МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

КАЛИНИНГРАДСКОЙ ОБЛАСТИ

ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ

«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СТРОИТЕЛЬСТВА»

(ГБУ КО ПОО «КИТиС»)

|  |  |  |
| --- | --- | --- |
| Курсовой проект  допущен к защите  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (дата)  Зам.директора по УМР  \_\_\_\_\_\_\_\_Павленко Г.Я.  (подпись) |  | Курсовой проект  защищен с оценкой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  Руководитель работы  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) |

Пояснительная записка к курсовому проекту

по дисциплине: МДК 01.01 Разработка программных модулей

Тема Разработка игрового приложения «Шарики»

Специальность 09.02.07 «Информационные системы и программирование»

|  |  |
| --- | --- |
| Выполнил: | студент 3 курса,  группы ИСп19-2к  Кашин Дмитрий Игоревич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |
| Руководитель: | Большакова-Стрекалова Анна Викторовна, преподаватель ГБУ КО ПОО КИТиС  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Калининград

2022

Содержание

[Введение 3](#_Toc101951928)

[1 Описание предмотной области 4](#_Toc101951929)

1.1 Аналоги разрабатываемого приложения [5](#_Toc101951930)

[1.2 Техническое задание 7](#_Toc101951931)

[1.3 Основание для разработки 9](#_Toc101951932)

[2 Описание разработки приложения 10](#_Toc101951934)

[2.1 Обоснование средств разработки 11](#_Toc101951935)

[2.2 Разработка интерфейса 14](#_Toc101951936)

[2.3Разработка логики работы приложения.Схема взаимодействия компонентов проектов 17](#_Toc101951938)

[2.4Описание переменных, компонентов, класов подклассов 18](#_Toc101951938)

[3 Тестирование и установка приложения 20](#_Toc101951934)

[Заключение 23](#_Toc101951934)

[Список использованной литературы 24](#_Toc101951934)

[Приложения А. Листинг кода 25](#_Toc101951934)

Введение

Игра основана на играх серии [Breakout](https://ru.wikipedia.org/wiki/Breakout_(%D0%B8%D0%B3%D1%80%D0%B0)) фирмы [Atari](https://ru.wikipedia.org/wiki/Atari). Именно её название стало нарицательным для класса подобных игр.

Игрок контролирует небольшую платформу-ракетку, которую можно передвигать горизонтально от одной стенки до другой, подставляя её под шарик, предотвращая его падение вниз. Удар шарика по кирпичу приводит к разрушению кирпича. После того как все кирпичи на данном уровне уничтожены, происходит переход на следующий уровень, с новым набором кирпичей. Есть и некоторое разнообразие: определённые кирпичи нужно ударять несколько раз, иногда появляются летающие враги, от которых отталкивается шарик, удар по некоторым кирпичам приводит к выпаданию из них капсул-призов — приз активируется, если поймать такую капсулу ракеткой. [1]

1. Описание предметной области

«Арканоид» — классическая компьютерная игра.

Игра «Арканоид» представляет собой однопользовательскую игру. Предназначена для развлечения пользователей.

Суть игры такая:

- сверху игрового поля стоят несколько рядов блоков;

- по полю движется шарик, который при касании убирает блок и отскакивает в противоположную сторону;

- от стен и верха шарик тоже отскакивает;

- внизу есть подвижная платформа, как ракетка;

- чтобы шарик не упал вниз, игрок двигает платформу влево или вправо, подставляя её под шарик;

- если шарик падает мимо платформы — игра останавливается или заканчивается совсем;

- цель игры — сбить все кирпичи и не дать шарику упасть.

* 1. Аналоги разрабатываемого приложения

— INV — научно-фантастический арканоид с видом сверху. Вам предстоит отстреливать инопланетных вторженцев и спасти Землю от захватчиков;



Рисунок 1 – INV

— Super Destronaut DX — классический арканоид с видом сбоку в научно-фантастическом сеттинге. Игра вдохновлена популярными аркадами из 90-х, где вам предстоит сопротивляться вторжению инопланетных захватчиков. Пилотируя космический штурмовик, вам нужно уничтожить как можно больше пришельцев, при этом зарабатывая очки;

