенно усложняет игру. Игрок управляет направлением движения головы змеи (обычно 4 направления: вверх, вниз, влево, вправо), а хвост змеи движется следом. Игрок не может остановить движение змеи [2].

**1.2 Постановка задачи**

Цель данной работы заключается в том, чтобы разработать игровое поле, включающее в себя змею и игровые очки(бонусы) [2].

Чтобы игра имела смысл должно быть создано условие окончания игры, игра заканчивается, если голова змеи встретится с хвостом. При завершении игры будет выводится результат игрока, включающий в себя время и счет.

Что касается алгоритма передвижения змеи, здесь нет каких-либо особенностей, самое главное, чтобы перемещение хвоста работало также корректно, как и перемещение головы, при этом должна быть реализована возможность прохода через границы игрового поля [2].

**1.3 Выводы по главе**

В данной главе был проведен аналитический обзор игры «Snake», поставлена цель текущей практической работы, а так же были установлены необходимые ключевые функции.

1. **Алгоритмическое конструирование**

Основной алгоритм данной игры заключается в том, чтобы обрабатывать нажатие клавиш пользователем, тем самым изменяя направление движения змеи. Для реализации одной из самых главных функций (движение змеи) игры необохдимо:

* иметь данные о текущем положении головы змеи и положении хвоста;
* на каждой итерации проверять следующую ячейку массива(игрового поля) на наличие игрового бонуса;
* в зависимости от нажатой клавиши менять направление движения змеи;
* прекращать движение змеи, если голова встретилась с хвостом;
* на каждом шаге проверять условия конца игры(встреча головы змеи с хвостом).

Алгоритмы движения змеи и генерации бонусов реализованы следующим образом.

На старте игры запускается таймер, по мере увеличения значения которого срабатывает функция, которая в свою очередь в зависимости от нажатой игроком клавиши двигает голову змеи. Все координаты змеи, в том числе и голова хранятся в массиве, при изменении положения головы, вслед за ним двигается хвост: каждый элемент массива, получает координаты следующего. Так же функция включает в себя проверку на встречу головы с хвостом, в случае встречи игра заканчивается, выводится результат. На каждой итерации обрабатывается случай подбора монетки, если монетка подобрана, она исчезает, появляется новая и змея становится длиннее.

Ниже на рисунке 1 изображена блоксхема функции передвижения змеи

Если задано какое-либо направление

Да

Да

Нет

Если на слежующем шаге лежит монетка

Проверяем ячейку на наличие хвоста, перемещаем змею вслед за головой

Убираем монетку, добавляем длину змее

Рисунок 1 – Блоксхема функции передвижения змеи

Конец

Начало

Нет

## 2.1 Алгоритм паузы и окончания игры

В игре реализована возможность временной остановки игры(паузы), а так же предусмотрен случай полной остановки игры (проигрыша).В случае нажатии определенной клавиши (Esc), игра ставится на паузу, все таймеры останавливаются, змейка не двигается, при повторном нажатии, действие игры возобнавляется, при этом во время паузы выводится прозрачное модальное окно с информацией об остановке игры, то же самое происходит и при проигрыше, только при этом выводится информация и о результате игрока.Ниже, на рисунке 2 приведена блоксхема функции остановки игры

Рисунок 2 – Блоксхема функции остановки игры

Начало

Конец

Если зажата клавиша Escape

Да

Нет

Остановить игру

## 2.2 Выводы по главе

В главе были выделены основные функции, необходимые для реализации поставленной задачи, а именно функция паузы и окончания игры и функция передвижения змеи, а так же разобрана их логика. Были приведены блоксхемы функций с подробным описанием их алгоритмов

1. **Программное конструирование**

В данной главе обосновывается выбор языка программирования и среды разработки, описывается работа функций и модулей программы.

