МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ(№ 43)

КУРСОВАЯ РАБОТА   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е.О. Шумова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
|  |
| ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ Курсовой проект |
| Разработка оконного приложения для организации взаимодействия объектов при заданных критериях |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4631 |  |  |  | С.А. Гришин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Содержание**

|  |  |
| --- | --- |
| 1. Задание…………………………………………….................................... | 3 |
| 2. Анализ предметной области………………………………….…………. | 3 |
| 3. Разработка программного обеспечения……………………………....... | 3 |
| 3.1. Разработка структуры иерархии классов…………….…………..… | 3 |
| 3.2. Проектирование интерфейса программы.……………………..….... | 4 |
| 3.3. Разработка модулей программного продукта ………..……………. | 5 |
| 4. Тестирование.…………...……………………………………..…...…….. | 6 |
| 5.Руководство пользователя………………………….…………………… | 10 |
| Заключение ………………………………………………………….……... | 10 |
|  |  |
| Приложение……………………………………………………………..….. | 11 |
|  |  |
|  |  |

1. **Задание**

Разработать программный продукт представляющий собой игру со следующим содержанием.

Суть игры заключается в постройке трубопровода от одного края игрового поля на время. Играет один игрок. Компьютер предлагает очередной узел трубопровода, игрок может присоединить его к концу трубопровода. В определенный момент времени с начала игры, в трубопровод начинает подаваться вода. Если трубопровод будет заполнен до того, как игрок закончит строительство — игрок проиграл.

1. **Анализ предметной области**

За общую логику в игре будет отвечать класс “Game”. Он отвечает за инициализацию ресурсов игры, таких как файлы с изображениями, текст и стили к игре. Все действия игры происходящие с трубопроводом будут отображаться на экране пользователя.

Класс “Pipeline”, который будет отвечать за игровое поле, которое будет состоять из элементов класса “Pipe”, реализующие один элемент трубы. Отдельный класс будет отвечать за игровую логику “GameLogic”: таймер, пуск воды и подсчет кол-ва очков.

Проверка правильности водопровода будет осуществляться с помощью рекурсивной проверки от начал водлпровода и идти по водопроводу. Если в конечном итоге мы не пришли к выходу из водопровода, значит водопровод не закончен.

1. **Разработка программного обеспечения**
   1. **Разработка структуры иерархии классов**

Главным классом данной игры будет класс “Game”. Он будет инициализировать другие объекты, занимать первоначальной загрузкой. Этот класс создаст объект класса “Pipeline”.

Все игровые объекты, которые создаются из классов, будут созданы с помощью паттерна “Прототип”. Данный паттерн был выбран из-за того, что в языке JavaScript он используется для создания объектов из описания класса. Даже класс “Game” был реализован с помощью данного паттерна.

Игровой класс “Game” после загрузки инициализирует объекты на основе классов “Pipeline” и “GameLogic”.

Пример описания класса на языке JavaScript с использованием паттерна Прототип:

function Pipe(img) {

this.img = img;

}

Pipe.prototype.GetInfo = function() {

return this. img.split("/").pop().split(".")[0].split("-")[1].split("\_")

};

Pipe.prototype.Create = function(img) {

return new Pipe(img);

};

Класс “Pipeline” создает элементы трубы, объекты класса “Pipe”, используя паттерн “Фабрика”. “Pipeline” определяет в методе MakePipe какой тип трубы должен получиться и он возвращается как результат работы функции. В конечном итоге объект Pipe создается с помощью паттерна “Прототип” из-за особенностей языка. Так как JavaScript является мультипарадигменным языком. В нем можно писать как в OOP, так и в функциональном стиле.

В классе GameLogic применяется паттерн “Одиночка”. Из-за того, что данный паттерн должен быть представлены только водном экземпляре в течении всей игры.