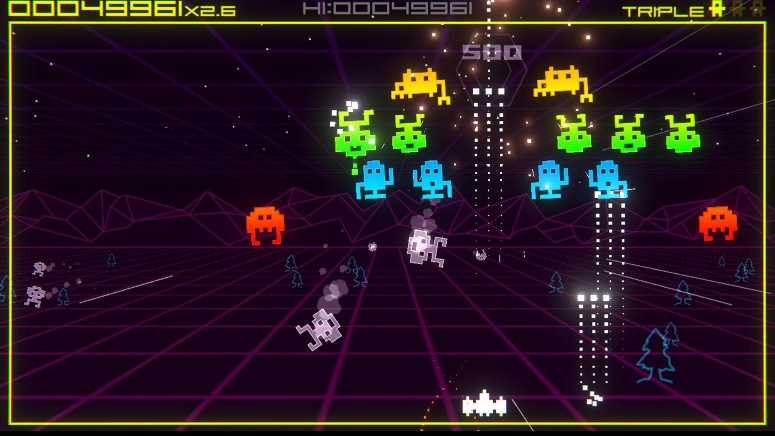


Рисунок 2 — Super Destronaut DX

— 33 Rounds — аркадный экшен с видом сверху в научно-фантастическом сеттинге, в котором вы будете управлять небольшим звездолетом. Главная цель — уничтожать врагов и уклоняться от снарядов. В игре есть 33 уровня, возможность прокачивать характеристики своего корабля и четыре уровня сложности. [2]



Рисунок 3 — 33 Rounds

* 1. Техническое задание

## Данное техническое задание предназначено для разработки игры «Арканоид», которая является развлекательным приложением под операционную систему Windows, позволяющим хорошо проводить свободное время.

Вся игра — это один большой цикл, где каждое новое положение элементов отрисовывается в новом кадре. Последовательность действий будет такая:

1. Очищаем игровое поле.
2. Сдвигаем платформу, если было нажатие на стрелки, и следим, чтобы она не улетела за границы стен.
3. Двигаем шарик со своей скоростью и направлением и смотрим, чтобы он отскакивал от стен и верхней границы.
4. Если шарик упадёт вниз мимо платформы — помещаем его на стартовую позицию и останавливаем, пока не игрок не нажмёт Enter.
5. Если шарик коснулся платформы — делаем отскок в противоположную сторону.
6. Когда шарик коснётся блока — тоже делаем отскок и сразу убираем этот блок.

Требования к составу выполняемых функций

Игра «Арканоид» должна выполнять следующие функции:

* Двигать шарик со своей скоростью и направлением и смотреть, чтобы он отскакивал от стен и верхней границы;
* Движение платформы при нажатии кнопок на клавиатуре;
* Разбивать блоки шариком;
* Развивать реакцию пользователя;

# Программа не должна самостоятельно завершаться или выдавать ошибку при корректных действиях пользователя.

Системные требования

# Для функционирования программы «Арканоид» необходим компьютер PC совместимый и следующие технические средства:

# процессор Intel или совместимый;

# стандартная клавиатура;

* + мышка.
  1. Основание для разработки

Документ, на основании которого ведется разработка

Задание на курсовую работу по дисциплине «Разработка программных модулей».

Игра для ПК «Арканоид» предназначена для игрового времяпровождения. Вовремя, которого пользователям предоставляется управлять платформой в браузере, которая управляет шариком для разбития блоков.

## Описание разработки приложения

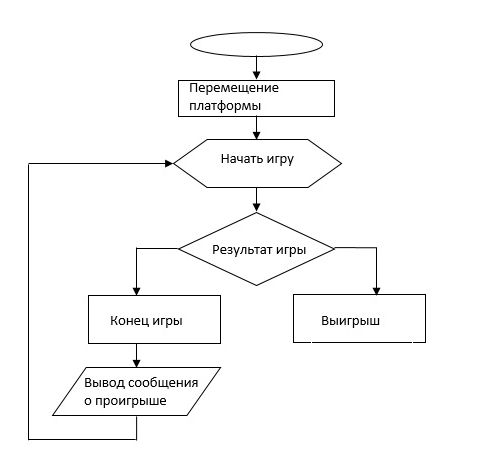
****

Рисунок 4 – Блок Схема

* 1. Обоснование средств разработки

Выбор операционной системы

Windows — это операционная система, сделанная корпорацией Microsoft (Майкрософт). Операционная система (ОС) — это главная программа, которая запускается при включении компьютера. Она позволяет пользователям компьютера работать с файлами, пользоваться Интернетом и запускать в окошках другие программы, игры, фильмы, музыку. Windows переводится как окна. [3]

Выбор среды разработки:

Visual Studio Code – это один из наиболее популярных редакторов кода, разработанный корпорацией Microsoft. Он распространяется в бесплатном доступе и поддерживается всеми актуальными операционными системами: Windows, Linux и macOS. VS Code представляет собой обычный текстовый редактор с возможностью подключения различных плагинов, что дает возможность работать со всевозможными языками программирования для разработки любого IT-продукта.