## 3.1 Выбор среды для разработки

Так как был выбран браузерный способ реализации игры, выбор языка упал на JavaScript. Весь код был написан в функциональном стиле, функции были разделены на модули, в зависимости от их предназначения. Всю визуальную составляющую проекта взял на себя язык гипертекстовой разметки HTML и каскадные таблицы стилей CSS. Расматривался вопрос кросс-браузерности, но было принято решение разрабатывать игру под браузер Chrome, в других браузерах игра будет работать некорректно, либо не будет работать вовсе.

## 3.2 Описание функций и модулей

Ниже можно увидеть описание всех переменных, функций и модулей программы см.таблицы 1-4.

### 3.2.1 Модуль functions.js

Модуль functions.js отвечает за заполнение игрового поля, движение змеи, генерацию монет, хранит в себе данные о положении змеи и монетах, отвечает за проверку условия окончания игры и тд. Ниже представлены главные переменные и функции модуля functions.js см. таблицы 1-2.

Таблица 1. Переменные модуля functions.js

|  |  |
| --- | --- |
| Переменная | Описание |
| var head[] | Массив,отвечающий за положение головы змеи |
| var x, var y | Координаты головы змеи |
| var snake[] | Массив, хранящий в себе координаты головы и хвоста змеи |
| var timer | Переменная, инициализирующая таймер |
| var direction | Текущие направление головы змеи |
| var coinsCount | Переменная, хранящая в себе кол-во собранных бонусов |
| var looseGame | Переменная, хранящая в себе состояние конца игры |

Таблица 2. Функции модуля functions.js

|  |  |
| --- | --- |
| Функция | Описание |
| function generateField () | Функция, генерирующая игровое поле |
| function drawSnake () | Функция, перерисовывающая положение змеи |
| function move () | Функция, которая отвечает за движение змеи |
| function pressButton () | Функция, отслеживающая нажатие клавиш пользователем |
| function startGame () | Функция, которая непосредственно запускает игру |

### 3.2.2 Модуль userSettings.js

Модуль userSettings.js отвечает за старт таймера игры, подсчет кол-ва собранных бонусов, хранит в себе функции паузы и окончания игры,а так же функцию генерации монеток, функцию проверки на проигрыш, функцию рестарта игры см.Таблицы 3-4.

Таблица 3. Переменные модуля userSettings.js

|  |  |
| --- | --- |
| Переменная описание | Описание |
| var score | Кол-во собранных бонусов |
| var time | Кол-во секунд,прошедших со старта игры |
| var scoreField | Переменная, хранящая в себе ссылку на DOM элемент,в который будет выводится кол-во собранных бонусов |
| var timeField | Переменная, хранящая в себе ссылку на DOM элемент,в который будет выводится кол-во секунд, прошедших со старта игры |
| var secondsTimer | Переменная, инициализирующая таймер времени игры |
| var pausedGame | Переменная, хранящая в себе состояние паузы игры |

Таблица 4. Функции модуля userSettings.js

|  |  |
| --- | --- |
| Функция | Описание |
| function startTime () | Функция, запускающая таймер времени игры |
| function pauseGame () | Функция, которая отвечает за остановку игры |
| function generateCoin () | Функция, генерирующая бонус в случайном месте |
| function checkLoose () | Функция, проверяющая состояние проигрыша |
| function replayGame () | Функция, которая отвечает за перезапуск игры |

## 3.4 Выводы по главе

В этой главе были описаны все модули, функции и переменные данной программы, структура и их предназначение.Так же был разобран принцип их работы, отдельно описаны алгоритмы всех подпрограмм. Были приведены таблицы для структурированного отображения переменных и функций программы.

# Тестовый пример

При включении перед нами открывается интерфейс игры, таймер, счет игрока и текущий рекорд. Так же есть возможность посмотреть горячие клавиши с помощью кнопки «Tutorial». Интерфейс игры представлен на рисунке 3.

