ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Кафедра информационных систем в искусстве и гуманитарных науках

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий Кафедрой информационных систем в искусстве и гуманитарных науках

_____ **(Борисов Н.В.)**
“ 23 **”** мая **2016 г.**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
Основная образовательная программа
«Прикладная информатика в области искусств и гуманитарных наук»
Направление 230700 «Прикладная информатика»
Уровень Бакалавриат

«Создание интерактивной трехмерной инсталляции в веб-среде»

Студента *Сколова Юрия Владимировича*

_____
(подпись студента)

Руководитель: *кандидат физ.- мат. наук, доцент СПбГУ,*
Захаркина Валентина Валентиновна

_____
(подпись руководителя)

Санкт-Петербург
2016

АННОТАЦИЯ

выпускной квалификационной работы Сколова Юрия Владимировича

«Создание интерактивной трехмерной инсталляции в веб-среде»

Ключевые слова: HTML5, WEBGL, 3D ГРАФИКА, ВЕБ-ПРИЛОЖЕНИЯ, ИССЛЕДОВАНИЕ ТЕХНОЛОГИЙ ОТОБРАЖЕНИЯ 3D ГРАФИКИ В БРАУЗЕРЕ БЕЗ ИСПОЛЬЗОВАНИЯ СТОРОННИХ ПЛАГИНОВ, СОЗДАНИЕ ТРЕХМЕРНЫХ ИНТЕРАКТИВНЫХ ВЕБ-РЕСУРСОВ

Целью данной выпускной квалификационной работы является исследование современных технологических решений для создания трехмерных интерактивных сцен в веб-среде и создание на основе полученных результатов двух демонстрационных проектов (инсталляций) для задач кафедры. Среди решенных в ходе работы задач можно выделить следующие:

- Проанализированы современные технологии для реализации трудоемких графических задач в веб-среде.
- Выработаны критерии для поиска необходимого для проектов фреймворка, осуществлено исследование наиболее современных решений, сделан выбор на основе сравнительной таблицы характеристик.
- Созданы веб-инсталляции "Трехмерная карта мира" для проекта гуманитарной направленности "Наглядная история" и "Корела Web" для проекта трехмерной виртуальной реконструкции крепости Корела.
- Проведена оптимизация графической структуры проектов для улучшения скорости их работы в веб-браузерах, осуществлено тестирование.

Разработка веб-ресурсов была выполнена с помощью технологий HTML5, WebGL и языка программирования JavaScript (ECMAScript 6).

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей пояснительной записке к дипломной работе применяются следующие термины с соответствующими определениями:

Фреймворк – программное обеспечение, облегчающее разработку и объединение компонентов большого программного проекта, обычно включает в себя несколько разных библиотек для одного класса задач.

Веб-среда – в контексте данной работы: современный веб-браузер, поддерживающий стандарты HTML5, без установленного дополнительного программного обеспечения с возможностью выхода в сеть Интернет.

Эксплоит-кит – компьютерная программа (вирус), использующий уязвимости в программном обеспечении для проведения атаки на вычислительную систему. Целью атаки может быть как захват контроля над системой (повышение привилегий), так и нарушение её функционирования.

Шейдер – компьютерная программа, предназначенная для исполнения процессором видеокарты, используется для определения параметров геометрических объектов или изображения, для изменения изображения (для создания эффектов сдвига, отражения, преломления, затемнения с учетом заданных параметров поглощения и рассеивания света и т. д.).

Текстура – изображение, накладываемое на поверхность полигональной трехмерной модели для придания ей цвета, окраски или иллюзии рельефа.

Материал – набор настроек, описывающий свойства поверхности объекта.

Префаб – особый тип ресурсов, позволяющий хранить весь объект со всеми компонентами и значениями свойств. Префаб выступает в роли шаблона для создания экземпляров хранимого объекта в сцене. Любые изменения в префабе немедленно отражаются и на всех его экземплярах в сцене.

Плагин – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения ее возможностей, часто написан сторонним разработчиком.

Композитинг – сборка финальной сцены из разрозненных ресурсов согласно определенным правилам или изначально выстроенному плану.

Модификатор – сущность в 3DS Max, применяемая к объектам и моделям для изменения их определенных свойств, чаще всего геометрии.

ВВЕДЕНИЕ

Актуальность исследования технологий для разработки трехмерных интерактивных ресурсов обусловлена тем фактом, что привычный нам Интернет уже давно перестал быть исключительно источником текстовой информации – все больше на страницах крупнейших сайтов появляются различные элементы мультимедиа: видео, анимация, музыка, в том числе и трехмерная графика, которая используется в веб-среде в таких задачах, как визуализация какой-либо информации (зачастую научной), обучающие и развлекательные браузерные игры, анимированные логотипы, сложные пространственные интерфейсы и многих других. К объектам исследования на нашей кафедре трехмерная графика относится непосредственно: спектр проанализированных технологий можно использовать в создании таких проектов, как демонстрация портфолио трехмерных моделей в режиме реального времени, исторические реконструкции архитектурных и прочих памятников, встраивание отдельных элементов трехмерности в дизайн веб сайтов и т. д. Но из-за недостатка документации и каких-либо проверенных и устовшихся решений в данной области, с самого начала создания таких проектов встает множество вопросов:

- Как грамотно и по возможности быстро экспортировать имеющиеся трехмерные модели и анимацию в веб-среду?
- Как перейти от манипулирования координатами отдельных вершин объектов к манипулированию целыми объектами?
- Как реализовать взаимодействие импортированных трехмерных объектов с DOM элементами существующего веб-ресурса?

- Каким образом возможно оптимизировать сложную, нагруженную графическую сцену для ее полноценной работы в веб-браузере?

Именно необходимость и важность найти обоснованные ответы на данные (и некоторые другие) вопросы и побудили начать исследование по указанной теме – потому как богатство открывающихся возможностей после преодоления препятствий из подобных вопросов поистине безгранично. С каждым днем веб-среда становится все более насыщенной в плане технологий, и трехмерная графика в этом плане – очередной шаг вперед, позволяющий пользователям просматривать приложения, использующие сложные графические вычисления без предварительных установок, настроек – как в начале 90-х годов XX века можно было читать обычные текстовые веб-странички, «блуждая» по гиперссылкам.

Целями выпускной квалификационной работы были выбраны два аспекта рассматриваемой проблемы: теоретический (исследование современных технологических решений для создания трехмерных интерактивных сцен в веб-среде) и практический (создание на основе полученных результатов двух демонстрационных проектов для задач кафедры).

По причине крайне скудного освещения данной темы в литературных источниках (как печатных, так и сетевых изданий), в работе будет использована информация, собранная с различных разрозненных официальных источников, среди которых – блоги компаний, учавствующих в разработке рассматриваемых технологий, новости с крупнейших и проверенных порталов, таблицы и графики специальных аналитических агенств, занимающихся вопросами IT и веб технологий. В результате проделанного исследования и завершения практической части работы ожидается получить два полноценных проекта, а также ряд решений различных технологических проблем, с нуля разработанных на основе детального анализа поведения применяемых программных продуктов.

АНАЛИЗ ТЕКУЩЕГО СОСТОЯНИЯ ПРОБЛЕМЫ

Необходимость использования сложных графических вычислений в веб-среде стояла перед разработчиками чуть ли не со середины 90-х годов прошлого столетия – именно тогда начал набирать популярность созданный компанией Macromedia мультимедиа плагин Adobe Flash Player, позволяющий отображать в браузере достаточно ресурсоемкие графические приложения с анимацией, интерактивностью и различными визуальными эффектами. Построенный на основе технологии NPAPI (Netscape Plugin Application Programming Interface) Flash Player являлся посредником между браузером пользователя и исполняемым приложением, которое работало не на основе родной для веб-платформы связке HTML и JavaScript, а при помощи некой виртуальной машины - ActionScript Virtual Machine, которая и поставлялась вместе с плагином. К 2010 году по статистике специализированных компаний [1] около 50% сайтов в Интернете осуществляют хотя бы один Flash-запрос, а это получается, что почти каждый второй веб-ресурс (из числа 10000 самых посещаемых по версии Alexa Top) использует данную технологию. На основе данной платформы было создано немало графического контента, в том числе и среди проектов нашей кафедры: виртуальные музеи, двухмерная анимация, рисованные интерфейсы различной степени сложности, браузерные игры – одним словом практически все, что касалось отображения мультимедиа в Сети. В середине 2000-х были разработаны различные конкуренты Flash Player, такие как Silverlight от компании Microsoft и JavaFX от компании Oracle. На рубеже тех же времен начали появляться и развиваться первые решения для использования полноценной трехмерной графики в браузере – например, Unity Web Player (работающий по тому же принципу виртуальной машины).

Но критические недостатки как самой архитектуры NPAPI, на которой и базировались все плагины, так и самой идеи закрытой виртуальной машины (другими словами посредника) внутри браузера, породило со временем немало проблем – что в итоге побудило главу компании Apple опубликовать открытое письмо с призывом к мировому сообществу разработчиков отказаться от использования технологии Adobe Flash в Интернете [2].

И причин для такого шага было множество. Ниже будут перечислены лишь основные недостатки данной платформы:

- Высокая уязвимость. Согласно отчету о киберугрозах аналитической компании NTT Group [3], плагины Adobe Flash и Java входят в пятерку самых используемых хакерами технологий для создание эксплоит-китов. Ниже представлено краткое резюме по наиболее популярным уязвимостям из того же исследования (см. рис. 1). В истории отмечены случаи, когда «при помощи» Adobe Flash хакерам удалось взламывать компьютеры [4], операционные системы и получать удаленный доступ к оборудованию пользователей (такому как веб-камеры и микрофоны).