### Возможности **Visual Studio Code**

* Встроенные инструменты интеграции с GitHub, GIT, а также Visual Studio Team Services для быстрого тестирования, сборки, упаковки и развертывания разных типов приложений;
* Удобство работы с Unity-проектами;
* Аабота с Mono и Node.js с помощью встроенного отладчика;
* Поддержка TypeScript и JavaScript;
* Публикация созданных приложений в Microsoft Azure через сервис Visual Studio Team Services;
* Поддержка практически всех языков программирования;
* Написание кода для конкретной задачи с его последующей интеграцией в проект (с надстройкой или напрямую);
* Обширная библиотека шаблонов, готовых фрагментов кода и сниппетов с возможностью добавления своих элементов;
* Одновременная работа с несколькими проектами (в нескольких окнах);
* Интерфейс можно разделить на две панели для сравнения кода;
* Функция отладки.

### Особенности **Visual Studio Code**

* VS Code позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ. [4]
* В редакторе присутствуют встроенный отладчик, инструменты для работы с Git и средства рефакторинга, навигации по коду, автодополнения типовых конструкций и контекстной подсказки.
* Продукт поддерживает разработку для платформ ASP.NET и Node.js, и считается легковесным решение, которое позволяет обойтись без полной интегрированной среды разработки.
* Большим плюсом редактора является поддержка большого количества языков, таких как C++, C#, Python, PHP, JavaScript и других.

### Преимущества **Visual Studio Code**

* множество настроек (как всей программы, так и интерфейса);
* расширяемая библиотека дополнений и готовых решений;
* мультифункциональность (редактор поддерживает почти все языки, используемые для создания приложений);
* простота и гибкость. [4]

Выбор языка программирования:

JavaScript (JS) — высокоуровневый язык программирования, который поддерживает императивный, функциональный, событийно-ориентированный и другие подходы. Относится к языкам с динамической типизацией, входит в группу интерпретируемых языков.

В число основных особенностей JS входят:

* Динамическая типизация — тип данных определяется в момент присваивания значения константе или переменной.
* Интерпретируемый язык — код приложения интерпретируется при обращении, не требуется предварительная компиляция.
* Функции как объекты первого класса, то есть функции в JavaScript можно возвращать из функций, передавать в качестве параметров в другие функции, [присваивать переменным](https://ru.hexlet.io/courses/js-functions/lessons/first-class-citizen/theory_unit).
* Поддержка портативного и объектно-ориентированного подхода.
* Универсальность — все популярные браузеры поддерживают JavaScript. [5]

## 2.2 Разработка интерфейса

Пользовательский интерфейс — это средства общения пользователя с вашей программой, которые могут включать в себя изображения, звуки и текст. Ориентируясь на среднего пользователя, интерфейс должен быть простым и удобным. Это снизит вероятность ошибок. [6]

Интерфейс состоит из 5-х компонентов Image, 1 компонентом text. Картинки в интерфейсе: background, ball, platform, block1, block2

Image:

Platform – png картинка платформа (будет двигаться и управлять шариком)



Рисунок 5 – платформа

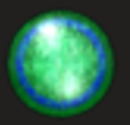
Ball – png картинка щариком (будет разбивать блоки под управлением платформы)

Рисунок 6 – шарик

Background - png картинка фон

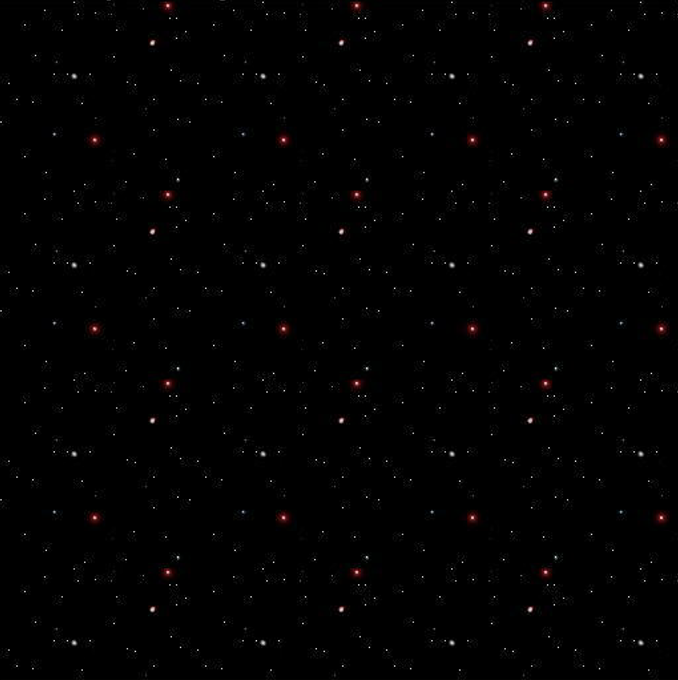


Рисунок 7 – фон

Block1 – png картинка блока



Рисунок 8 – блок1

Block2 – png картинка блока



Рисунок 9 – блок2

Text: после того как шарик не попал на платформу будет выведено сообщение об окончании игры и надо будет нажать Enter для повторной попытке