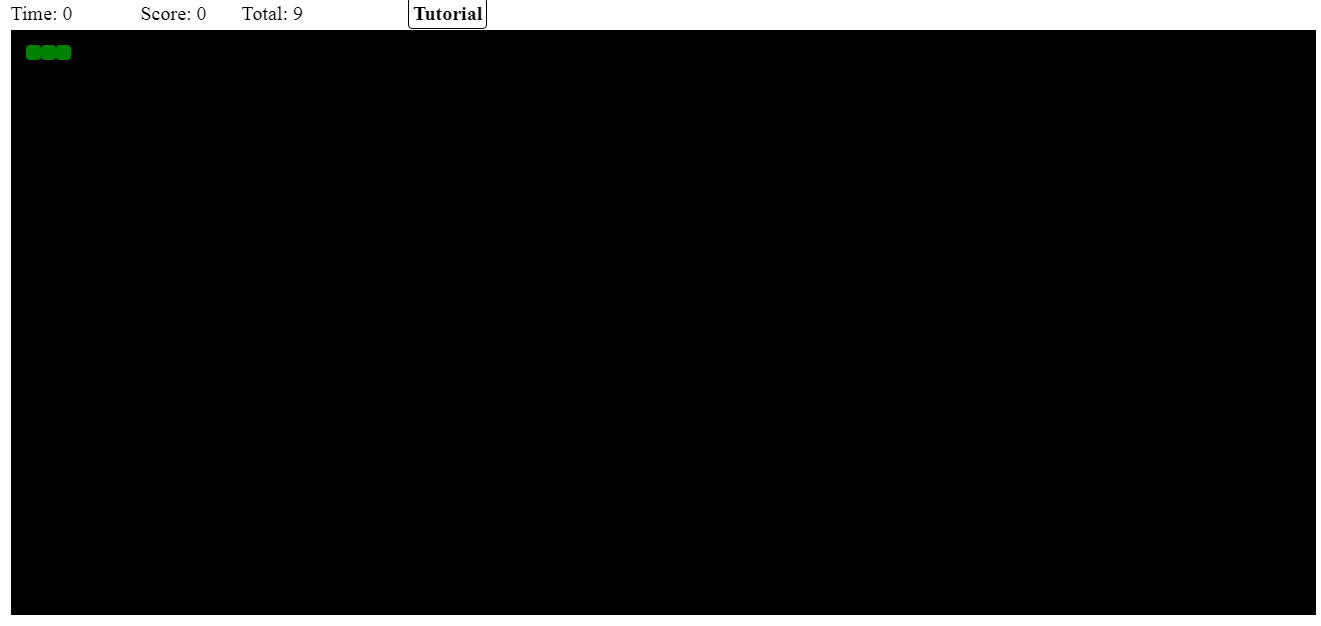


Рисунок 3 – Интерфейс игры

Начало игры происходит в случае нажатия на одну из четырех клавиш передвижения, змейка начинает движение, игрок получает управление. На рисунке 4 изображен процесс сбора игровых бонусов.