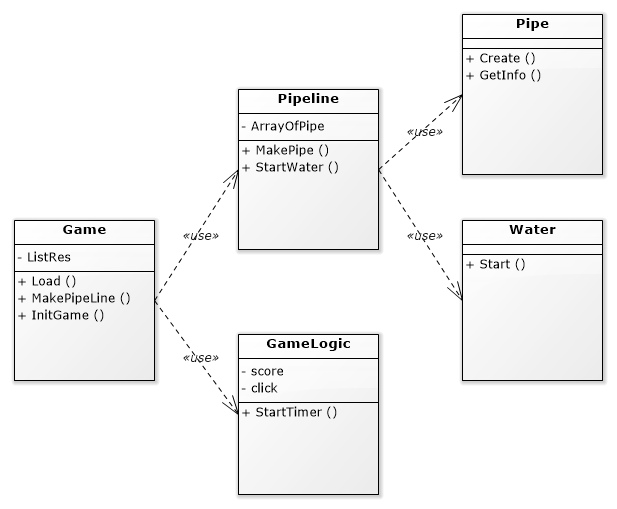


Рис.3.1.1

* 1. **Проектирование интерфейса программы**

Первым пользователю будет представлен экран с возможностью выбора, ознакомления с правилами или началом игры. Откуда у пользователя будет возможность вернуться на стартовый экран и начать игровой процесс, по завершении которого пользователь опять же будет перенаправлен на стартовый экран. Схема представлена на рис.3.2.

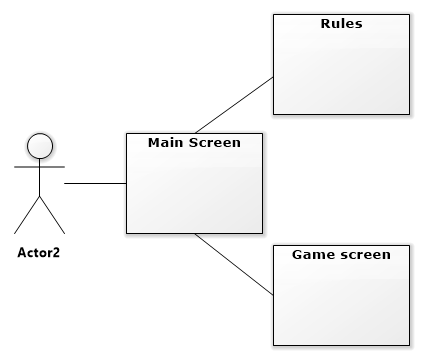


Рис.3.2

* 1. **Разработка модулей программного продукта**

Согласно разбиению на логические модули, экраны для взаимодействия пользователя представляют собой 3 отдельных файла, каждый из которых подключает к себе необходимые для отображения CSS файлы. И необходимые для реализации игровой логики JS файлы. Загружаемые ресурсы, находятся в папке img. Схема на рис.3.3.1.

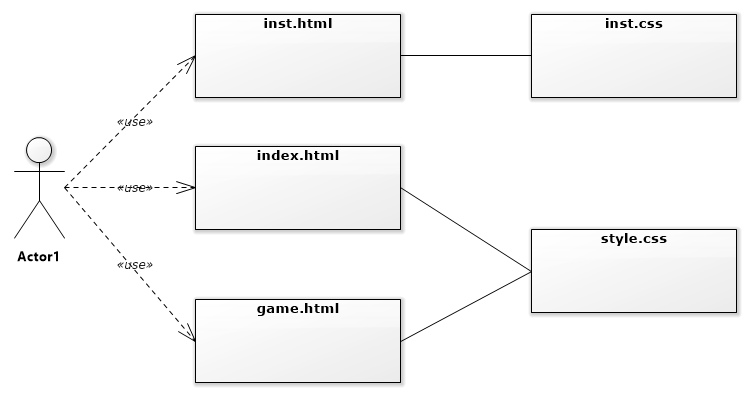


Рис.3.3.1

В начале игры будет заранее размещены элементы начала и конца трубопровода. Они будут не интерактивны и необходимы для реализации алгоритма проверки правильности водопровода. Мы будем рекурсивно начинать проверку правильности водопровода с начала элемента и до конца водопровода. Если последний элемент окажется заранее, статично заданным элементом водопровода, то водопровод является верным. В начале игры все поле заполняется пустыми элементами, при клике на ячейку происходит взятие элемента из верхнего меню. В меню происходит выбор случайного элемента, а предыдущий перемещается на место клика. Процесс изменения ячейки – это изменение “src” HTML элемента <img/>. Примеры атрибута src представлены ниже.

Алгоритм проверки водопровода:

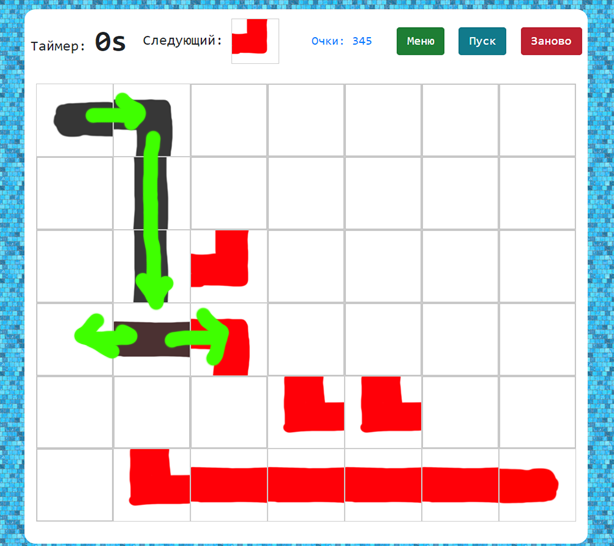


Рис.3.3.2

Зелёными линиями на Рис.3.3.2 показано где шла проверка, куда могла пойти вода. Так как выход предыдущего не совпадал с входом другого, то водопровод является не завершённым.