Рисунок 10 – конец игры

2.3 Разработка логики работы приложения. Схема взаимодействия компонентов проекта

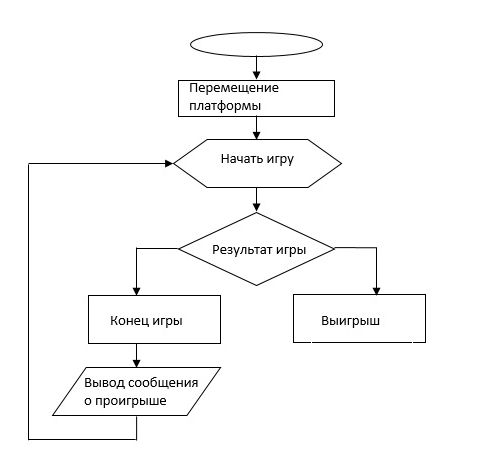
****

Рисунок 11 – Блок схема

2.4 Описание переменных, компонентов, классов и подпрограмм

Таблица 1 – Переменные

|  |  |
| --- | --- |
| Переменные | Описание |
| Let | позволяет объявить локальную переменную с областью видимости, ограниченной текущим блоком кода. |
| Const | создаёт константу (новую именованную ссылку на область памяти), доступную только для чтения. |

Таблица 2 – Функции

|  |  |  |
| --- | --- | --- |
| Функции | | Описание |
| function drawBall(ball) | | Рисует мяч |
| function drawBlock(block) | | Рисует блок |
| function drawPlaforma(platforma) | | Рисует платформу |
| function drawResult() | | Вывод информации |
| function clearCanvas () | | Очищает канвас |
| function drawRect (param) | Функция рисует прямоугольник с параметрами из объекта param | |
| function isIntersection (blockA, blockB) | Функция возвращает true, если произошло столкновениме 2-х блоков | |
| function getRandomFrom (array) | Функция будет возвращать случайное число из массива | |
| requestAnimationFrame | Функция обновляет канвас | |
| function toggleItem | Функция удаления | |

Таблица 3 – Методы

|  |  |
| --- | --- |
| Методы | Описание |
| document. querySelector('canvas') | Обращение к дом элементу |
| Object.assign() | Используется для копирования значений всех собственных перечисляемых свойств из одного или более исходных объектов в целевой объект. |

1. Тестирование и установка приложения

Запуск программы осуществляется путем запуска исполняемого файла index.html из директории программы на жестком диске.

Когда страница теста включает блокировку, рабочий стол учащегося блокируется, а приложение запускается над экраном блокировки Windows. Таким образом создается песочница, которая гарантирует, что учащийся может взаимодействовать только с приложением "Тестирование". После перехода к экрану блокировки приложение "Тестирование" применяет локальные политики MDM для дополнительной блокировки устройства. Весь процесс запуска над экраном блокировки и применения политик — это и есть блокировка. Процесс блокировки является атомарным. Это означает, что в случае сбоя какой-либо части операции блокировки приложение не будет заблокировано и не будет применять ни одной из политик. При запуске над экраном блокировки:

* Приложение работает в полноэкранном режиме без хрома;
* Аппаратная клавиша PRINT SCREEN отключена;
* В зависимости от параметра, заданного через схему или выделенную учетную запись, содержимое приложения может быть скрыто в программах для захвата экрана;
* Системный буфер обмена очищен;
* Веб-приложения могут отправлять запросы процессам, работающим на устройстве пользователя;
* Расширенный экран отображается черным; авто заполнение отключено.