CVE	Percentage of kits which exploit this vulnerability	Affected Technology
CVE-2013-2551	62%	Microsoft Internet Explorer
CVE-2014-0515	52%	Adobe Flash
CVE-2013-0074	38%	Microsoft Silverlight
CVE-2013-2465	38%	Oracle Java SE
CVE-2013-0634	29%	Adobe Flash
CVE-2013-2460	29%	Oracle Java SE
CVE-2014-0322	29%	Microsoft Internet Explorer
CVE-2014-0497	29%	Adobe Flash
CVE-2014-0569	29%	Adobe Flash
CVE-2012-0507	24%	Oracle Java SE
CVE-2012-1723	24%	Oracle Java SE

Рисунок 1. Резюме по наиболее популярным уязвимостям среди хакеров.

- Закрытость. Технология Adobe Flash – собственность компании Adobe, которая обладает исключительным правом на исправление ошибок и уязвимостей в своем продукте. Как результат, это порождает весьма неторопливое их исправление, так как тот же плагин является совершенно бесплатным, и компания не сильно заинтересована в том, чтобы уделять его развитию большое количество своих ресурсов.
- Недостаточная оптимизация. Согласно открытому письму [2] Стива Джобса, батарея мобильных устройств Apple при просмотре видео с помощью технологии Adobe Flash разряжается в два раза быстрее, чем при помощи использования современного видео стандарта H.264. Немало статей в Интернете было также посвящено и ограниченности технологии при работе с большим количеством векторной графики.
- Трудности в установке. Наслышанные о крайней уязвимости технологии Flash, многие системные администраторы отключают данный плагин в корпоративной среде для повышения безопасности. Более того, несмотря на то, что сам плагин зачастую поставляется вместе со всеми популярными браузерами, его обновление требует наличия обязательных администраторских прав, что за пределами дома также может оказаться препятствием к его использованию.

Итак, все перечисленные недостатки (и часть чуть менее значительных) привели к тому, что в 2015 году компания Adobe сама признала победу открытого стандарта HTML5 над Flash Player и призвала пользователей как можно скорее отказаться от его использования. Параллельно с этой новостью, большинство популярных браузеров, среди которых можно точно назвать Mozilla Firefox и Google Chrome [5] отключили поддержку Adobe Flash по умолчанию, а также и ряд других NPAPI плагинов (см. рис. 2), страдающих практически теми же проблемами, что и продукт от Adobe.

В результате, появилась необходимость разработки принципиально нового решения для поддержки сложных графических вычислений в браузере. И если для двухмерной графики таких подходов на данный момент имеется несколько (CSS3 анимация, DOM-элемент Canvas), то для 3D графики был создан единый открытый стандарт под названием WebGL. И на данный момент этот стандарт является *единственным* возможным и рекомендуемым для отображения полноценной трехмерной графики в браузере.

КРАТКИЙ ОБЗОР ТЕХНОЛОГИИ WEBGL, ЕЕ ОСНОВНЫЕ ПРЕИМУЩЕСТВА И НЕДОСТАТКИ

Описание технологии WebGL будет приведено в трех пунктах.

1. Что такое WebGL? Зачем он нужен? Для чего используется?

WebGL (Web-based Graphics Library) — программная библиотека для языка программирования JavaScript, позволяющая создавать интерактивную трехмерную графику, функционирующую в широком спектре совместимых с ней веб-браузеров. В ее основе лежит технология OpenGL ES (Open Graphics Library for Embedded Systems 2.0), отвечающая за ускорение и вывод трехмерной графики на мобильных платформах, которая уже доказала свою состоятельность и перспективность. Первая версия стандарта данной технологии была выпущена 3 марта 2011 года. В рабочую группу WebGL помимо компании Khronos Group (собственно создателей этой библиотеки, а также схожих по функциональным возможностям библиотек OpenGL для настольных платформ и OpenGL ES для мобильных устройств) входят представители ведущих разработчиков веб-браузеров, таких как Apple Safari, Google Chrome, Mozilla Firefox и Opera, а также специалисты AMD и Nvidia. На момент написания работы ведется активная работа над версией 2.0, черновой вариант стандарта можно посмотреть на сайте [6].

Ключевая особенность проекта – обеспечение необходимых функций по работе с трехмерной графикой в окне браузера на базе обычного JavaScript кода, *без использования* посторонних плагинов и виртуальных машин, которые и послужили причиной большинства нареканий в том же Adobe Flash Player и других аналогичных ему продуктах.

Сейчас WebGL можно считать уже достаточно состоявшейся библиотекой, которая не без успеха используется в ряде крупных проектов, среди которых:

симуляция марсохода Curiosity [7] на поверхности Марса (инициатором создания приложения стала компания NASA), трехмерная инфографика о загрязнении окружающей планеты (созданная российским подразделением компании GreenPeace), внедрение различных экспериментальных функций в таких крупнейших ГИС как Яндекс.Карты, 2ГИС, множество игровых и обучающих приложений в социальных сетях Vkontakte, Facebook и т. д.

2. В чем же основные преимущества библиотеки WebGL?

Разумеется, такое широкое распространение и уважение среди лидеров IT-индустрии (Apple, Google, Mozilla, Microsoft и других) данная технология получила не просто так. Основная причина заключается в том, что WebGL является действительно принципиальным переосмыслением подхода по созданию и обработке трехмерного графического контента в веб-среде (см. рис. 2). Таким образом, WebGL не только избавился от ряда критических недостатков своих предшественников (в лице Flash Player, Silverlight и Unity Web Player), но и получил ряд важнейших преимуществ, в результате отказа от устаревшей и полной уязвимостей архитектуры NPAPI и перехода к полностью открытому стандарту. Рассмотрим достоинства чуть подробнее:

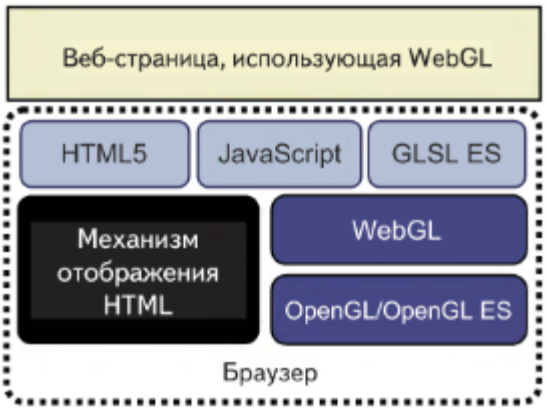

Рисунок 2. Архитектура веб-приложений, использующих WebGL.

- Полная открытость (open source). Стандарт WebGL открыт на суд мировому сообществу разработчиков, где его недостатки не остаются без

внимания – каждый день в исходный код вносятся новые изменения, правки, модификации. Это позволяет разработчикам оперативно устранять все погрешности, а также оставаться более гибкими к внедрению новых возможностей в последующих версиях.

- Безопасность. WebGL работает на основе тех же технологий, что и самые обычные веб-страницы и веб-приложения, без использования посторонних плагинов. Более того, WebGL – это 100% JavaScript код, который каждый при желании может просмотреть и убедиться в отсутствии каких-либо посторонних подозрительных вызовов [8].
- Простота установки. Технология WebGL уже поддерживается большинством современных браузеров «из коробки», нет необходимости что-либо устанавливать или обновлять вручную. Более того, можно сказать, что конечный пользователь может вообще не задумываться или не подозревать о существовании технологии.
- Широкая распространенность. По данным наиболее свежей на момент написания работы статистики специализированной организации – около 90% веб-браузеров пользователей [9] уже поддерживают WebGL (из приблизительно 15 миллионов «опрошенных» по всему миру), в том числе и на всевозможных мобильных устройствах (см. рис. 3).

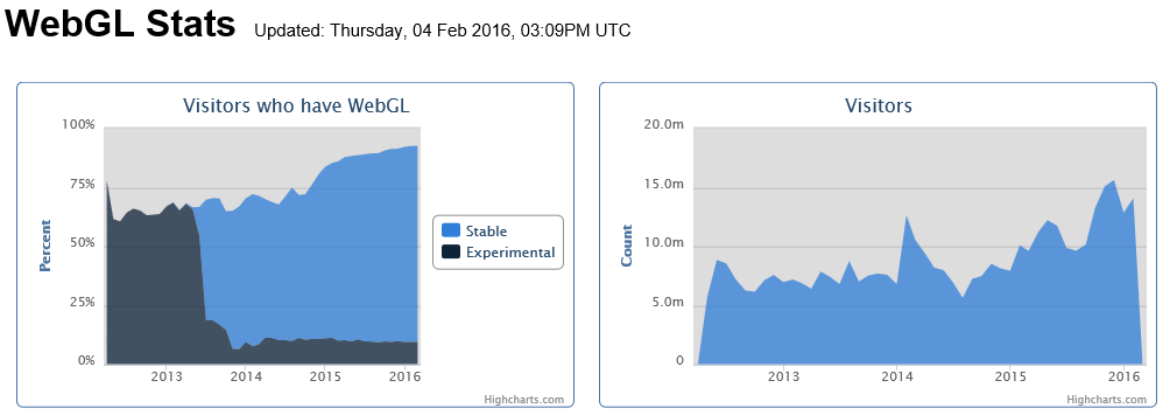

Рисунок 3. Статистика поддерживаемости технологии WebGL на разных платформах.

