# Разработка и реализация алгоритмов деформирования трехмерной геометрии анатомии человека на основе WebGL

Андрей Лушников

Апрель 2012 год

# Содержание

# 1 Введение

Программа - плохо, SAAS - хорошо!

# 2 Постановка цели

Скопировать текст из тезисов - зря писал что ль?

# 3 Архитектура

## 3.1 Описание архитектуры

Рассказать про то, что было выбрано написать Web-приложение с развитой серверной частью. В целом рассказть про архитектуру: HTML5 + WebGL, node.js, Thrift. Перечислить достоинства, которые мы получили от использования этих технологий, а так же обсудить некоторые недостатки.

## 3.2 Альтернативы

### 3.2.1 Flash на стороне пользователя

Рассказать почему Flash на стороне клиента не очень хорош как для создания пользовательского интерфейса, так и в свете применения многочисленных 3D-движков. В то же время не забыть про его достоинства: например, по нему очень много материалов и на нем просто сделать богатую и интересную графику.

### 3.2.2 unity3D вместо webGL

В принципе ок вариант: мощный такой 3Д движок, десктопная и мобильная версии которого получили большое распространение. Однако мобильная версия требует установки плагина, что не очень понравилось (по тем же причинам, что и flash)

### 3.2.3 Ruby on Rails вместо node.js

Хорошая штука! Но отдельные усилия нужны для создания и поддержания протокола общения клиента с сервером (разные языки - конвертация!)

### 3.2.4 Protocol Buffers вместо Apache Thrift

Тут рассказать почему выбрали второе а не первое (есть в письме к Саше)

# 4 Детали реализации

## 4.1 Клиентские технологии

### 4.1.1 Движок для 3D графики

Тут надо написать про Three.js, про то, что он является оберткой над специальном языком для шейдеров и не только, и что у него есть два уровня: уровень WebGL, или уровень графической карты, на котором можно делать любые вещи, которые позволены вершинным и пиксельным шейдерами, а так же уровнем js-кода, который оборачивает это безобразие в высокоуровневые конструкции языка javascript

### 4.1.2 Инструменты для обработки графики

Тут надо написать про шаблон Strategy при использовании тулов для обработки графики

### 4.1.3 Тесселляция

Тут надо написать про тесселляцию из движка three.js, про то что она не очень здорово работала и что обсуждение проблем было поднято в сообществе, однако результатов оно не дало

### 4.1.4 Проблема поворота

Тут надо написать про проблему с поворотом объекта, про разные координатные системы и про решение этой проблемы ввиде перемножения матриц

### 4.1.5 Шина событий

Тут надо рассказать про EventBus, про обеспечение двунаправленных коммуникаций между модулями приложения, а так же почему этот подход можно считать достаточно удачным.

### 4.1.6 Инструмент “Деформация”

Тут надо написать про инструмент деформация, рассказать про его формулы, а так же не забыть его доделать так, чтобы он не изменял те вершины, что находятся в другом направлении от направления деформации

### 4.1.7 Использование технологии AJAX

Тут надо написать про то, что все клиент-серверные взаимодействия сделаны по технологии AJAX, зачем это сделано, и что, в частности, пришлось пойти на уступки и добавить поддержку полных путей для загрузки моделей через url для удобства. Частая ошибка, которая все ломала: относительные, а не абсолютные адреса ссылок у действий.

## 4.2 Серверные технологии

### 4.2.1 Серверная платформа node.js

Тут надо сказать пару слов об этой платформе, почему она такая хорошая, какие ей есть альтернативы (тут heavy wiki!)

### 4.2.2 Фреймворк Express.js

Тут надо обосновать необходимость использования серверного фреймворка, аргументируя решением таких задач, как маршрутизация и рендеринг представлений

### 4.2.3 HTML препроцессор Jade

Тут нужно обосноват использование html-препроцессора jade, рассказать про его достоинства и про то, почему его удобно использовать (видимо, вики)

### 4.2.4 Конвертация загруженных *.obj-объектов

Рассказать, как это делается, что происходит, какие врапперы для чего были написаны, как и где хранятся объекты

## 4.3 Серверные вычисления

### 4.3.1 Формулировка задачи

Тут надо еще раз кратко рассказать (еще раз - потому как один раз это уже должно было быть в ведении) про большой codebase на C++ и про необходимость как-то интегрировать этот код

### 4.3.2 Применение thrift-технологии

Рассказать про то, что есть один файл model.thrift, что он описывает типы данных, а так же один сервис. Можно даже скопировать частично этот файл сюда, по крайней мере код описания сервиса так точно. Что по этому файлу генерятся стабы для C++ и для js, а так же скелет для C++ сервера. Что этот скелет потом с помощью скрипта-генератора наполняется содержанием на основе содержимого папки algo, и с помощью готового makefile'а можно быстро собрать готовый thrift-сервер.

### 4.3.3 Проблема с десериализацией 8-байтного вещественного типа

Тут надо рассказать, что у node-thrift'а была проблема с десериализацией Double, о том что она была локализована, устранена, и соответствующий патч был послан в сообщество на рассмотрение

### 4.3.4 Организация C++ кода

Рассказать про то, что C++ код можно писать довольно-таки обособленно, рассказать про проблему с идентификацией алгоритмов и с решением, включающим в себя мета-информацию в комментариях к алгоритму, которая парсится специальным скриптом. Рассказать про специальные ruby-скрипты для пользователя и их опции

## 4.4 Деплоинг приложения

Проблема деплоинга приложения, использование сначала сбоственной машины, а потом облачного сервиса heroku.com и его стека приложений CEDAR для временного хостинга приложения

# 5 Результат

## 5.1 Общий результат

Сделано бла-бла-бла, можно сказать, проведен эксперимент по возможности использования WebGL и что его можно считать удачным.

## 5.2 Проведенные тесты

### 5.2.1 Тесты производительности Three.js

Надо рассказать про некоторое количество конфузов, которые возникли при работе с движком. Например, про то, что пересечение луча со сценой работает на уровне JavaScript'а и потому не очень скоростное, хотя и использует отсечения по описывающей сфере

### 5.2.2 Тесты производительности node-thrift

Рассказать про неожиданное падение в скорости при сериализации объектов в node-thrift, о некоторых замерах времени, а так же сказать, что это очень плохо и хотелось бы это пофиксить. Можно зафигачить какую-нибудь табличку со сравнительными данными.

## 5.3 Выводы