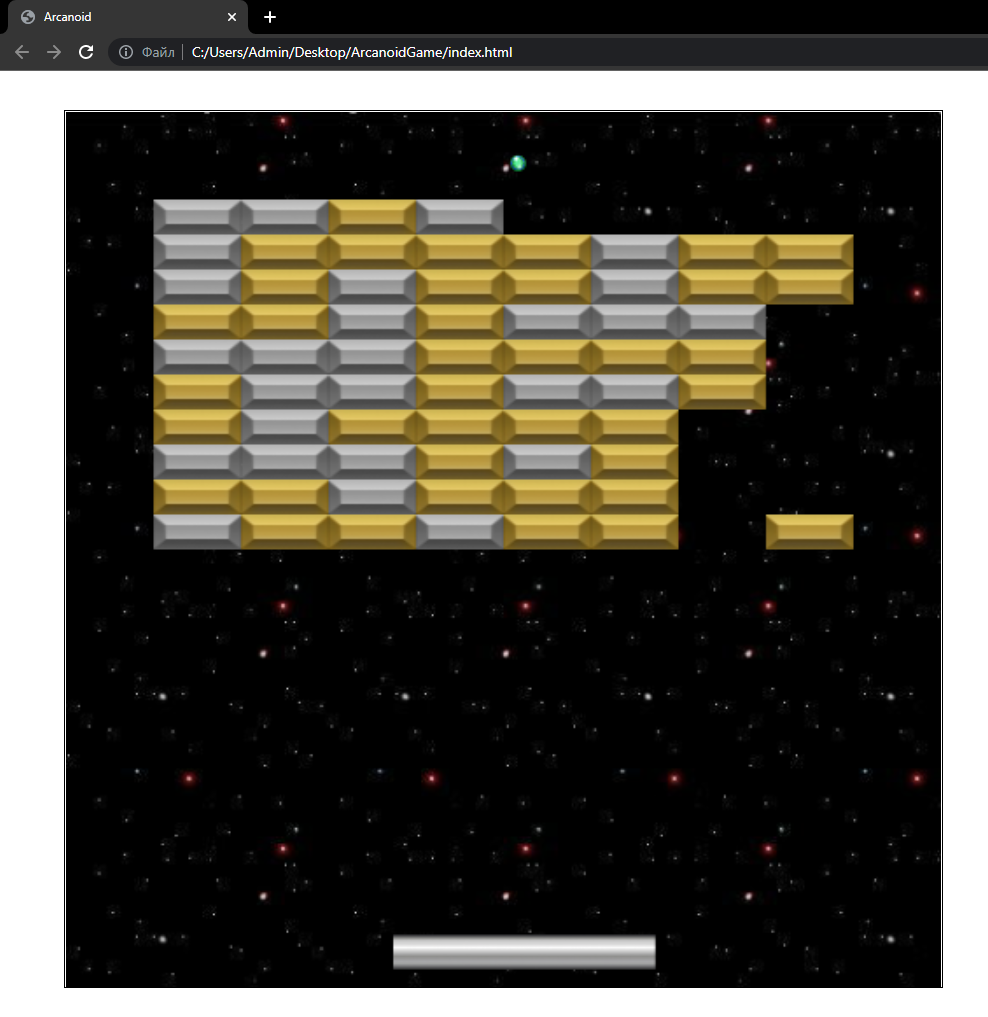
****

Рисунок 12 – проверка работоспособности игры

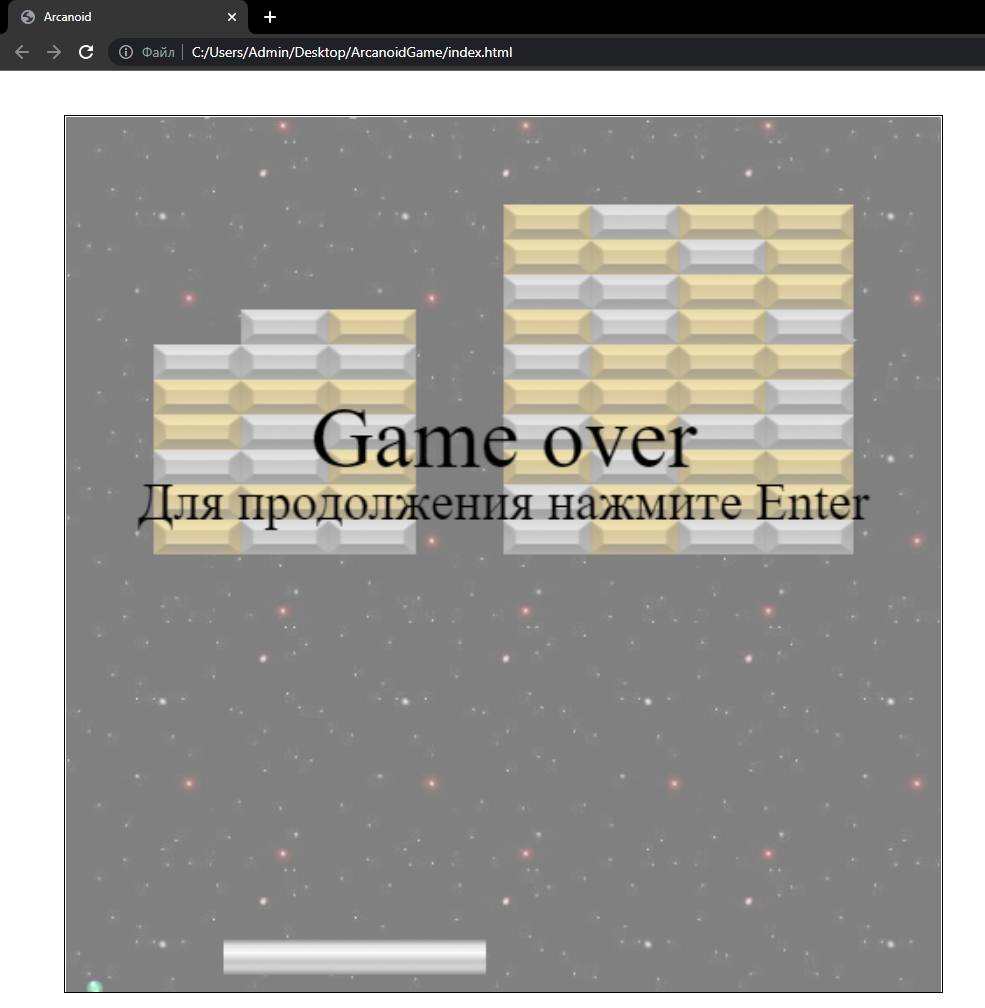


Рисунок 13 – Конец игры

Программа успешно прошла тестирование

Заключение

Была поставлена и достигнута цель работы, состоявшая в «Разработке игрового приложения «Шарики». в указанный период, путем решения следующих задач:

* Движение платформы, если было нажатие на стрелки;
* Движение шарика со своей скоростью и направлением;
* Шарик падает вниз мимо платформы— помещаем его на стартовую позицию и останавливаем, пока не игрок не нажмёт Enter;
* Если шарик коснулся платформы — делаем отскок в противоположную сторону;
* Когда шарик коснётся блока — тоже делаем отскок и сразу убираем этот блок.

Список использованной литературы

1. [Википедия арканоид](https://ru.wikipedia.org/wiki/Arkanoid). (https://ru.wikipedia.org/wiki/Arkanoid)
2. [Аналоги арканоида](https://vgtimes.ru/games/arkanoid/similar/). ( https://vgtimes.ru/games/arkanoid/similar/)
3. [Операционная система](https://ru.wikipedia.org/wiki/Windows).( https://ru.wikipedia.org/wiki/Windows)
4. [Среда разработки](https://ru.education-wiki.com/3958588-what-is-visual-studio-code).

( https://ru.education-wiki.com/3958588-what-is-visual-studio-code)

1. [Язык программирования](https://habr.com/ru/company/ruvds/blog/445272/). (https://habr.com/ru/company/ruvds/blog/445272/)
2. [Интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). (https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81\_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F)
3. Этан Браун, изучаем JavaScript – 2017 г.