Направление воды, куда может пойти, записано в имени изображения трубы.

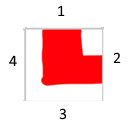


Рис.3.3.3

На Рис.3.3.3 цифрами показаны направления потока воды.

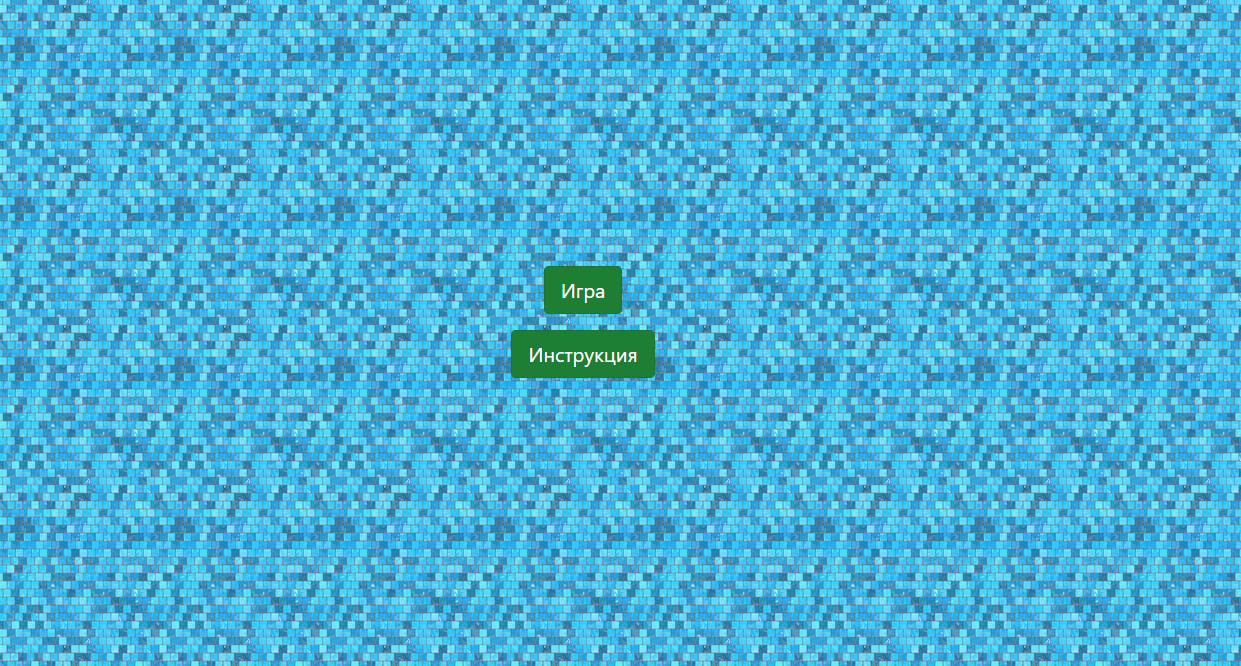
Таким образом для данного элемента трубы будет задано значение : **p-1\_1\_0\_0.png**

Далее это имя будет преобразовано в массив [1, 1, 0, 0], которое будет использовать при проверке, совпадают ли трубы.