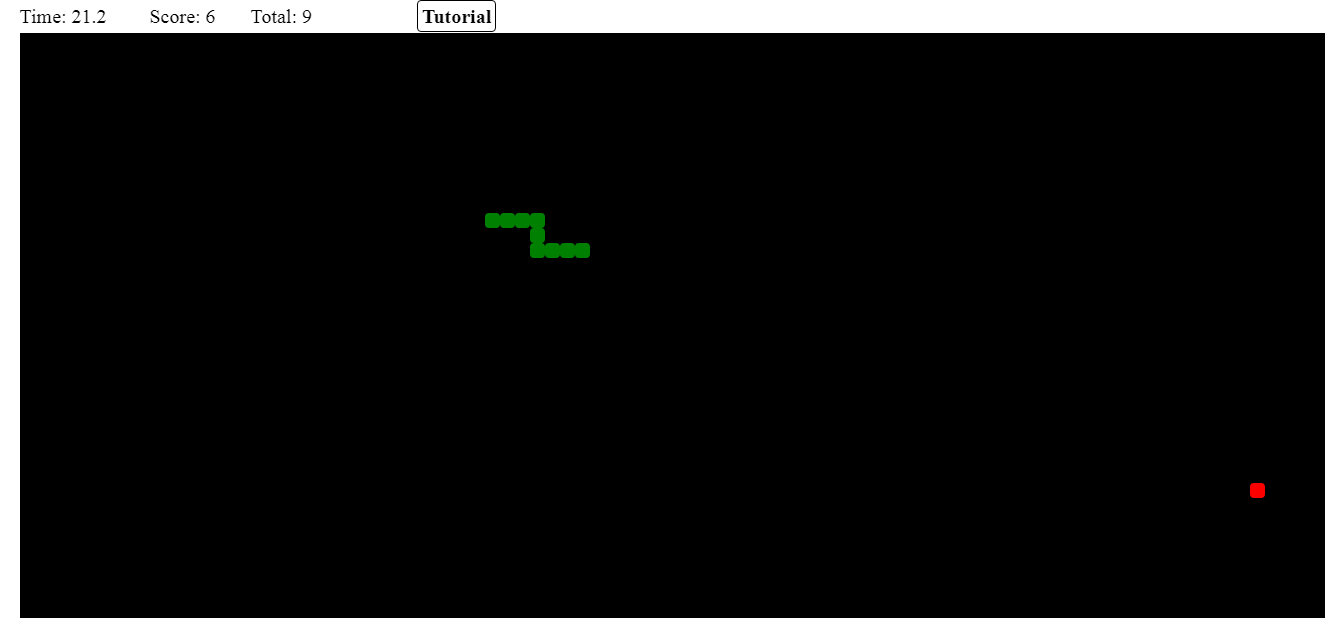


Рисунок 4 – Сбор бонусов и передвижение змеи

В случае нажатия на горучую клавишу Esc, игра ставится на паузу, время останавливается, на рисунке 5 изображен момент остановки игры.