- Бесплатность. Использование WebGL бесплатно не только в чисто персональных, но и коммерческих целях – что почти сразу же породило множество дополнительных надстроек и фреймворков для упрощения реализации различного функционала, будь то создание ландшафтных панорам, обработка двухмерных изображений, работа с трехмерной графикой, поддержка трехмерной анимации и т. д. В случае с Flash Player и Unity Web Player за создание и поддержку новых функций отвечали в основном только сами издатели продуктов.
- Мультиплатформенность. В отличие от технологии Flash Player, которая была не способна поддерживать touch-интерфейсы, WebGL уверенно работает как на настольных компьютерах под операционными системами Windows, Mac OS, Linux, так и на мобильных устройствах – в веб-браузерах Safari, Opera Mini, Chrome.
- Hardware ускорение. Библиотека WebGL рационально распределяет используемые ресурсы между центральным процессором и видеокартой, что положительно сказывается на времени работы устройств от батареи, так и на скорости работы веб-браузеров в целом [8].
- Полноценная трехмерность. WebGL позволяет создавать приложения, использующие трехмерную графику, любого класса – от небольших рекламных баннеров до визуализации сложных химических процессов и редакторов по созданию собственных трехмерных моделей.

Данная информация была получена как из числа указанных источников, так и из личного опыта использования всех вышеперечисленных технологий (Adobe Flash Player, Unity Web Player, WebGL).

3. В чем заключаются главные недостатки WebGL?

Несмотря на все вышеперечисленные достоинства, WebGL – довольно молодая библиотека, не имеющая коммерческого начала, поэтому помимо всего прочего обладает рядом значительных недостатков. Среди них:

- Необходимость низкоуровневого программирования. WebGL – низкоуровневая библиотека для работы с графикой, где существует необходимость оперировать такими объектами, как вершины, самостоятельно настраивать множество различных параметров отображения, и даже простая программа по выводу квадрата на «полотно» графического контекста может занимать до 200 строк кода. Для более сложных программ уже желательно понимание линейной алгебры, что безусловно усложняет разработку приложений «с нуля».
- Малое количество документации. Несмотря на достаточно большое сообщество разработчиков, поддерживающих WebGL, технология на данном этапе очень сильно страдает от недостатка документации по многим непростым, но широко распространенным моментам. В итоге складывается ситуация, когда при создании каждого нового приложения программист вынужден заново «изобретать велосипед», от чего разработка может растянуться в 2 или 3 раза по сравнению с разработкой на хорошо изученных технологиях OpenGL и OpenGL ES.
- Низкая производительность. Не секрет, что язык JavaScript является намного более медленным решением по сравнению с языком C++ (более подробные тесты сравнения производительности – [10]), а обработка трехмерной графики *в реальном времени* сама по себе задача достаточно ресурсоемкая. Из всего этого вытекает, что приложения на WebGL никогда не смогут сравниться по производительности с аналогичными настольными приложениями. А значит, для вывода в окошко браузера какой-либо более-менее

сложной трехмерной сцены требуется тщательное тестирование, а также разработка и выполнение определенных требований для оптимизации работы финальной сцены.

Но потребность в работе с трехмерной графикой в веб-среде имеется уже сейчас, а как уже было указано выше, WebGL на данный момент является *единственным* возможным и поддерживаемым стандартом для реализации задач подобного круга. Следовательно, по причине отсутствия готовых решений по указанным выше проблемам, было решено самостоятельно разработать новые технологические подходы по их разрешению, а из списка данных «проблем», которые встречаются при разработке любого приложения, основанного на базе библиотеки WebGL, и сформировать основные цели и задачи для данной выпускной квалификационной работы.

ЦЕЛИ И ЗАДАЧИ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ, ОПИСАНИЕ ОСНОВНЫХ ПОДХОДОВ К ИХ РЕШЕНИЮ

Тезисно цели и задачи данной выпускной квалификационной работы были описаны в разделах «Аннотация» и «Введение» текущей пояснительной записки, здесь же рассмотрим их чуть подробнее.

Итак, в результате формирования проблемы было решено разделить цели работы на два аспекта: теоретический и практический.

- Теоретический аспект. Цель: проведение исследования современных технологических решений для создания трехмерных интерактивных сцен в веб-среде. Круг решаемых задач:

  1) Анализ современных технологических решений для реализации трудоемких графических задач в веб-среде.

  2) Поиск подходящего для задач проекта фреймворка на основе сравнения имеющихся на рынке решений по заранее выработанным критериям, составление сравнительной таблицы на основе результатов.

- Практический аспект. Цель: Создание двух типовых демонстрационных проектов (инсталляций) для задач кафедры на основе полученных результатов исследования. Круг решаемых задач:

  1) Создание веб-инсталляции "Трехмерная карта мира" для проекта гуманитарной направленности "Наглядная история".

  2) Создание веб-инсталляции "Корела Web" для проекта виртуальной трехмерной реконструкции крепости Корела.

  3) Разработка критериев для оптимизации графической структуры проектов, применение полученных критериев к проектам для оптимизации скорости их работы в веб-среде.

4) Тестирование реализованных проектов в различных современных браузерах на предмет корректности выполнения.

В самой первой из решаемых задач («анализ современных технологических решений для реализации трудоемких графических задач в веб-среде») будут рассмотрены современные программные продукты, связанные с возможностью работы с 3D графикой в пространстве веб-браузера, проведено сравнение как между выбранными технологиями, так и с устовшимися индустриальными стандартами, в итоге чего будет сделан обоснованный выбор определенного фреймворка для дальнейшей работы.

В практической части работы также будут рассмотрены методы решения обозначенных выше недостатков выбранной библиотеки (WebGL).

Для полного ухода от *необходимости низкоуровневого программирования* и последующего перехода на новый уровень абстракции (от отдельных элементов трехмерных объектов – вершин, полигонов – к полноценным объектам) будет выбран фреймворк – набор библиотек и программных средств, облегчяющих разработку сложного программного продукта. Таким образом, вывод в окно браузера такого примитива как куб будет занимать уже не 200 строчек кода, а всего одну. Соответственно, более сложные задачи, как например, работа с физикой трехмерных объектов и создание камер, будут отнимать гораздо меньше времени на их решение (см. рис. 4).

По причине *малого количества документации* будет разработан ряд технологических решений вопросов, вставших в ходе реализации проектов, на часть из которых не было найдено ответа в сети Интернет, а значит, была необходимость при каждом таком случае проводить подробный анализ поведения веб-ресурса и экспериментировать с полученным результатом. Все подобные случаи, как в некоторой степени инновационные, будут указаны в разделе «Заключение» в рамках перечисления результатов работы.

С другой стороны, так как оба проекта-инсталляции будут разработаны, как говорится, «с нуля», часть найденных решений общего характера могут быть впоследствие использованы при создании любого аналогичного проекта, что сможет частично компенсировать недостаток документации по технологии.

В конце реализации каждого из проектов будет проведено тестирование в различных веб-браузерах и на его основании вынесено решение о необходимости оптимизации графической структуры проекта. После чего, на основе выработанных критериев (как на основе рекомендаций из официального стандарта [6], так и на основе практических экспериментов) будет проведена сама оптимизация – различными способами, которые также могут быть применены в дальнейшем при создании проектов на основе выбранной в данной работе технологии. Главной задачей проведения оптимизации является сохранение визуальной привлекательности проектов при более стабильной и быстрой работе (при необходимости).

```
// FreeCamera >> You can move around the scene with mouse and cursor keys
// Parameters : name, position, scene
    var camera = new BABYLON.FreeCamera("FreeCamera", new BABYLON.Vector3(0, 1, -15), scene);
```

Рисунок 4. Пример создания объекта «камера» во фреймворке Babylon.js в одну строку.

АКТУАЛЬНОСТЬ И ПРИКЛАДНАЯ ЗНАЧИМОСТЬ РЕШАЕМЫХ В РАБОТЕ ЗАДАЧ

Актуальность всей выпускной квалификационной работы в целом можно оценить по двум независимым друг от друга параметрам: актуальность самого объекта исследования (в нашем случае – технологии WebGL) и актуальность конкретного исследуемого аспекта (в нашем случае – анализа харакетристик программных продуктов, упрощающих разработку с использованием этой бибилиотеки). Как было упомянуто выше, с течением времени WebGL лишь набирает популярность, а заинтересованность данной технологией становится все выше как среди профессиональных веб-разработчиков, так и среди обычных энтузиастов (любителей программирования и компьютерной графики). Об этом говорит множество фактов, среди которых можно отметить следующие:

- Компания Google выделила специальный домен первого уровня для сайта, содержащего ссылки и описание экспериментальных веб-проектов, большинство из которых реализовано при помощи WebGL [11].
- На популярном среди программистов ресурсе (хранилище репозиториев исходного кода) GitHub количество «звездочек» (количество пользователей, подписавшихся на проект и следящих за его изменениями и модификациями) у наиболее популярного WebGL фреймворка под названием Three.js на момент написания работы – около 25170. Для сравнения аналогичный показатель у языка программирования Ruby – 9716, а у одного из популярнейших фреймворков CakePHP (для языка программирования PHP) – 6474. Можно сказать, что данный показатель напрмую отражает заинтересованность разаботчиков и пользователей к той или иной технологии, а также степень ее текущей актуальности.

