

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»


Кафедра математической
кибернетики и компьютерных наук

РАЗРАБОТКА ИГРЫ JOHN'S LONG WAY С ИСПОЛЬЗОВАНИЕМ
PHASER.JS И GO

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ


студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Олейника Ильи Андреевича

Научный руководитель
доцент, к. ф.-м. н.


10.06.2017

Ю. Н. Кондратова

Заведующий кафедрой
к. ф.-м. н.


10.06.2017.

С. В. Миронов

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Описание игры	4
2 Серверная часть игры	5
2.1 База данных	5
2.2 Серверное приложение	5
2.2.1 Используемые библиотеки	6
2.2.2 Описание структур и глобальных переменных	6
2.2.3 Функции в приложении	7
3 Клиентская сторона	10
3.1 Загрузка игры	10
3.2 Авторизация	10
3.3 Главное меню	11
3.4 Магазин	11
3.5 Страница достижений	11
3.6 Страница статистики	11
3.7 Игровой процесс	11
3.8 Тестирование	13
ЗАКЛЮЧЕНИЕ	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	15

ВВЕДЕНИЕ

Разработка игр в настоящее время всё развивается и у неё есть огромный потенциал. Компьютерными играми интересуются и взрослые, и дети. Кроме того, они используются не только для развлечения, но и для обучения, например, развивающие игры для детей. Недавно, в России компьютерные игры официально признаны спортом. Вместе с повышением спроса на компьютерные игры, возрастает набор инструментов для создания различного рода игр.

Общая постановка задачи: исследовать возможности игрового движка Phaser при создании игровых приложений. Для решения этой задачи необходимо:

- Реализовать интерфейсы для игровых объектов;
- Создать основной геймплей;
- Сделать основные экраны (меню, конец игры, статистика, достижения);
- Спроектировать и реализовать базу данных;
- Написать серверную часть для сохранения профилей;
- Привязать клиент к серверу.

1 Описание игры

Жанр runner существует уже много лет и получает всё большую популярность. Одним из первых игр в жанре runner является игра PepsiMan.

Самую большую популярность игры в жанре runner получили на мобильных устройствах, таких как смартфоны и планшетные компьютеры. Одной из самых популярных игра в данном жанре для мобильных устройств является Subway Surfers.

В данной игры необходимо избегать столкновений с различными объектами и собирать монеты.

А одной из последних игр в этом жанре стала Super Mario Run. Причём изначально эта серия была платформером (жанр игр, в котором надо прыгать по платформам, чтобы достичь определённой цели и выхода из уровня). Именно смесь из этих двух жанров представляет игра, которая была разработана в рамках этой работы.

Данная игра реализована именно в жанре runner, и в ней персонажу необходимо прыгать по платформам, при этом не касаясь пола. При касании пола у персонажа отнимается одна жизнь. При старте игры у персонажа есть 3 жизни, их можно подбирать на платформах, но нельзя взять более 3 жизней одновременно. При касании пола персонаж резко подпрыгивает вверх и становится доступен второй прыжок, для простоты воскрешения и гарантированного попадания на платформу после смерти.

Все изображения и спрайты являются свободно распространяемыми и были взяты с сайта phaser.io.

2 Серверная часть игры

Игра состоит из клиентской (frontend) и серверной (backend) частей. Клиентская часть отображает пользователю саму игру, а серверная обрабатывает данные, полученные от клиента, такие как результаты прохождения, покупки магазина и авторизация.

2.1 База данных

Одной из частей серверной стороны является база данных, в которой хранятся все данные о пользователях игры, что позволяет играть на нескольких устройствах без потери прогресса и с корректным подсчётом статистики. Используемая СУБД - MySQL, она была выбрана из-за её открытости и удобства использования.

В таблице users хранятся данные для авторизации пользователей, а также статус купленных предметов и уникальный идентификатор. Данные для авторизации – имя пользователя, хэш пароля и соль, последние два из которых нужны для проверки пароля, чтобы не хранить его в открытом виде, из-за этого даже при краже данных из БД злоумышленники не смогут узнать реальные пароли пользователей. Текущее количество золота хранится в таблице money в виде целочисленного значения. Статус купленных предметов представляет собой строку из трёх цифр от 1 до 3. Так как предметы можно покупать только последовательно, 1 – куплен первый предмет, 2 – куплен первый и второй предмет, 3 – куплен первый, второй и третий предметы. Первое число означает купленные фоны, второе – платформы, третье – персонажи.

В таблице token хранятся токены авторизации, связи между уникальными идентификаторами и случайно сгенерированным токеном. На стороне клиента хранится только токен, что позволяет осуществлять безопасное определение пользователя.