4. Кайл Симпсон, Типы и грамматические конструкции JS 2015 г.

Приложение А. Листинг кода

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

</title><title>Arcanoid</title>

<style>

canvas {

border: 1px solid black;

margin: 25px auto;

display: block;

}

</style>

</head>

<body>

<canvas></canvas>

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

</body>

</html>

const canvas = document.querySelector('canvas') // обращение к дом элементу

const context = canvas.getContext('2d')

// данные канваса

canvas.width = 500

canvas.height = 500

// Добавляем изображение

let image = new Image

image.src = "image.png"

// Добавляем изображение

let space = new Image

space.src = "space.jpg"

// координаты мяча, платформы, блоков

const atlas = {

ball: { x: 4, y: 588, width: 36, height: 36 },

grey: { x: 232, y: 0, width: 42, height: 20 },

brown: { x: 232, y: 36, width: 42, height: 20 },

platforma: { x: 109, y: 177, width: 208, height: 15 }

}

const ball = {

x: canvas.width / 2,

y: canvas.height - 50,

width: 10,

height: 10,

speed: 200,

angle: Math.PI / 4 + Math.random() \* Math.PI / 2

}

const platforma = {

x: canvas.width / 2 - 100,

y: canvas.height - 30,

width: 150,

height: 20,

speed: 200,

leftKey: false,

rightKey: false,

}

const blocks = []

// Добавление остальных блоков с рандомным цветом

for(let x = 0; x < 8; x++) {

for (let y = 0; y < 10; y++) {

blocks.push({

x: 50 + 50 \* x,

y: 50 + 20 \* y,

width: 50,

height: 20,

color: getRandomFrom(["grey", "brown"])

})

}

}

// 4 невидисых блока по краям канваса. Расположены с внешней стороны вплотную.