В то же время другие WebGL фреймворки (помимо Three.js), несмотря на то, что по своим свойствам и производительности не уступают данному, а как будет видно ниже из сравнения, чаще всего – превосходят, совершенно не освещены в Интернете, печатной литературе (достаточно попробовать найти по ним книгу на ресурсах, вроде Amazon или Ozon для русскоязычной аудитории), когда как по Three.js существует немало книжных изданий (как минимум 3 – написанные разными авторами в разное время) и советов в сети Интернет по рациональному использованию этого программного продукта. Дело здесь в том, что первая версия Three.js вышла еще в 2010 году (во время нестабильной ветки WebGL) и почти за 6 лет [12] успела обрасти большим сообществом вокруг себя и накопленным количеством рекомендаций (с другой стороны, большая их часть в современной версии не действительна). Остальные фреймворки начали появляться и развиваться уже после роста популярности самого WebGL, соответственно, вопросы, когда-то решенные для three.js, остаются все еще открытыми для них. И ввиду их сильного технологического превосходства над Three.js, решение этих вопросов для получения возможности преимуществами воспользоваться в дальнейшем при разработке проектов имеет непосредственную прикладную значимость. Стоит добавить, что по результатам исследования, полученный стек технологий можно будет использовать для задач различной направленности без особых изменений – начиная от создания интерактивной инфографики или трехмерного логотипа для сайта, заканчивая полноценными виртуальными реконструкциями памятников культурного наследия.

Актуальность и прикладная значимость самих проектов раскрывается как в демонстрации использования исследованных технологий, так и в их самостоятельном значении для определенных проектов. В данном случае, веб-инсталляция «Трехмерная карта мира» представляет собой трехмерный интерфейс, призванный заменить (или дополнить) собой существующую двухмерную карту проекта, по ряду причин уступающую ее трехмерному

аналогу, как то: нагроможденность иконок, невозможность центрировать карту по своему усмотрению (например, расположить в центре Африку), достаточно медленная работа при большом количестве используемой графике или, как следствие, необходимость строгого ограничения этого самого количества, что негативно сказывается на визуальном восприятии.

Трехмерный вариант благодаря технологии WebGL, обладающей возможностью аппаратного ускорения графики, призван решить вопросы с производительностью, а также благодаря использованию полноценного трехмерного пространства, решить проблему удобства использования – карту можно будет крутить и располагать в любом желаемом положении.

Веб-инсталляция «Корела Web» представляет собой некоторый реинжиниринг и переосмысление структуры версии для настольных систем, направленный на запуск и полноценную работу приложения в веб-среде. Данное приложение является ярким примером работы с достаточно нагруженными в графическом плане проектами, поэтому дополнительно проведенная работа над изучением возможностей оптимизации может быть использована в игровых и обучающих приложениях с детализированными трехмерными элементами. Таким образом, после смены платформы, проект реконструкции крепости Корела станет доступен без дополнительных установок или компиляции для пользователей операционных систем Windows, Mac OS и семейства Linux прямо через браузер, например, при переходе с заглавного сайта проекта. Дополнительными преимуществами сетевой версии станут более низкие по сравнению с настольной версией системные требования, отсутствие необходимости вручную выделять дополнительное место на жестком диске компьютера и иметь привилегии Администратора в политиках групповой безопасности для запуска проекта.

РАЗРАБОТКА КРИТЕРИЕВ ДЛЯ ПОИСКА ФРЕЙМВОРКА, ИССЛЕДОВАНИЕ СУЩЕСТВУЮЩИХ НА РЫНКЕ РЕШЕНИЙ НА ПРЕДМЕТ УДОВЛЕТВОРЕНИЯ ПОСТАВЛЕННЫМ КРИТЕРИЯМ

Посте того, как цели и задачи были окончательно сформулированы, их актуальность и прикладная значимость были обоснованы, необходимо решить на какой технологической базе будет строится прикладная часть.

Итак, в качестве наиболее удачного варианта низкоуровневой библиотеки была выбрана технология WebGL, о причинах выбора и об основных преимуществах которой подробно рассказано в разделе «Краткий обзор технологии WebGL, ее основные преимущества и недостастатки» данной пояснительной записки. Там же был указан один из основных недостатков такого решения – необходимость писать сложный, низкоуровневый программный код. В разделе «Цели и задачи выпускной квалификационной работы, описание основных подходов к их решению» отмечено, что этот недостаток будет решаться поиском фреймворка – специального набора блиблиотек и программных модулей, облегчающих разработку сложных программных продуктов. Если говорить простыми словами, то фреймворк автоматизирует большинство низкоуровневых задач, и при обращении к API того или иного фреймворка, мы на самом деле обращаемся к функциям и методам библиотеки WebGL, только опосредованно. В качестве аналогии здесь может служить то, что при нажатии кнопки «Сохранить как...» в редакторе Microsoft Word, он автоматически вызывает диалог Проводника с предложением выбора нужного места на жестком диске, а уже после программа обращается к программным библиотекам ОС Windows для работы с интерфейсом и драйвером жесткого диска, притом пользователю каждый раз необходимости запускать весь список вызовов к драйверам нет

– он просто нажимает кнопку «Сохранить как...», как бы поднимаясь по уровням абстракции на несколько ступеней вверх.

Так и наличие фреймворка, несмотря на оставшуюся необходимость программировать, позволяет не задумываться о том, из каких вершин и в каких коорината состоит тот или иной объект, а манипулировать самим объектом, как самостоятельной структурой, таким образом, сосредоточив больше внимания и времени на логике и функционале приложения.

Данная практика является распространенной, и можно упомянуть, что в настоящее время все более-менее сложные веб-ресурсы написаны с использованием одного или нескольких фреймворков для языков PHP, JavaScript, Python и других (в зависимости от нужд самих проектов).

Открытым вопросом остается выбор необходимого и наиболее подходящего под разрабатывамые в ходе данной работы проекты продукта – так как ввиду актуальности и популярности технологии WebGL, количество предложений стремится к 40 позициям разной степени качества (согласно информации, взятой на официальном сайте Khronos Group [13], разработчиков WebGL). Соответственно, в начале опишем примерный алгоритм поиска:

1. Разрабатываем список критериев, по которым будет определяться степень соответствия того или иного продукта нашим требованиям.

2. Составляем полный список предлагаемых на рынке фреймворков.

3. На первом этапе отбора пользуемся наиболее критичными для разработки критерями из массива выделенных ранее.

4. На втором этапе отбора анализируем оставшиеся продукты по второй части выработанных ранее критериев.

5. Составляем сравнительную таблицу характеристик, из которой делаем окончательное решение и заключительный вывод исследования.

Среди наиболее критичных для дальнейшей разработки критериев для искомого технологического решения были выделены следующие:

- JavaScript-ориентированность. Среди модулей, работающих на JavaScript изначально, имеются также и те, которые способны компилировать свой код в JavaScript из других языков, в частности из Haxe (BabylonHx), Java (Parallax), а также C# (Unity 5 Export) и C++ (Unreal Engine Export). Такой подход имеет ряд существенных недостатков, как то: «загрязненный код» в результате его автоматической генерации и, как следствие, большой объем файлов и достаточно долгая загрузка (см. раздел «Проектная часть выпускной квалификационной работы. Разработка веб-инсталляции «Корела Веб», где сделан и описан более подробный анализ и сравнение результатов экспорта Unity 5 с выбранным фреймворком). Также обычно при первоначальной работе на другом языке программирования не всегда есть возможность обратиться напрямую к DOM элементам веб-ресурса, что оказалось критичным для сцены «Трехмерная карта мира».
- Бесплатность. Выбранный фреймворк должен предоставлять полный функционал на бесплатной основе для некоммерческого использования.
- Актуальность (обновляемость). Крайне желательно, чтобы разработчики активно занимались своим продуктом и выпускали обновления.
- Инструментарий для экспорта объектов из редакторов трехмерного моделирования. Так как большая часть трехмерных элементов проектов будет создаваться в специализированном редакторе, фреймворк должен иметь возможность работы с каким-либо из распространенных форматов файлов для трехмерной графики и уметь правильно его обрабатывать.
- Наличие базовой документации. На сайте есть подробное описание начала работы с продуктом и разбор его основного функционала.

Вышеперечисленные критерии были проверены на списке фреймворков с официального сайта Khronos Group [13], после чего были сделаны выводы, в результате которых получается, что большинство из перечисленных продуктов находятся или в состоянии технологического несовершенства (как вариант, просто в ранней стадии разработки), или не обновляются по различным причинам сроком от года и более (что равносильно тому, что они перестали отвечать современным стандартам постоянно развивающегося WebGL). Например, были отброшены следующие варианты: Famo.us, Scene.js, PhiloGL, SpiderGL и другие, с аналогичными недостатками.

Итоговый список фреймворков, допущенных по критериям ко второму этапу отбора выглядел таким образом:

- Goo Create
- Three.js
- Blend4Web
- Turbulenz
- Babylon.js
- PlayCanvas

Разработанный на следующем этапе список критериев относится уже непосредственно к процессу разработки и его удобству, эргономичности. В данном контексте имеет смысл задать такие вопросы, как: С помощью какого инструмента разработка будет выполнена наиболее продуктивно? Какой из фреймворков позволяет осуществить экспорт проекта в веб-среду наиболее комфортно? Какой из этих продуктов обеспечивает оптимальное соотношение предоставляемых возможностей и итоговой скорости работы?