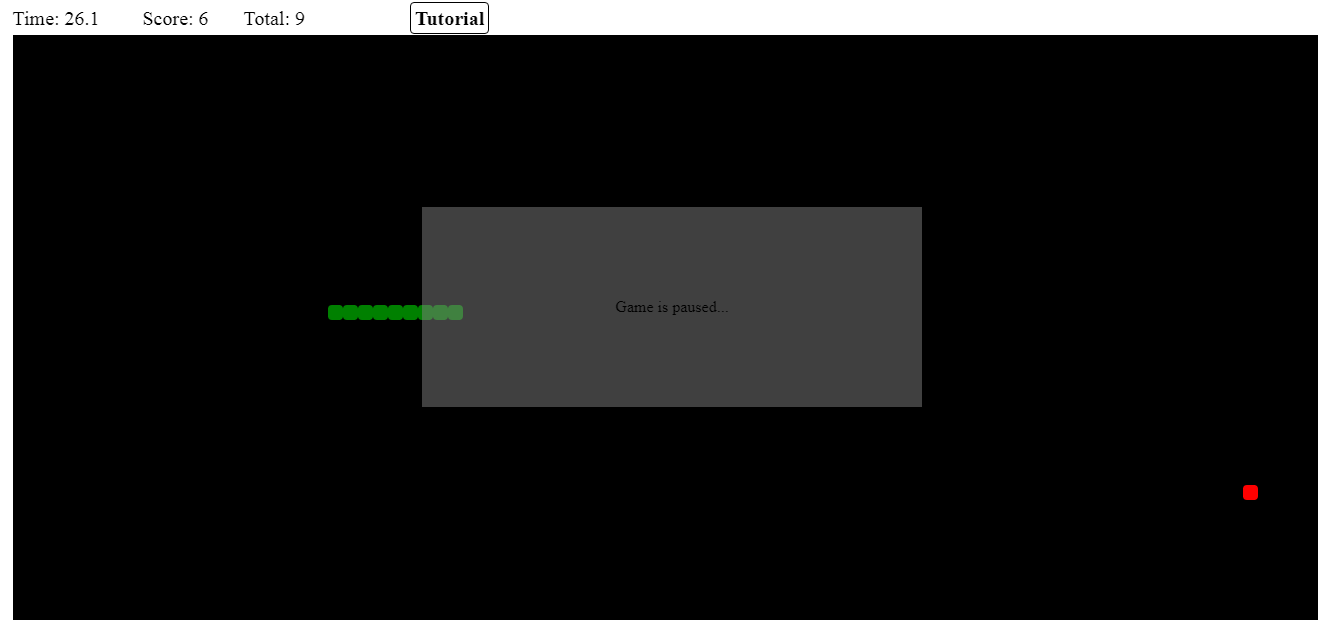


Рисунок 5 – Пауза

В случае проигрыша игра останавливается, выводится модальное окно с информацией о текущем счете и времени игрока, на рисунке 6 изображен момент проигрыша игрока.



Рисунок 6 – Конец игры(проигрыш)

# ЗАKЛЮЧЕНИЕ

При выполнении данной работы, был подробно изучен принцип работы с массивами JavaScript и DOM в языке гипертекстовой разметки HTML. Был понят алгоритм многих аркадных игр. Также был изучен способ ускорения работы программы путем делегирования задач разным функциям.

В процессе написания программы не возникло особых трудностей, все функции были разделены на модули, для более понятной структуры кода. Изучены статьи о происхождении игры «Snake», особенностях этой игры, тонкостях алгоритмов,а также о ее разработчиках.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1.История игры Snake [Электронный ресурс]. - https://habr.com/company/microsoftlumia/blog/136629,дата обращения(12.08.2018);

2.Работа с двумерными массивами в JavaScript [Электронный ресурс]. - http://sozdatisite.ru/js/dvumernye-masivy,дата обращения(12.08.2018) ;

3.Работа с DOM в HTML [Электронный ресурс]. - http://javascript.ru/tutorial/dom,дата обращения(12.08.2018) ;

4.Модули в JavaScript [Электронный ресурс]. - https://learn.javascript.ru/modules,дата обращения(12.08.2018).

# ПРИЛОЖЕНИЕ А Исходный код

var x = 3,

y = 1,

head = [y,x],

snake = [[1,1],[1,2],head],

timer,

direction = [0,0],

coinsCond = false,

coinsCount = 0,

xCoin,

yCoin,

prevHead = [],

looseGame = false

function generateField () {

var field = document.getElementById('main')

field.innerHTML = ''

for (var y = 0; y < 39; y++) {

for (var x = 0; x < 87; x++) {

var div = document.createElement('div')

div.classList.add('cell')

div.id = y + ',' + x

field.appendChild(div)

}

}

}

function drowSnake () {

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

var id = snake[i].join(',')

document.getElementById(id).classList.add('snakeCells')

}

}

function move () {

timer = setInterval(()=>{

if (direction[0] !== 0 || direction[1] !== 0){

if (head[0] + direction[0] === yCoin && head[1] + direction[1] === xCoin) {

document.getElementById(yCoin + ',' + xCoin).classList.remove('cellCoin')

snake.push([yCoin,xCoin])

head = [yCoin,xCoin]

coinsCond = false

coinsCount++

}

checkLoose()

document.getElementById(snake[0].join(',')).classList.remove('snakeCells')

for (var i = 0; i < snake.length - 1; i++) {

snake[i][0] = snake[i + 1][0]

snake[i][1] = snake[i + 1][1]

}

if (head[0] + direction[0] < 0) {

head[0] = 39

}

if (head[0] + direction[0] > 38) {

head[0] = -1

}

if (head[1] + direction[1] > 86) {

head[1] = -1

}

if (head[1] + direction[1] < 0) {

head[1] = 87

}

prevHead[0] = head[0]

prevHead[1] = head[1]

head[0] += direction[0]

head[1] += direction[1]

snake[snake.length - 1] = head

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

document.getElementById(snake[i].join(',')).classList.add('snakeCells')

}

drowSnake()

generateCoin()

scoreField.innerHTML = 'Score: ' + coinsCount

}

},60)