const limits = [

{ x: 0, y: -20, width: canvas.width, height:20 },

{ x: canvas.width, y: 0, width: 20, height: canvas.height },

{ x: 0, y: canvas.height, width: canvas.width, height: 5 },

{ x: -20, y: 0, width: 20, height: canvas.height },

]

// регистрирует определённый обработчик события

// Нажатия клавишей клавиатуры

document. addEventListener("keydown", function (event) {

if (event.key === "ArrowLeft") {

platforma.leftKey = true

}

else if (event.key === "ArrowRight") {

platforma.rightKey = true

}

// Если конец игры нужно нажать Enter

else if (plaing === false && event.key === "Enter") {

plaing = true

// свойство объектов без смен ссылки на этот объект

Object.assign (ball, {

x: canvas.width / 2,

y: canvas.height - 50,

width: 10,

height: 10,

speed: 200,

angle: Math.PI / 4 + Math.random() \* Math.PI / 2

})

// свойство объектов без смен ссылки на этот объект

Object.assign (platforma, {

x: canvas.width / 2 - 100,

y: canvas.height - 30,

width: 150,

height: 20,

speed: 200,

leftKey: false,

rightKey: false,

})

blocks.splice(0, blocks.length - 1)

for(let x = 0; x < 8; x++) {

for (let y = 0; y < 10; y++) {

blocks.push({

x: 50 + 50 \* x,

y: 50 + 20 \* y,

width: 50,

height: 20,

color: getRandomFrom(["grey", "brown"])

})

}

}

}

})

document. addEventListener("keyup", function (event) {

if (event.key === "ArrowLeft") {

platforma.leftKey = false

}

else if (event.key === "ArrowRight") {

platforma.rightKey = false

}

})

requestAnimationFrame(loop)

let pTimestamp = 0

let plaing = true

function loop (timestamp) {

requestAnimationFrame(loop)

clearCanvas ()

if (plaing) {

const dTimestamp = Math.min(16.7, timestamp - pTimestamp)

const secondPart = dTimestamp / 1000

pTimestamp = timestamp

// Движение мячика

ball.x += secondPart \* ball.speed \* Math.cos(ball.angle)

ball.y -= secondPart \* ball.speed \* Math.sin(ball.angle)

// Движение платформы.

// Если должна двигаться влево

if (platforma.leftKey) {

// Подвинуть левее, но не далее границы поля.

platforma.x = Math.max(0, platforma.x - secondPart \* platforma.speed)

}

// Если должна двигаться вправо

if (platforma.rightKey) {

// Подвинуть правее, но не далее границы поля.

platforma.x = Math.min(canvas.width - platforma.width, platforma.x + secondPart \* platforma.speed)

}

// Пройти по всем блокам.

for (const block of blocks) {

// Если мя столкнулся с выбранным блоком

if (isIntersection(block, ball)) {

// Удалить блок

toggleItem(blocks, block)

// Создать 4 контрольных блока чтобы понять, с какой стороны ударил мяч

// Верхний

const ctrl1 = {

x: block.x - 10,

y: block.y - 10,

width: 10 + block.width,

height: 10

}

// Правый

const ctrl2 = {

x: block.x + block.width,

y: block.y - 10,

width: 10,

hight: 10 + block.height

}

// Нижний

const ctrl3 = {

x: block.x,

y: block.y + block.height,

width: block.width + 10,

hight: 10

}

// Левый

const ctrl4 = {

x: block.x - 10,

y: block.y,

width: 10 ,

hight: block.height + 10

}

// Если мяч столкнулся с верхним или нижним вспомогательным блоком

if (isIntersection(ctrl1, ball) || isIntersection(ctrl3, ball)) {

ball.angle = Math.PI \* 2 - ball.angle

}

// Если мя столкнулся с правым или левым вспомогательным блоком

else if (isIntersection(ctrl2, ball) || isIntersection(ctrl4, ball)) {

ball.angle = Math.PI - ball.angle

}

break

}

}

// Если мяч ударил в верхнию платформу

if (isIntersection(limits[0], ball)) {

ball.angle = Math.PI \* 2 - ball.angle

}

// Если мяч ударил в правую или левую платформу

if (isIntersection(limits[1], ball) || isIntersection(limits[3], ball)) {

ball.angle = Math.PI - ball.angle

}

// Траектория полета

if (isIntersection(platforma, ball)) {

const x = ball.x + ball.width / 2

const percent = (x - platforma.x) / platforma.width

ball.angle = Math.PI - Math.PI \* 8 / 10 \* (percent + 0.05)