Список составленных для второго этапа отбора критериев выглядит так:

- Простота установки фреймворка, возможность его интеграции в уже существующий веб-ресурс (обозначение в таблице - I).
- Скорость реализации тестового мини-примера, прогулки по комнате с камерой «от первого лица» (обозначение в таблице – II).
- Качество, доступность и широта охвата (информативность) официальной документации фреймворка (обозначение в таблице – III).

Рассмотрим результаты, полученные по каждому критерию чуть подробнее. В плане установки наиболее сложным из всех оказался фреймворк Turbulenz, предлагающий компилировать свой исходный код «с нуля» после установки дополнительных модулей, а также интерпретатора Python только определенной версии (ветка 2.7). Часть продуктов (Goo Create и PlayCanvas) работают при помощи облачных технологий – удаленных серверов, которые предоставляют пользователю возможность взаимодействия с функционалом данных фреймворков, как с обычным сайтом, посредством веб-интерфейса.

В качестве тестового мини-примера была выбрана прогулка по комнате с камерой «от первого лица» с небольшим элементом интерактивности: при нажатии на барабанную установку (см. рис. 5), цвет двух противоположных стен комнаты изменяет свой цвет (с белого на бирюзовый).

Модели были выполнены в редакторе 3Ds Max, после чего экспортированы встроенными утилитами в поддерживаемый тем или иным фреймворком формат, для Babylon.js – в .babylon, для Turbulenz – в .json и т. д.

Стоит отметить, что сложность API и документации Turbulenz не позволила реализовать задание на должном уровне при приемлемом объеме времени. С другой стороны, использование самой распространенной на данный момент WebGL-библиотеки под названием Three.js дало, пожалуй, наименее

выразительные результаты в плане скорости разработки и полученного в итоге объема кода (суммарный объем JavaScript составил более 300 строк, когда как в других вариантах, при использовании аналогов, удавалось достичь объема около 50 строк и менее). При использовании фреймворка Blend4Web все модели и текстуры пришлось сначала переносить в редактор трехмерного моделирования Blender, а уже после – экспортировать на веб-сайт, так как данный продукт позиционирует себя как некое логическое продолжение интерфейса упомянутого редактора Blender.

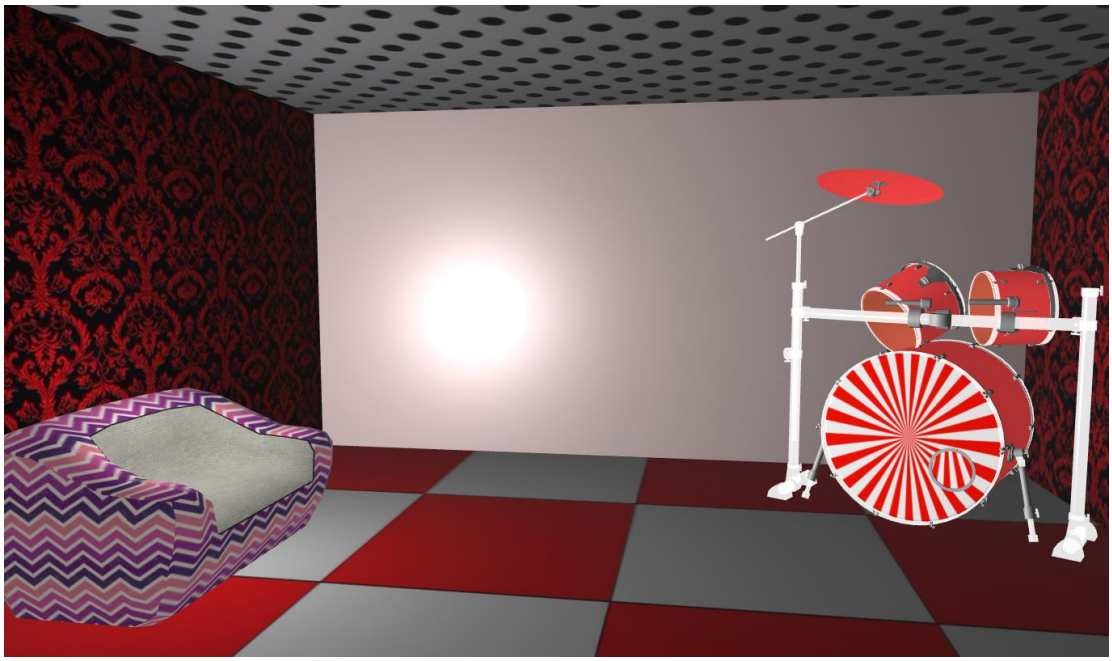

Рисунок 5. Выполненный тестовый мини-пример прогулки по комнате.

В плане широты охвата и доступности документации было вызвано наибольшее количество нареканий, так как зачастую описывались только самовые базовые вещи, вроде установки фреймворка, экспорта моделей, создание и настройка камер и освещения и т. д. Тем не менее, документация фреймворков Goo Create, Babylon.js и Blend4Web отличается наибольшей логичностью и простотой изложения, а также достаточным объемом тем, раскрываемых в ней, для уверенной реализации тестового мини-примера.

В итоге было решено составить сводную таблицу по заявленным выше критериям для выбранной группы фреймворков, где галочка в ячейке обозначает отсутствие нареканий по тому или иному пункту (см. рис. 6).

N/N	I	II	III
Goo Create	✓	✓	✓
Three.js	✓		
Blend4Web	✓		✓
Turbulenz			
Babylon.js	✓	✓	✓
PlayCanvas	✓	✓	

Рисунок 6. Сравнительная таблица характеристик фреймворков.

В результате проведенного исследования доступных на рынке технологических решений для работы с библиотекой WebGL были разработаны специальные критерии, согласно которым наиболее подходящими по всем описанным харакетристикам фреймворками являются Goo Create и Babylon.js. Как следствие того, что первый продукт работает «из облака» и имеет вспомогательные средства для разработки логики приложений, в качестве основного инструмента был выбран именно он.

Но, в ряде случаев, когда необходима возможность работы *без Интернета*, следует обратить внимание на фреймворк Babylon.js (создателями которого, к слову, является французское подразделение корпорации Microsoft, с его же помощью реализовавшая часть функционала игровой приставки Xbox One).

ПРОЕКТНАЯ ЧАСТЬ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ. РАЗРАБОТКА ВЕБ-ИНСТАЛЛЯЦИИ «ТРЕХМЕРНАЯ КАРТА МИРА»

Веб-инсталляция «Трехмерная карта мира» представляет собой трехмерный интерактивный интерфейс в виде глобуса, на котором расположены зоны взаимодействия с основным контентом ресурса «Наглядная история», который разрабатывается в рамках дипломных проектов на нашей кафедре.

Основными преимуществами трехмерного представления и использования технологии WebGL в разработке будут:

- Более быстрая работа с графикой благодаря возможности аппаратного ускорения ядром графического процессора.
- Более эргономичная навигация по сравнению с двухмерным аналогом, не имеющим возможности передвигать карту.
- Решение вопроса с нагромождением графических элементов и анимационных эффектов поверх зон взаимодействия.
- Новизна и оригинальность визуального представления карты (так как все похожие проекты используют именно двухмерный вариант).

Теперь чуть подробнее стоит обрисовать функционал, предоставляемый разрабатываемым интерфейсом. Основным способом интеракции пользователя с информационным контентом ресурса гуманитарной направленности «Наглядная история», представленным в нем в виде так называемой тайм-линии (временной шкалы), на которой расположены события того или иного периода в том или ином регионе (см. рис. 7), является как раз карта мира. На ней пользователь может выбрать интересующий его регион, после чего выбрать необходимую дату (которые

в свою очередь для удобства разбиты на декады), после чего автоматически система переключается на выбранную декаду на тайм-линии.

Рисунок 7. Пример тайм-линии с указанными на ней историческими событиями.

Основную сложность в данном случае представляют два момента:

- Взаимодействие с существующими DOM элементами сайта, генерация новых узлов и использование уже существующих.
- Интеграция трехмерного интерфейса в существующий двухмерный сайт.

Именно эти проблемы и будут рассмотрены далее чуть подробнее, а также некоторые другие особенности и решения вопросов, возникших в процессе реализации задуманного функционала на WebGL фреймворке Goo Create.

1. Создание трехмерных моделей, настройка базовой сцены (с камерой и освещением) во фреймворке Goo Create.

Моделирование трехмерной карты (далее – глобус) осуществлялось в бесплатной версии программы Cinema4D Lite. Особенность модели состоит в том, что каждый из регионов необходимо было выделить в отдельный трехмерный объект и продублировать его расположение на поверхности самого глобуса (грубо говоря, каждый из таких регионов будет являться на сайте кнопкой или триггером для определения характера взаимодействия).

После создания объекта и подбора необходимой текстуры (см. рис. 8), следующим шагом он был экспортирован в распространенный формат обмена трехмерными моделями .fbx, разработанный компанией Autodesk. Так как модель глобуса не содержит данных об анимации, настройки экспорта для облегчения размера итогового файла выставлены следующим образом (см. рис. 9). С другой стороны, текстуры, использованные для данной модели необходимо включить в тот же файл, где находится и ее геометрия, для автоматической подгрузки и установки текстурных координат на веб-сайте *в том же положении*, что и в программе Cinema4D.