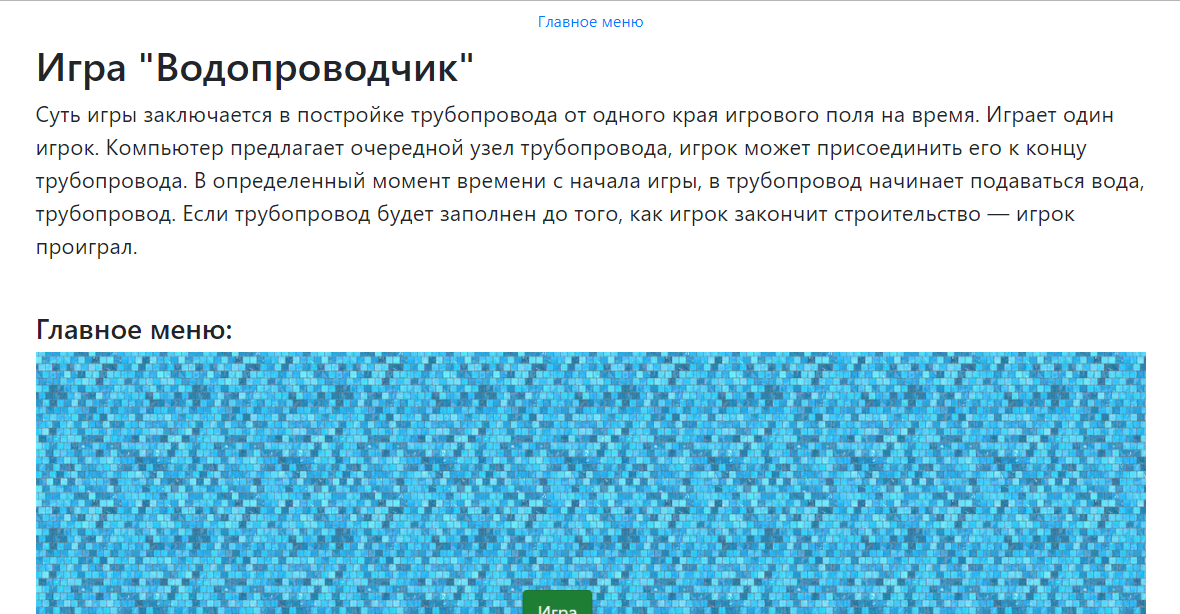
Все трубы, по которым начинает течь вода, начинают изменять цвет. Это реализовано с помощью изменения CSS свойств элемента трубы. Из-за того, что возможности CSS ограничены, был выбран вариант перевода в серый цвет с помощью фильтров.

filter: grayscale(100%);