}

// конец игры

if (isIntersection(limits[2], ball)) {

plaing = false

}

}

// Рисуем мяч

drawBall(ball)

for (const block of blocks) {

// Рисуем блок

drawBlock(block)

}

// Рисуем платформу

drawPlaforma(platforma)

// Вывод информации

if (!plaing) {

drawResult()

}

}

// Функция очищает канвас

function clearCanvas () {

context.drawImage(space, 0, 0, canvas.width, canvas.height)

}

// Функция рисует прямоугольник с параметрами из объекта param

function drawRect (param) {

context.beginPath()

context.rect(param.x, param.y, param.width, param.height)

context.strokeStyle = 'blue'

context.stroke()

}

// Функция возвращает true, если произошло столкновениме 2-х блоков

function isIntersection (blockA, blockB) {

// вершины блока А

const pointsA = [

{ x: blockA.x, y: blockA.y },

{ x: blockA.x + blockA.width, y: blockA.y },

{ x: blockA.x, y: blockA + blockA.height },

{ x: blockA.x + blockA.width, y: blockA.y + blockA.height }

]

// Пройти по всем вершинам блока А

for (const pointA of pointsA) {

// Находиться ли какая либо точка блока А в внутри блока B?

if (blockB.x <= pointA.x && pointA.x <= blockB.x + blockB.width && blockB.y <= pointA.y && pointA.y <= blockB.y + blockB.height) {

return true

}

}

// Вершины блока В

const pointsB = [

{ x: blockB.x, y: blockB.y },

{ x: blockB.x + blockB.width, y: blockB.y },

{ x: blockB.x, y: blockB + blockB.height },

{ x: blockB.x + blockB.width, y: blockB.y + blockB.height }

]

// Пройти по всем вершинам блока В

for (const pointB of pointsB) {

// Находиться ли какая либо точка блока В в внутри блока А?

if (blockA.x <= pointB.x && pointB.x <= blockA.x + blockA.width && blockA.y <= pointB.y && pointB.y <= blockA.y + blockA.height) {

return true

}

}

return false

}

function toggleItem (array, item){

if(array.includes(item)){

const index = array.indexOf (item)

array.splice(index, 1)

}

else {

array.push(item)

}

}

// Вырезаем мяч и рисуем изображение

function drawBall (ball) {

context.beginPath()

context.drawImage(

image,

atlas.ball.x, atlas.ball.y, atlas.ball.width, atlas.ball.height,

ball.x, ball.y, ball.width, ball.height

)

}

// Вырезаем блок и рисуем изображение

function drawBlock (block) {

context.drawImage(

image,

atlas[block.color].x, atlas[block.color].y, atlas[block.color].width, atlas[block.color].height,

block.x, block.y, block.width, block.height

)

}

// Вырезаем платформу и рисуем изображение

function drawPlaforma (platforma) {

context.drawImage(

image,

atlas.platforma.x, atlas.platforma.y, atlas.platforma.width, atlas.platforma.height,

platforma.x, platforma.y, platforma.width, platforma.height

)

}

// Функция будет возвращать случайное число из массива и наоборот

function getRandomFrom (array) {

const index = Math.floor(Math.random() \* array.length)

return array [index]

}

// Вывод информации, затемнение, цвет, шрифт, расположение

function drawResult () {

context.beginPath()

context.rect(0, 0, canvas.width, canvas.height)

context.fillStyle = "rgba(255, 255, 255, 0.5)"

context.fill()

context.fillStyle = "black"

context.font = "50px Times New Roman"

context.textAlign = "center"

context.fillText("Game over", canvas.width / 2, canvas.height / 2 - 50)

context.fillStyle = "black"

context.font = "30px Times New Roman"

context.textAlign = "center"

context.fillText("Для продолжения нажмите Enter", canvas.width / 2, canvas.height / 2 - 20)

}

Рецензия

В работе на тему Разработка игры “Шарики” под управлением ОС Windows, было разработано приложение. Проект состоит из введения, трёх глав, заключения, списка использованных источников и приложения А. Во введении кратко рассказывается о истории возникновения игры, а также отмечены: цель, задачи. В первой главе были проанализированы приложения- конкуренты, разработано техническое задание. Вторая глава посвящена практической части, где обусловлен выбор среды разработки и языка программирования А в третьей главе описывается тестирование приложения В заключении данного курсового проекта развернуто подытожена работа.

Можно сделать вывод, что в целом, автор выполнил работу по всем требованиям. Каждая часть проекта рассматривалась с применением необходимой профессиональной терминологией. Студент излагал работу с логичной последовательностью, все названия глав полностью соответствуют тексту.

Ссылка GitHub https://github.com/inque12/Kursovaya