В таблице records хранятся результаты игр: уникальный идентификатор игры, идентификатор пользователя, пройденное расстояние и полученные монеты. В дальнейшем эти данные будут использоваться для подсчёта достижений пользователя.

2.2 Серверное приложение

Сервер написан на языке программирования go [1, 5, 12–15] и скомпилирован с помощью компилятора go 1.8.1 (последняя, на данный момент,

версия).

2.2.1 Используемые библиотеки

В серверном приложении используется множество стандартных библиотек go, такие как:

- `crypto/sha256` - для шифрования паролей алгоритмом sha256 для обеспечения безопасности паролей
- `encoding/hex` - для перевода хэшей в шестнадцатичную систему для удобства хранения
- `encoding/json` - для парсинга и закидывания в переменную файла конфигурации, а также для чтения запроса и ответа на него
- `errors` - для создания сообщений об ошибке
- `flag` - для считывания флагов командной строки
- `fmt` - для вывода информации в поток вывода http сервера
- `io/ioutil` - для работы с файлом конфигурации
- `log` - для логирования информации в консоль приложения
- `math/rand` - для создания случайных токенов
- `net/http` - http сервер
- `time` - работа со временем [11]

Также используются библиотеки с открытым исходным кодом для более удобной работы с базой данных:

- `github.com/go-sql-driver/mysql` – как драйвер для MySQL [4]
- `github.com/jmoiron/sqlx` для упрощения работы БД с сложными структурами данных. [3]

2.2.2 Описание структур и глобальных переменных

Стоит начать с переменных, так как они обеспечивают работу программы и связь с базой данных:

- `config` - служит для хранения данных конфигурации, которые получаются из json файла, подробнее это будет описано в разделе функций.
- `configFile` - представляет из себя флаг, который можно указать при старте приложения из консоли, по умолчанию он указан как `conf.json`, в нём должны храниться данные авторизации для базы данных.

В программе используется множество структур данных, которые необходимы для обработки json запросов и отправки json ответов, а также для

работы с базой данных. Список структура приведён ниже:

- `userInfo` - нужна для записи данных о пользователе в базу данных и состоит из логина, хэша пароля и соли. При этом, при регистрации и авторизации пользователю будет отправлена переменная структуры данных `token`, которая содержит только токен авторизации, которого достаточно для однозначной идентификации пользователя
- `responseData` - нужна для передачи данных о завершённой игре от клиентской части к серверной. Она содержит в себе токен авторизации, пройденное расстояние и собранные монеты. Ответ возвращается в виде переменной структуры `Response`, в которой содержится код ответа и состояние, код будет равен 200 при успешно выполненной команде.
- `LoginData` - нужна для авторизации пользователя, при вводе логина и пароля происходит авторизация, или возвращается ошибка.
- `database` - нужна для хранения соединения с базой данных. Все хандлеры являются функциями этой структуры. Это позволяет держать одно соединение с базой данных на приложение, что позволяет экономить ресурсы.
- `Records` - нужна для передачи страницы рекордов пользователей. Она содержит в себе результат и имя пользователя.
- `Items` - нужна для отображения на клиентской части страницы магазина, она соержжит в себе состояние предметов и количество монет для пользователя. Для покупки предметов используется структура `BuyItems`, которая принимает токен авторизации и порядковый номер предмета, который пользователь хочет купить.
- `AchievementsStatus` - нужна для отображения достижений пользователя и содержит в себе данные статистики о пользователе, такие как общее количество заработанных монет, количество игр, общая дистанция и максимальная дистанция.

2.2.3 Функции в приложении

В ходе разработки было написано множество функций. Функции, которые заканчиваются на слово `Handler` выполняют роль веб-страниц, которые принимают `json` данные и отдают ответ. Большая часть функций нужна для работы с данными и базой данных, чтобы пользователь мог играть сразу на нескольких устройствах без потери прогресса. Список функций:

- resultHandler - страница результата, которая принимает json данные со структурой resultData, вызывает функции записи в базу данных и возвращает json данные структуры Response в качестве ответа.
- submitResult - принимает токен пользователя, пройденное расстояние и количество заработанных монет. Записывает эти данные в таблицу рекордов и прибавляет монеты пользователю игры. Возвращает nil, если ошибки нет, или ошибку.
- loadConfig - загружает конфигурационный файл в переменную config. Принимает путь до файла с конфигурацией. Возвращает nil, если ошибки нет, или ошибку.
- generateSalt - генерирует 3х буквенный код для шифрования пароля.
- generateToken - генерирует 30 символьный ключ авторизации для пользователя
- GetUserFromDBByLogin - запрашивает из базы данных уникальный идентификатор юзера по его логину
- GetUserSaltFromDBById - запрашивает из базы данных захешированный пароль и соль по идентификатору пользователя
- AddUserToDB - добавляет пользователя в базу данных
- hashingPassword - хэширует пароль [2]
- registerHandler - страница регистрации пользователя. Принимает json данные структуры LoginData, проводит проверки на длину логина и пароля (больше 5 символов), генерирует соль, хэширует с её помощью пароль, создаёт сессию и возвращает клиенту ключ авторизации.
- authHandler - страница авторизации пользователя. Принимает json данные структуры LoginData, проводит проверки существования логина и правильности пароля и возвращает клиенту токен.
- GenerateSession - создаёт в базе данных ключ авторизации и привязывает его к идентификатору пользователя.
- GetStat - запрашивает из базы данных статистику лучших результатов
- getStatisticsHandler - страница статистики, возвращает топ лучших результатов - массив структуры Records.
- GetItems - запрашивает из базы данных данные о предметах пользователя. Используется для отображения магазина.
- buyItemsHandler - страница покупки предметов. Принимает json данные

структуры `BuyItems` и возвращает ошибку или новое состояние предметов.

- `getItemsHandler` - страница запроса предметов пользователя. Принимает ключ авторизации, а возвращает данные о предметах пользователя и количество его монет.
- `getAchievementsInfo` - запрашивает из базы данных информацию о достижениях пользователя.
- `getAchievementsHandler` - страница достижений, принимает токен, а возвращает клиенту информацию о: всего заработанных монетах, количестве игр, общую дистанцию и лучший результат.
- `main` - основная функция приложения. В ней происходит вызов функции загрузки конфигурации, соединение с базой данных, настройка и запуск `http` сервера.

3 Клиентская сторона

Клиентская часть написана на JavaScript с использованием игрового фреймворка `phaser.js` [7].

Приложение состоит из самого кода игры (в папке `src`), статического контента, такого как спрайты, стили и внешние библиотеки. Также в корке проекта есть `gulpfile.js` и `package.json`, которые используются для сборки и запуска проекта.

Точка старта программы – файл `index.js`. В нём происходит подключение файлов со стейтами, определение окна игры далее происходит запуск стейта `boot`.

3.1 Загрузка игры

Экран загрузки является начальным этапом подготовки игры к запуску. На нём запускаются проверки на существование `cookies` для отображения интерфейса игры. В них хранятся текущие значения фонов, игрока и платформ. После этого стейта запускается стейт `preload`, который загружает графику игры, то есть 3 фона, 3 спрайта персонажей, 3 платформы, фон меню, монеты и сердца разных размеров.

Графика в игре выполнена в стиле пиксель-арта. Второй и третий персонажи в спрайте содержат только анимацию движения вперёд, так как в игре не предусмотрено движение назад, но для совместимости с первым спрайтом и, чтобы не использовать лишний код, было решено второго и третьего персонажа начинать рисовать только с середины спрайта. Это, в итоге, позволило сократить код и сделать его более красивым.

3.2 Авторизация

При загрузке страницы авторизации происходит проверка авторизации и, если пользователь уже авторизован, происходит загрузка меню игры. Если же пользователь ещё не авторизован, ему отображается страница авторизации. Страница авторизации, как и регистрации, использует для полей ввода библиотеку `phaser-input.js`. Они состоят из 2 полей – логин и пароль. [6] Как при успешной авторизации, так и при успешной авторизации пользователя перекинет в меню игры.

В обоих случаях будет произведён запрос данных с сервера и проверка ответа. Если возникнет ошибка авторизации, она будет показана пользователю.

лю.

3.3 Главное меню

После успешной авторизации будет показано меню игры. Оно состоит из 4 кнопок: начало игры, магазин, достижения и статистика. [8] Этот экран является связующим звеном всех основных компонентов приложения.

3.4 Магазин

В магазине можно купить и выбрать элементы интерфейса игры. Они играют только косметическую роль для уравнивания пользователей игры.

Купить можно только следующий по порядку предмет. Первые предметы доступны по умолчанию, вторые стоят 1 000 монет, а третьи – 10 000 монет. Чтобы купить все улучшения, необходимо 33 000 монет. Монеты зарабатываются во время игры.

Использование косметических улучшений позволяет пользователю лучше ассоциировать себя с персонажем.

3.5 Страница достижений

Страница достижений представляет из себя список достижений, полученных игроком во время игры. Это должно стимулировать игрока продолжать играть до получения всех достижений, не все из которых лёгкие.

3.6 Страница статистики

Страница статистики представляют из себя список пользователей с лучшими результатами и итоговый результат. Это, также, как и достижения, должны стимулировать людей продолжать играть лучше, чтобы попасть в этот список.