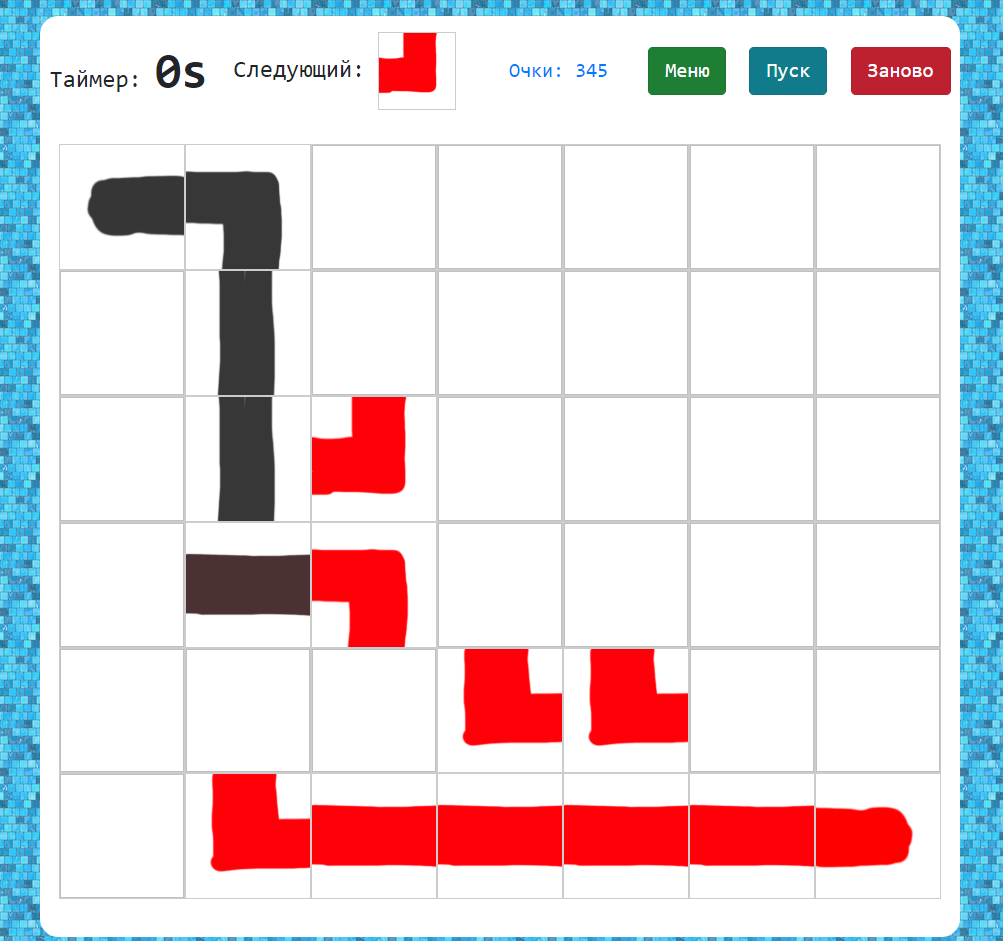
1. **Тестирование**
2. Стартовый экран



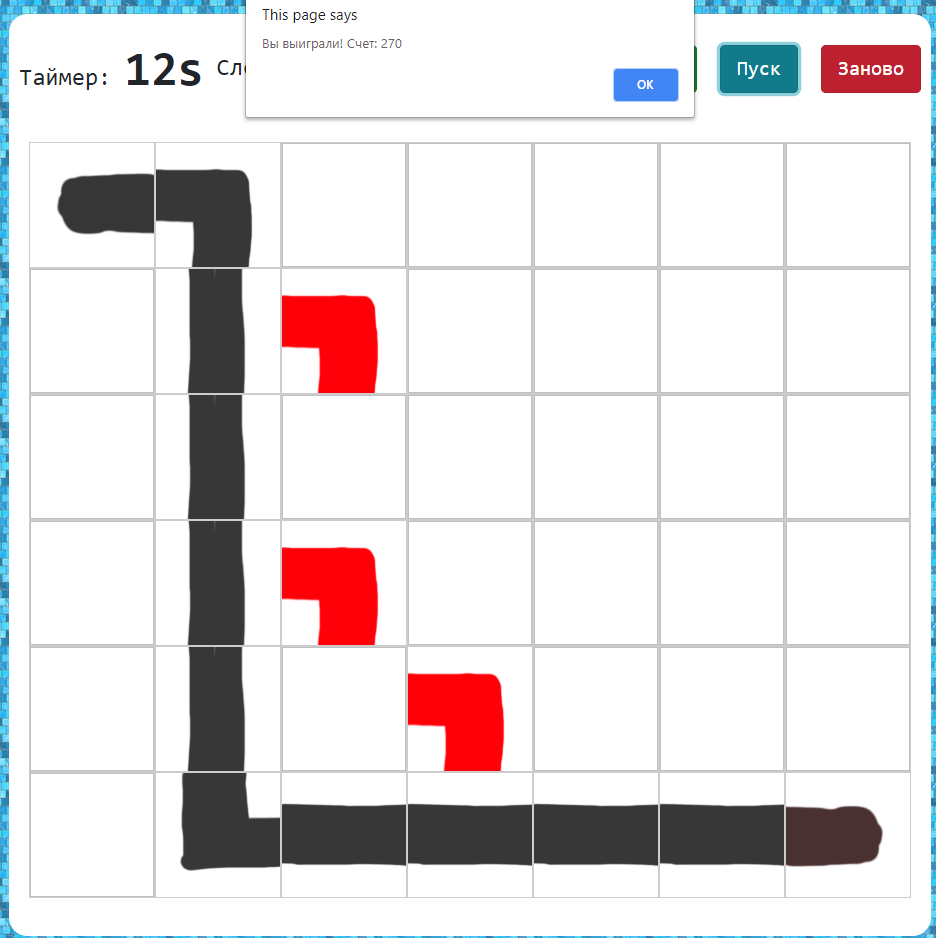
1. Экран с правилами



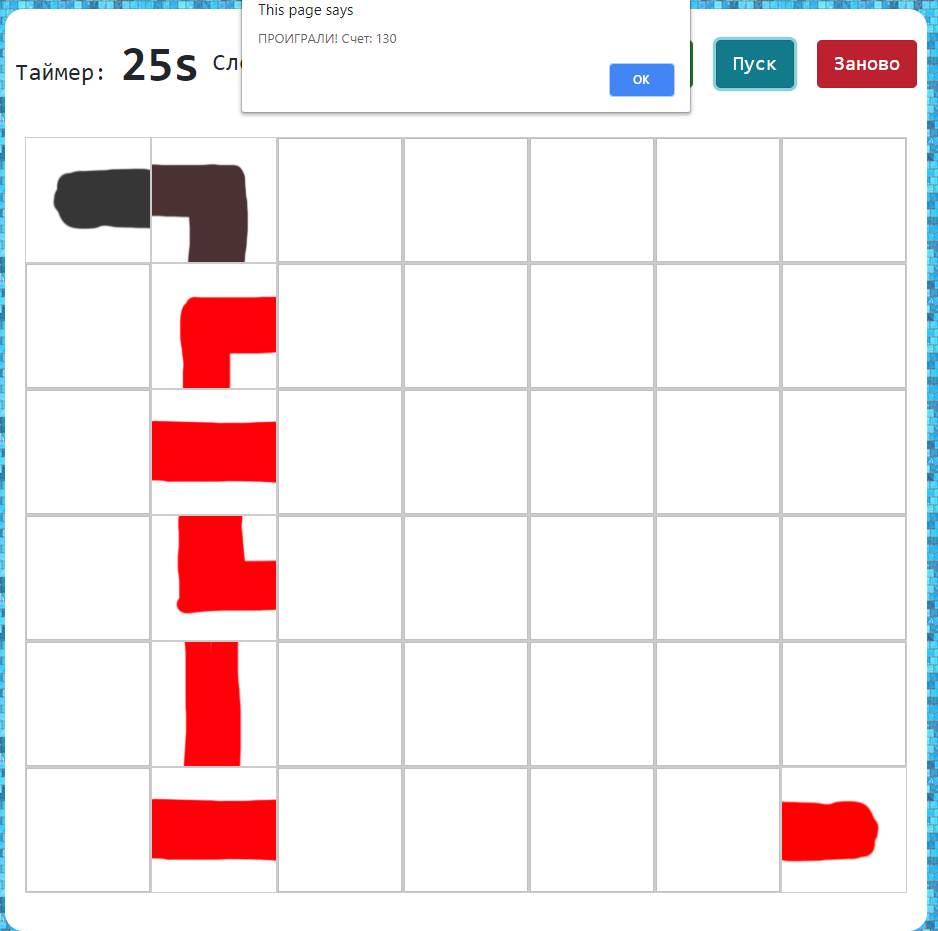
1. Игровой экран



1. Удачно построенный водопровод



1. Игрок не успел построить



1. **Руководство пользователя**

**Программные требования к игре:** Браузер Сhrome или Firefox (версия не младше 50)

Игра распространяется при помощи архива.

Для установки игры следует распаковать содержимое архива и открыть файл «index.html» при помощи браузера.   
 Управление осуществляется при помощи мышки. Дополнительную информацию можно получить в блоке «Инструкция».

**Заключение**

В ходе выполнения курсового проекта было разработано приложение: «Водопроводчик». Позволяющая пользователю посоревноваться в скорости со злобными гибби в процессе сбора зацветших растений.

В качестве среды выполнения используется Web-браузер. Для выполнения работы использовался язык разметки html, каскадные таблицы стилей css, для добавления возможности взаимодействия пользователя с web-страницей используется язык JavaScript.   
 В ходе написания программных модулей использовался Объектно-ориентированный подход, что позволило структурировать код и в значительной мере облегчило процесс разработки и поддержки программного продукта.  
 Так же были использованы такие паттерны, как Прототип, Фабрика и Одиночка. Что позволило сделать код легко читаемым и понимаемым для других программистов, а так-же позволило избежать архитектурных ошибок в ходе разработки программного продукта.

**Библиографический список**

1. Рамбо, Якобсон, Буч: Введение в UML от создателей языка – ДМК-Пресс, 2015. – 496c.
2. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования – Диалектика, 2016. –736c.
3. М. Хавербеке. Выразительный JavaScript 2-е издание, 2015.-306с.
4. Томас А. Пауэл, Фриц Шнайдер. Полный справочник по JavaScript - ДМК-Пресс, 2016. – 700c
5. Пайлон, Питмен: UML 2 для программистов – Питер, 2012 – 240c.