}

function pressButton (event) {

switch (event.keyCode) {

case 1099:

if (direction[0] !== -1 && head[0] + 1 !== prevHead[0] && head[1] !== prevHead[1])

direction = [1,0]

break;

case 1094:

if (direction[0] !== 1 && head[0] - 1 !== prevHead[0] && head[1] !== prevHead[1])

direction = [-1,0]

break;

case 1092:

if (direction[1] !== 1 && head[0] !== prevHead[0] && head[1] - 1 !== prevHead[1])

direction = [0,-1]

break;

case 1074:

if (direction[1] !== -1 && head[0] !== prevHead[0] && head[1] + 1 !== prevHead[1])

direction = [0,1]

break;

}

console.log(event.charCode)

}

function startGame () {

direction[0] = 0

direction[1] = 0

coinsCond = false

time = 0

coinsCount = 0

generateField()

head = [y,x]

snake = [[1,1],[1,2],head]

drowSnake()

window.addEventListener('keypress',pressButton)

move()

startTime()

saveResult()

}

startGame()

var time = 0,

scoreField = document.getElementById('score'),

timeField = document.getElementById('time'),

secondsTimer,

pausedGame = false

function startTime () {

secondsTimer = setInterval(() => {

if (direction[0] !== 0 || direction[1] !== 0){

time += 0.1

timeField.innerHTML = 'Time: ' + time.toFixed(1)

}

},100)

}

function pauseGame (event) {

var pausedModal = document.getElementById('pausedGame')

if (event.keyCode === 27 && pausedGame === false && (direction[0] !== 0 || direction[1] !== 0)) {

pausedGame = true

window.removeEventListener('keypress',pressButton)

clearInterval(timer)

clearInterval(secondsTimer)

pausedModal.style.display = 'block'

} else if (event.keyCode === 27 && pausedGame === true){

pausedGame = false

window.addEventListener('keypress',pressButton)

move()

startTime()

pausedModal.style.display = 'none'

}

}

function generateCoin () {

if (!coinsCond) {

yCoin = Math.floor(Math.random() \* (38 - 0)) + 0;

xCoin = Math.floor(Math.random() \* (86 - 0)) + 0;

document.getElementById(yCoin + ',' + xCoin).classList.add('cellCoin')

coinsCond = true

}

}

function checkLoose () {

if (document.getElementById(Number(head[0] + direction[0]) + ',' + Number(head[1] + direction[1])) !== null)

if (document.getElementById(Number(head[0] + direction[0]) + ',' + Number(head[1] + direction[1])).classList.contains('snakeCells')){

document.getElementById('looseGame').style.display = 'block'

document.getElementById('looseGame').innerHTML = 'You loose with Score:<b>' + coinsCount + '</b> and time:<b>' + time.toFixed(1) + 's</b>. ' + 'Press ENTER to replay.'

window.removeEventListener('keypress',pressButton)

window.removeEventListener('keydown',pauseGame)

looseGame = true

clearInterval(secondsTimer)

clearInterval(timer)

saveResult()

}

}

function saveResult () {

if (localStorage.getItem('result') === null) {

localStorage.setItem('result', 0)

}

var result = localStorage.getItem('result')

document.getElementById('total').innerHTML = 'Total: ' + result

if (coinsCount > parseInt(result)) {

localStorage.removeItem('result')

localStorage.setItem('result', coinsCount)

document.getElementById('total').innerHTML = 'Total: ' + result

}

}

function replayGame (event) {

if (looseGame && event.keyCode === 13) {

document.getElementById('looseGame').style.display = 'none'

window.addEventListener('keypress',pressButton)

window.addEventListener('keydown',pauseGame)

looseGame = false

timeField.innerHTML = 'Time: 0'

scoreField.innerHTML = 'Score: 0'

startGame()

}

}

window.addEventListener('load',startTime)

window.addEventListener('keydown',pauseGame)

window.addEventListener('keypress',replayGame)

document.getElementById('Tutorial').addEventListener('click',function () {

document.getElementById('modalTutorial').classList.toggle('open')

})

window.addEventListener('load', saveResult)