3.7 Игровой процесс

Перед началом игры происходит выбор сложности. Сложности отличаются скоростью персонажа и гравитацией. Чем сложнее, тем быстрее скорость и выше гравитация. [10]

После выбора уровня сложности начинается сама игра. Она представляет из себя runner, в котором надо прыгать по платформам и не коснуться пола.

В начале игры у игрока есть 3 жизни, по их истечении персонаж умирает, и игра заканчивается. На уровне можно поднять как монеты, так и жизни, но нельзя собрать больше 3 жизней, что добавляет дополнительную сложность. Если персонаж упал на пол и у него ещё есть жизни, он сильно подпрыгивает вниз и получает возможность ещё одного прыжка. При прыжке с платформы также доступен двойной прыжок. В процессе тестирования на среднем уровне сложности был поставлен рекорд в 58146 очков. Количество монет на платформе определяется рендомом, как и вероятность дополнительной жизни на платформе. После конца жизней показывается экран результата и данные о прохождении отправляются на сервер.

На экране результатов можно начать заново с той же сложностью или выйти в главное меню игры.

3.8 Тестирование

В процессе разработки было проведено ручное тестирование игры. Найденные ошибки были исправлены и в данный момент новых обнаружено не было. Всего было сыграно около 300 игр, при этом максимальный результат, который был достигнут составляет 58146 - расстояние в пикселях от начала игры. Максимально было получено монет за игру - 200.

ЗАКЛЮЧЕНИЕ

В ходе разработки игры были изучены такие языки программирования, как: HTML5, JavaScript, Go. Также, был изучен фреймворк phaser.js. Были достигнуты поставленные цели: была спроектированная база данных, разработаны сервер и клиент игры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Mark Summerfield Programming in go / М. Summerfield // Addison-Wesley / 2015
- 2 sha256 [Электронный ресурс] / 2017 URL: <https://golang.org/pkg/crypto/sha256/> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 3 Sqlx [Электронный ресурс] / 2017 URL: <https://github.com/jmoiron/sqlx> (Дата обращения: 21.05.2017). Загл. с экр. Яз. англ.
- 4 Mysql [Электронный ресурс] / 2017 URL: <https://github.com/go-sql-driver/mysql> (Дата обращения: 21.05.2017). Загл. с экр. Яз. англ.
- 5 Documentation – The go programming language [Электронный ресурс] / 2017 URL: <https://golang.org/doc/> (Дата обращения: 21.05.2017). Загл. с экр. Яз. англ.
- 6 Phaser-input [Электронный ресурс] / 2017 URL: <https://github.com/orange-games/phaser-input> (Дата обращения: 21.05.2017). Загл. с экр. Яз. англ.
- 7 Phaser [Электронный ресурс] / 2017 URL: <http://phaser.io/> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 8 Phaser buttons [Электронный ресурс] / 2017 URL: <http://phaser.io/examples/v2/category/buttons> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 9 Phaser animation [Электронный ресурс] / 2017 URL: <http://phaser.io/examples/v2/category/animation> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 10 Phaser physics [Электронный ресурс] / 2017 URL: <http://phaser.io/examples/v2/category/arcade-physics> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 11 Golang packages [Электронный ресурс] / 2017 URL: <https://golang.org/pkg/> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.

- 12 Go недостатки [Электронный ресурс] / 2017 URL: <http://bolknote.ru/2011/06/06/3258> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 13 О плюсах и минусах Go [Электронный ресурс] / 2017 URL: <https://habrahabr.ru/post/229169/> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 14 Почему Go не так хорош [Электронный ресурс] / 2017 URL: <https://habrahabr.ru/post/228849/> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 15 За что ругают Go [Электронный ресурс] / 2017 URL: <https://habrahabr.ru/post/282588/> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 16 Тайловая графика в Phaser.js и принципы построения тайловых карт [Электронный ресурс] / 2017 URL: <https://habrahabr.ru/post/236809/> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 17 MMO на Phaser и Node.js [Электронный ресурс] / 2017 URL: <https://habrahabr.ru/post/330058/> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 18 HTML MDN [Электронный ресурс] / 2017 URL: <https://developer.mozilla.org/en-US/docs/Web/HTML> (Дата обращения: 20.05.2017). Загл. с экр. Яз.рус.
- 19 CSS docs [Электронный ресурс] / 2017 URL: <https://www.w3schools.com/css/> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 20 JS docs [Электронный ресурс] / 2017 URL: <https://www.w3schools.com/js/> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.
- 21 Go programming language sources [Электронный ресурс] / 2017 URL: <https://github.com/golang/go> (Дата обращения: 20.05.2017). Загл. с экр. Яз. англ.

Авт
10.06.2017