**Приложение**

const imgs = [

"p-0\_0\_1\_1.png",

"p-0\_1\_0\_1.png",

"p-0\_1\_1\_0.png",

"p-1\_0\_0\_1.png",

"p-1\_0\_1\_0.png",

"p-1\_1\_0\_0.png"

];

let click = 0;

let score = 0;

const maxHUE = 100;

function Pipe(img) {

this.img = img;

}

Pipe.prototype.GetInfo = function() {

return this.img

.split("/")

.pop()

.split(".")[0]

.split("-")[1]

.split("\_");

};

Pipe.prototype.Create = function(img) {

return new Pipe(img);

};

function infoByImg(x, y, path) {

if (!path || path.length == 0) {

return [];

}

const direct = path

.split("/")

.pop()

.split(".")[0]

.split("-")[1]

.split("\_");

const where = direct.map((val, inx) => {

if (inx == 0 && val != 0) {

return {

x: x - 1,

y: y

};

}

if (inx == 1 && val != 0) {

return {

x: x,

y: y + 1

};

}

if (inx == 2 && val != 0) {

return {

x: x + 1,

y: y

};

}

if (inx == 3 && val != 0) {

return {

x: x,

y: y - 1

};

}

return false;

});

return where.filter(el => el);

}

Object.defineProperty(Array.prototype, "chunk\_inefficient", {

value: function(chunkSize) {

var array = this;

return [].concat.apply(

[],

array.map(function(elem, i) {

return i % chunkSize ? [] : [array.slice(i, i + chunkSize)];

})

);

}

});

function getMap() {

const cells = document.querySelectorAll(".cell");

const st = document.querySelector(".cell\_start");

const ed = document.querySelector(".cell\_end");

all = [st, ...cells, ed];

return all.chunk\_inefficient(7);

}

function select(event) {

const next = document.querySelector(".next-cell");

const next\_img = next.src.split("/").pop();

next.src = `./img/${imgs[Math.floor(Math.random() \* imgs.length)]}`;

event.toElement.src = `./img/${next\_img}`;

score += 5;

}

window.onload = function name(params) {

getMap();

const cells = document.querySelectorAll(".cell");

for (const el of cells) {

el.addEventListener("click", select);

}

document.querySelector("#start").addEventListener("click", startWater);

startTimer(35);

updateScore();

};

function startTimer(start) {

let t = setInterval(() => {

document.querySelector(".timer").innerHTML = start + "s";

score += 5;

if (start <= 0) {

clearInterval(t);

startWater();

}

start -= 1;

}, 1000);

}

function updateScore() {

setInterval(() => {

const el = document.querySelector("#score");

el.innerHTML = `Очки: ${score}`;

}, 100);

}

function startWater() {

setInterval(() => {

addWater(0, 0);

}, 200);

setTimeout(function() {

const end = getMap()[5][6];

if (addHUE(end, 0) > 0) {

alert(`Вы выиграли! Счет: ${score}`);

} else {

alert(`ПРОИГРАЛИ! Счет: ${score}`);

}

}, 500);

}

function addWater(x, y) {

const el = getMap()[x][y];

if (addHUE(el, 10) >= maxHUE) {

return;

}

let next = infoByImg(x, y, el.src).filter(el => {

const nextEl = getMap()[el.x][el.y];

const isConnect = infoByImg(el.x, el.y, nextEl.src).some(

i => i.x == x && i.y == y

);

return isConnect;

});

for (const iter of next) {

addWater(iter.x, iter.y);

}

}

function addHUE(el, inc) {

const cur = el.style.filter.match(/\d+/g) | ["0"];

const next = cur + inc;

el.style.filter = `grayscale(${next}%)`;

return next;

}

}

function addWater(x, y) {

const el = getMap()[x][y]

if (addHUE(el, 10) >= maxHUE) {

return;

}

let next = infoByImg(x, y, el.src).filter(el => {

const nextEl = getMap()[el.x][el.y]

const isConnect = infoByImg(el.x, el.y, nextEl.src).some(i => i.x == x && i.y == y)

return isConnect

})

for (const iter of next) {

addWater(iter.x, iter.y)

}

}

function addHUE(el, inc) {

const cur = el.style.filter.match(/\d+/g) | ["0"];

const next = cur + inc;

el.style.filter = `grayscale(${next}%)`;

return next;

}