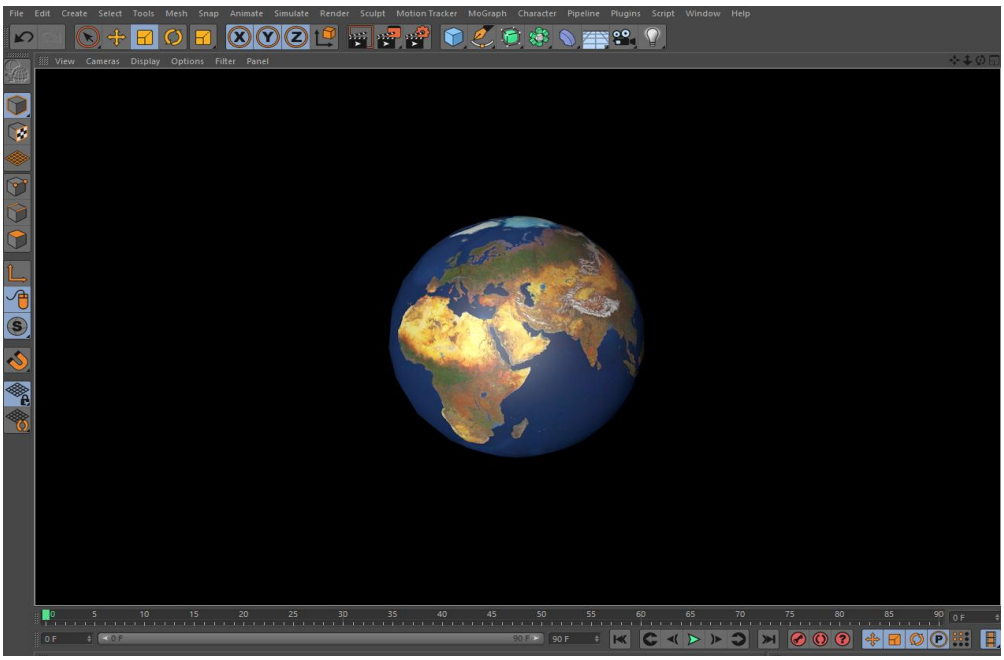

Рисунок 8. Прототип модели «глобуса» в интерфейсе редактора Cinema4D Lite.

Уже полученная трехмерная модель глобуса импортировалась в редактор Goo Create. После чего было необходимо настроить сцену: установить на ней созданную нами модель, настроить материалы, установить камеру, задать ей соответствующие задачам проекта настройки, установить и настроить свет.

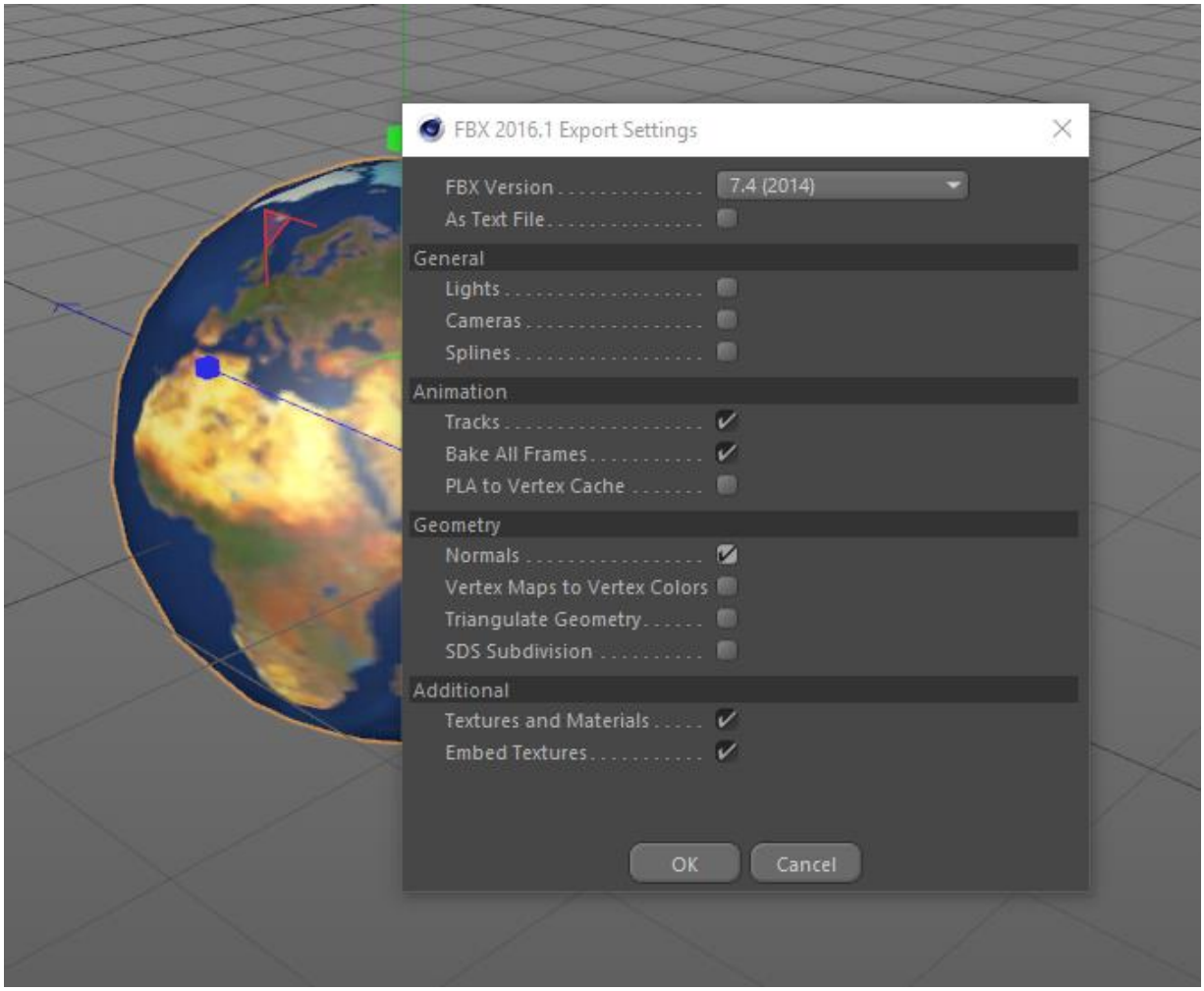

Рисунок 9. Правильная настройка экспорта трехмерной модели в формат FBX для ее дальнейшего использования в WebGL фреймворке.

Так как при любом способе взаимодействия пользователя, глобус должен всегда оставаться в центре окна, но при этом или он, или камера вокруг него должны крутиться. В результате экспериментов был выбран второй вариант с вращающеймя камерой, как наиболее гибкий и менее ресурсоемкий. При вращении объекта в реальном времени фреймворк должен заново просчитывать блики, отражения и другие физические свойства материалов,

а также положение других объектов относительно глобуса. В нашем же случае менять свое положение будет только камера, которая не имеет собой никакого «физического носителя», то есть является исключительно программной сущностью, не имеющей физических характеристик.

В итоге был выбран и доработан шаблон скрипта OrbitCameraControl, который позволяет по нажатию левой клавиши мыши вращать камеру вокруг центра сцены (в нашем случае точки с координатами {0; 0; 0}). В настройках камеры были модифицированы параметры инерции и скорости вращения, так как в задачи пользователя вряд ли входит необходимость крутиться вокруг глобуса более, чем на 360 градусов за раз, соответственно, более низкий показатель скорости улучшает точность позиционирования.

Подобным же образом было решено смоделировать и экспортировать некоторые декоративные элементы трехмерной карты (национальные флаги, деревья, опознавательные элементы и т. д.). Данные элементы не будут являться зонами взаимодействия пользователя с интерфейсом, а просто дополнять эстетическое восприятие при работе с картой.

Что касается источников освещения, то они были настроены таким образом, чтобы «глобус» не бликовал, что происходит с ним по умолчанию. Для этого уровень блика (Specular Level) был снижен до нуля, а в свойствах материала самого объекта был отключен слой, отвечающий за отображение бликов.

Итого на сцене было размещено и настроено в соответствии с ожидаемым поведением несколько объектов: трехмерных моделей, вращающаяся вокруг центра сцены камера и два источника освещения (основной и контровой).

2. Написание собственного шейдера для выделенных объектов.

Для объектов (которые представляют собой зоны взаимодействия) встала необходимость при действии пользователя менять цвет на отличающийся от изначального, например, на красный. Сделать это можно как загрузкой

новой текстуры (что достаточно ресурсозатратно), так и написанием шейдера – программного участка кода, отвечающего за определение параметров геометрии объектов или их отображения. Шейдеры бывают фрагментными и вершинными, в нашем случае именно первый вариант отвечает за изменение свойст цвета объекта. Фрагментный шейдер работает с фрагментами растрового изображения и текстурами — обрабатывает данные, связанные с пикселями (например, цвет, глубина, текстурные координаты). Используется на последней стадии графического конвейера для формирования конечного фрагмента изображения [14].

Шейдер пишется на специализированном языке GLSL, после чего полученный код встраивается в исходный JavaScript-код инсталляции при помощи текстовых литералов (заключенных в кавычки). В данной ситуации будет использоваться такой параметр, как «gl_FragColor», который как раз отвечает за базовую настройку цвета объекта. Свойства цвета задаются по шкале RGBA (Red, Green, Blue, Alpha) в пределах [0, 1]. Итоговый код фрагментного шейдера, встроенный в JavaScript-код приложения показан далее (см. рис. 10). После, было необходимо применять указанный шейдер при нажатии пользователем левой кнопкой мыши по определнному участку карты. Для этого использовалось встроенное в Goo Create событие onPick( ), срабатывающее при выборе (клике) объекта курсором мыши. Для того, чтобы сохранить изначальный материал объекта, и после перехода на таймлинию вернуть его назад к первоначальному виду, был применен следующий код, в котором в переменной ctx.oldMaterial хранятся данные о первоначальной версии материала, до применения шейдера (см. рис. 11).

```
var fragmentShader = [
    'void main(void){',
    ' gl_FragColor = vec4(0.8, 0.1, 0.1, 1.0);',
    '}'
].join('\n');
```

Рисунок 10. Код фрагментного шейдера, раскрашивающего объект в красный цвет.

```
// Save the current material for later.
ctx.oldMaterial = ctx.entity.meshRendererComponent.materials[0];
```

Рисунок 11. Сохранение первоначальных свойств материала в отдельной переменной.

3. Взаимодействие с DOM-объектами и методы интеграции получившегося результата в уже существующий информационный веб-ресурс.

Что же происходит с ресурсом после того, как пользователь нажал на соответствующий регион и тот окрасился в красный цвет? После этого производится переход к встроенному в фреймворк событию onPick (аналог события onClick() в обычном JavaScript), которое перемещает фокус в точку тайм-линии, где располагается элемент с определенным заранее заданным id (идентификатором). Данный переход осуществляется с помощью скрипта с довольно предсказуемым кодом: `document.getElementById( )`, при помощи которого можно обратиться к любому узлу DOM-дерева ресурса, зная наперед заданный ему id. Стоит отметить, что подобный вызов есть возможность осуществить только благодаря тому, что WebGL и Goo Create изначально базируются на той же самой технологии, что и исходный веб-ресурс, то есть HTML5 и JavaScript. Таким образом обе сущности «общаются» как бы на одном и том же языке, понимая структуру друг друга. В иных формулировках, можно сказать, что тайм-линия открывает доступ к своему содержимому по нескольким «ярлыкам» для удобства пользования.

Интеграция получившейся сцены в уже существующий ресурс возможна несколькими способами, среди которых можно выделить два наиболее доступных: внедрение имеющихся DOM элементов на трехмерное полотно раздела <Canvas>, то есть добавление новых двухмерных элементов в созданное трехмерное пространство, или же обратный вариант – вставка трехмерной сцены в двухмерный сайт методом кадрирования (тег <iframe>). В данной работе был выбран второй вариант, как более разумный с учетом того, что исходный проект «Наглядная история» уже имеет определенную

по сложности структуру, которую воспроизводить заново в трехмерном пространстве может быть затратно по времени и не совсем удобно в плане размещения и позиционирования объектов. Элемент <iframe>, напротив, позволяет удобно и практически без лишних настроек внедрить созданную сценку в качестве отдельного объекта на веб-страничке с заранее указанной шириной (в 100% документа) и высотой (40% документа), а оставшееся место по вертикали выделить для тайм-линии с событиями на ней. Таким образом получаются две, с одной стороны, совершенно самостоятельных части веб-сайта, с другой стороны, плотно интегрированные друг с другом. Как пример, верхнюю часть сайта можно использовать также для отображения логотипа или анимации, если карта не требуется постоянно.

4. Краткие итоги проделанной работы над созданием веб-инсталляции «Трехмерная карта мира».

В результате проделанной работы был реализован весь необходимый функционал для взаимодействия пользователя с информационным контентом проекта «Наглядная история», также новый интерфейс успешно интегрирован в качестве опционального в текущую версию данного проекта. Дополнительно были добавлены неинтерактивные элементы для создания небольшого декоративного эффекта, а для интерактивных регионов карты был написан специальный шейдер на языке GLSL, позволяющий при нажатии на эти регионы левой клавишей мыши окрашивать их в красный цвет. Предварительное тестирование показало, что финальная сцена занимает не более 50 Мб, и грузится в браузерах Microsoft Edge и Google Chrome не более, чем за 3-4 секунды (в зависимости от скорости Интернета).

ПРОЕКТНАЯ ЧАСТЬ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ. РАЗРАБОТКА ВЕБ-ИНСТАЛЛЯЦИИ «КОРЕЛА WEB»

Веб-инсталляция «Корела Web» создана по образцу настольной версии приложения, являющегося виртуальной трехмерной реконструкцией объекта культурного наследия – защитной крепости Корела. Проект сделан на базе достаточно популярного решения для создания интерактивных трехмерных программных продуктов – Unity3D. Но на момент написания выпускной квалификационной работы, экспорт в WebGL был добавлен разработчиками в незавершенном режиме (бета-версии), поэтому говорить о его производительности практически не стоит. Тем не менее, в конце данного раздела будет приведено сравнение характеристик двух проектов – сделанного при помощи фреймворка Goo Create и экспортированного при помощи встроенных инструментов Unity 5. Тем не менее, необходимость перенести проект реконструкции в веб-среду несет собой несколько причин. Во-первых, для его просмотра не потребуется никаких дополнительных установок плагинов или программ-просмотрщиков, драйверов и прочих компонентов, которые могут понадобиться при использовании настольной версии приложения. Во-вторых, от пользователя не требуется иметь права администратора на компьютере, которые, например, могут быть запрещены политикой безопасности на рабочем месте. В-третьих, проект полностью хранится на удаленном сервере и не отнимает локального места на жестком диске пользователя. В-четвертых, проект, находящийся в веб-среде могут автоматически просматривать пользователи любой ОС, имеющей браузеры, поддерживающие стандарт WebGL, среди них Windows, Mac OS, Linux. При использовании настольной версии приложения, под каждую отдельную ОС проект пришлось бы «пересобирать» (компилировать заново), так как в

каждой из них используются свои собственные библиотеки, к которым обращается программа по выводу графики на дисплей и т. д. Таким образом, пользователь получает множество неоспоримых преимуществ, а недостаток, по большому счету, выясняется лишь один – относительно низкая производительность, которая отчасти «лечится» проведением грамотной оптимизации графической структуры приложения.

Итак, изначально имеется некая версия приложения для Unity 5, есть задача перенести его в веб-среду при помощи фреймворка Goo Create. Примерный алгоритм работы здесь следующий:

- Загрузка файлов проекта в Unity 5, разбор ресурсов, из которых состоит проект, их поиск и сбор в единую директорию.
- Анализ моделей и текстур проекта, проведение базовой оптимизации (уменьшение разрешения текстур, уменьшение количества полигонов у моделей – при необходимости). Соблюдение баланса между внешней неизменностью проекта и скоростью работы в веб-браузере.
- Композитинг полученных элементов согласно плану и структуре изначальной настольной версии в Goo Create.
- Создание и настройка камер и освещения, написание скриптов для создания эффекта «прогулки» по месту реконструкции.
- Тестирование и финальная оптимизация проекта, сравнение с результатами автоматического экспорта средствами Unity 5.

Первоначальная версия проекта имеет достаточно сложную иерархию каталогов, содержащую в себя различные ресурсы – среди них наиболее большой интерес представляют трехмерные модели и текстуры к ним. Для их поиска было решено воспользоваться стандартными средствами Unity 5.

Для этого каждый раз в окне редактора выделялся необходимый объект и в контекстном меню выбирался пункт «Select Prefab». После чего в нижней части окна редактора, где расположено меню со всеми ресурсами проекта, показывался префаб выбранной модели, по которому уже можно было отследить, где находится модель в директории на локальном диске (при помощи пункта контекстного меню «Show in Explorer...»). Таким образом было выяснено, что большинство нужных ресурсов хранится в директории «Korela_sources». Но практически все модели и материалы, которые там находятся, сохранены в формате файла, понятного лишь программе Unity 5.

Что делать? После ряда экспериментов было выяснено, что все ресурсы в указанной пользователем директории можно запаковать в единый Unity архив, для использования их в других проектах. После создания такого архива с ресурсами, в Unity 5 был создан новый, пустой проект, и уже в него был импортирован запакованный архив. После распаковки в папке нового проекта появились модели и текстуры в их изначальном формате, в котором они когда-то по всей видимости были интегрированы в настольную версию.

После завершения данного эксперимента и проверки того факта, что в папке нового проекта находятся примерно все модели, необходимые для переноса в веб-среду, следующим шагом было решено просмотреть все ресурсы, чтобы понять примерный объем и «тяжеловестность» будущей сцены.

Официальные рекомендации разработчиков Goo Create [15] рекомендуют ограничить объем импортируемых ресурсов насколько это возможно. Для соблюдения баланса между внешней неизменностью проекта и скоростью работы в веб-браузере был составлен следующий перечень действий по оптимизации графической структуры проекта:

- Текстуры по возможности ужимались до размеров 512*512, после чего сохранялись в формат .jpeg с 80% уровнем качества. Таким образом удалось уменьшить размер массива текстур практически в четыре раза.

- Количество полигонов в моделях замерялось стандартными средствами студенческой версии программы 3Ds Max 2016, после чего к некоторым из них применялся модификатор ProOptimizer (см. рис. 12), позволяющий уменьшить количество полигонов без резких изменений в геометрии объекта. Фактор уменьшения составлял обычно около 70-80%, при установки ниже, модели начинали терять внешнюю привлекательность или появлялись ошибки в их геометрии (пропадали различные элементы).

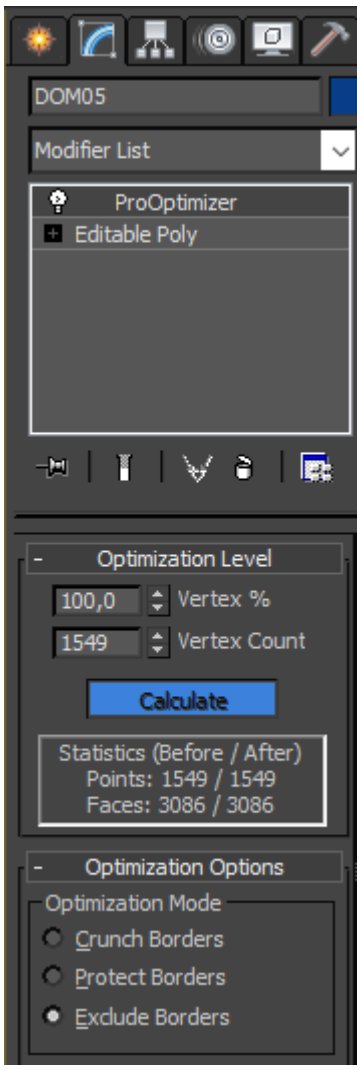

Рисунок 12. Применение модификатора ProOptimizer для уменьшения количества полигонов в объектах.

- Все ресурсы, не относящиеся к текстурам и моделям (например, звуки и мета-данные объектов) удалялись для уменьшения веса проекта.

- Количество источников света было решено установить равное трем – атмосферный свет и два локальных источника освещения.

- Анимированная вода в веб-версии была заменена максимально реалистичной текстурой воды без эффекта анимации.

- Все объекты на данном этапе разработки было решено оставить статическими – то есть без использования различных физических свойств, вроде симуляции силы тяжести, сопротивления, инерции и т. д.

После проделанных действий для оптимизации всех ресурсов проекта, они импортировались в Goo Create, после чего в результате тщательной сверки с оригиналом, расставлялись по сцене. Наиболее сложной частью оказалась сверка настроек материалов – так как автоматически они не сохранялись из Unity 5 проекта и единственной возможностью соблюсти соответствие было выставление точно таких же параметров вручную в редакторе Goo Create.

По завершению импорта моделей и настройки материалов, в сцену были добавлены три источника освещения, skybox (специальная текстура, призванная заменить или имитировать собой небесный свод) с эффектом вечернего заката, а также добавлена камера. К ней был прикреплен написанный скрипт, позволяющий перемещать ее по нажатию клавиш [WASD] на клавиатуре и вращать вокруг своей оси при помощи мыши. Как уже было сказано выше, в целях оптимизации, объекты сцены не были наделены каким-либо физическими свойствами, поэтому движения и перемещения камеры было решено искусственно ограничить по вертикальной оси, чтобы она не «проваливалась» под землю. Для этого было решено написать отдельный скрипт, следящий за изменениями координаты Y, и при изменении значения ниже нуля возвращающий камеру на изначальное положение. В верхней плоскости проекта камера двигается свободно.

В результате всех проделанных действий была создана демонстрационная сетевая версия проекта реконструкции «Корела», которая успешно запустилась в тестируемых браузерах Google Chrome, Mozilla Firefox и Microsoft Edge. Примерное время запуска было около 6-10 секунд, а конечный вес проекта составил чуть менее 200 Мб. Теперь стоит сравнить полученные результаты реализованного приложения с аналогичными у автоматического экспорта в WebGL средствами Unity 5. Автоматический экспорт (после некоторых настроек камер, освещения и материалов) сумел сгенерировать проект весом в 600 с лишним Мб, а время запуска такого архива составило в районе 30 секунд, порой даже больше, что, разумеется, для веб-проекта не очень подходит, потому как пользователь, привыкший к практически моментальной загрузке современных веб-ресурсов вряд ли станет ждать полминуты, смотря на полосу загрузки. Тем не менее, стоит отметить, что производительность и в данном случае была достаточно приемлимая во всех тестируемых браузерах (около 40-60 кадров в секунду).

В качестве небольшого итога можно сделать вывод о том, что фреймворк Goo Create, специализирующийся исключительно на технологии WebGL, смог опередить по характеристикам выходного файла экспорт из Unity 5, что еще раз подтверждает актуальность и прикладную значимость проведенного в начале работе исследования технологических решений. В ходе реализации данного проекта была произведена большая работа по переносу настольной версии виртуальной реконструкции в веб-среду: разбор всех ресурсов, разработка требований для оптимизации, проведение действий по оптимизации применительно к каждой сущности, скурпулезная настройка сцены в редакторе, написание программной части, тестирование в трех современных браузерах на предмет корректности запуска и работы. Теперь веб-инсталляция «Корела Web» доступна любому Интернет пользователю прямо из окна браузера, без установки плагинов и дополнительного ПО.

ЗАКЛЮЧЕНИЕ

В заключении хотелось бы упомянуть, что все поставленные в начале выпускной квалификационной работы задачи были успешно выполнены, а именно:

- Проанализированы современные технологии для реализации трудоемких графических задач в веб-среде.
- Выработаны критерии для поиска необходимого для проектов фреймворка, осуществлено исследование наиболее современных решений, сделан выбор на основе сравнительной таблицы характеристик.
- Созданы веб-инсталляции "Трехмерная карта мира" для проекта гуманитарной направленности "Наглядная история" и "Корела Web" для проекта трехмерной виртуальной реконструкции крепости Корела.
- Проведена оптимизация графической структуры проектов для улучшения скорости их работы в веб-браузерах, осуществлено тестирование.

В результате работы были разработаны следующие технологические решения и алгоритмы:

- Интеграция WebGL-проекта с элементами DOM существующего сайта
- Фрагментный шейдер на языке GLSL для выделенных объектов сцены
- Критерии для оптимизации тяжелых, графически нагруженных проектов
- Различные скрипты, написанные на языке JavaScript, среди которых скрипты настроек поведения камеры, изменения свойств объектов и т. д.

Обе цели были достигнуты – результаты проведенного исследования были успешно проверены при реализации двух практических проектов.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Chart: Bye Bye Flash! [Электронный ресурс] / Statista.com – URL: https://www.statista.com/chart/3796/websites-using-flash (дата обращения: 04.05.2016)

2. Thoughts on Flash [Электронный ресурс] / Apple.com – URL: http://www.apple.com/hotnews/thoughts-on-flash (дата обращения: 04.05.2016)

3. Global Threat Intelligence Report [Электронный ресурс] / Solutionary.com – URL: https://www.solutionary.com/_assets/pdf/research/2015-gtir.pdf (дата обращения: 04.05.2016)

4. За 2015 год в Flash Player нашли 316 уязвимостей [Электронный ресурс] / Nplus1.ru – URL: https://nplus1.ru/news/2015/12/29/oh-flash (дата обращения: 04.05.2016)

5. The Final Countdown for NPAPI [Электронный ресурс] / Chromium.org – URL: https://blog.chromium.org/2014/11/the-final-countdown-for-npapi.html (дата обращения: 04.05.2016)

6. WebGL 2 Specification [Электронный ресурс] / Khronos.org – URL: https://www.khronos.org/registry/webgl/specs/latest/2.0 (дата обращения: 08.05.2016)

7. Experince Curiosity [Электронный ресурс] / NASA.gov – URL: https://eyes.nasa.gov/curiosity (дата обращения: 08.05.2016)

8. Моичи К., Ли Р. WebGL: программирование трехмерной графики / Кацуда Моичи, Роджер Ли; перевод с англ. А. Н. Киселев – М.: ДМК Пресс, 2015. – 494 с. – ISBN 978-5-97060-146-4 (рус.)

9. WebGL Stats. Updated: Thursday, 04 Feb 2016 [Электронный ресурс] / Webglstats.com. – URL: http://webglstats.com (дата обращения: 10.05.2016)

10. JavaScript V8 versus C++ [Электронный ресурс] / Debian.org – URL: http://benchmarksgame.alioth.debian.org/u64/compare.php?lang=v8&lang2=gpp (дата обращения: 10.05.2016)
11. Chrome Experiments [Электронный ресурс] / Chromeexperiments – URL: https://www.chromeexperiments.com/webgl (дата обращения: 10.05.2016)
12. Jos Dirksen. Three.js Cookbook – Packt Publishing Ltd., 2015. – 300 с. – ISBN: 978-1-78398-118-2
13. User Contributions [Электронный ресурс] / Khronos.org. – URL: https://www.khronos.org/webgl/wiki/User_Contributions (дата обращения: 14.05.2016)
14. Andreas Anyuru. Professional WebGL Programming: Developing 3D Graphics for the Web – Wrox Press, 2012. – 361 с. – ISBN: 978-1119968863
15. Optimization [Электронный ресурс] / Goocreate.com – URL: https://learn.goocreate.com/tutorials/create/optimization (дата обращения: 14.05.2016)

48

Выпускная квалификационная работа выполнена мною самостоятельно. Использованные в работе материалы из опубликованной научной, учебной литературы и Интернет имеют ссылки на них.

Отпечатано в 1 экземплярах.
Библиография 15 наименований.
Один экземпляр сдан на кафедру.

Сколов Юрий Владимирович _____

Дата 23.05.2016 г.