МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им.В.П.АСТАФЬЕВА
(КГПУ им.В.П.Астафьева)

Институт/факультет Институт математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая кафедра Базовая кафедра информатики и информационных технологий в образовании
(полное наименование кафедры)

Атер Артур Александрович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема Проектирование и реализация интерактивных моделей для цифровых образовательных ресурсов по школьному курсу информатики

Направление подготовки 44.03.01 Педагогическое образование
(код и наименование направления)

Профиль Информатика
(наименование профиля для бакалавриата)

ДОПУСКАЮ К ЗАЩИТЕ
Заведующий кафедрой
д.п.н., профессор Пак Н.И.
(ученая степень, ученое звание, фамилия, инициалы)

(дата, подпись)

Руководитель к.ф-м.н., доцент кафедры ИИТвО Романов Дмитрий Валерьевич
(ученая степень, ученое звание, фамилия, инициалы)

Дата защиты

Обучающийся Атер А. А.
(фамилия, инициалы)

(дата, подпись)

Оценка
(прописью)

КРАСНОЯРСК 2017

Оглавление

Введение

В настоящее время одной из основных тенденций развития информатизации образования, становится комплексное использование интерактивных технологий.

Хорошо известно, что существует как минимум две модели применения интерактивных технологий в обучении. Первая модель - использование интерактивных технологий в рамках традиционной системы обучения как интерактивного средства поддержки и обеспечения учебного процесса. Эта модель была действующей пятнадцать или двадцать лет назад, что было вполне оправдано. Дидактические возможности современных интерактивных средств обучения настолько широки, что эта модель не позволяет в полной мере повысить эффективность и качество образования. Сейчас более актуальна вторая модель - построение образовательного процесса на основе целей обучения и дидактических возможностей интерактивных средств обучения. Это даёт возможность в полной мере раскрыть дидактические возможности интерактивных средств обучения, позволяющих значительно повысить качество образования.

Актуальность исследования. В условиях модернизации и информатизации образования учитель информатики должен являться наиболее активным участником данного процесса, поэтому перед профессиональной подготовкой учителя стоит задача формирования не только предметных знаний и умений, но и учителя, готового решать актуальные задачи образования современного этапа информатизации образования. Среди данных проблем выделяются следующие: создание методической системы обучения, направленной на развитие интеллектуального потенциала обучающегося, на формирование умений самостоятельно овладевать знаниями и умениями, осуществлять

аналитическую деятельность, а также деятельность по обработке, сбору, хранению, передаче информации; разработка исследовательских, демонстрационных электронных образовательных средств, включая, образовательные программные средства; системы и средства автоматизации процессов обработки учебного материала, в том числе и «виртуального». В решении подобных задач особое значение имеет информационно-аналитическая деятельность, на основе которой учитель информатики и ИКТ сможет решать указанные задачи с помощью моделирования для процесса обучения предметных, информационных, математических, компьютерных и многих других обучающих моделей на основе ИКТ. Многие разделы дисциплины содержат материал:

- сложной структуры
- высокой степени абстракции
- с большим количеством иерархических уровней

Представление такого материала в ЦОР представляет серьёзный дидактический вызов.

Цель исследования: получение учеником понимания структуры системы (иерархия, действующие элементы) и схемы действующих связей (связи между элементами иерархии) в ходе моделирования.

Объектом исследования является процесс обучения информатике.

Предметом исследования является приемы моделирования и применения цифровых образовательных ресурсов в обучении информатике.

Гипотеза исследования заключается в том, что организация процесса обучения информатике будет более эффективной, если в данном процессе учтены особенности применения информационных технологий и

применяются специально смоделированные цифровые образовательные ресурсы.

Исходя из темы, цели и гипотезы, были поставлены следующие задачи исследования:

1. проанализировать методическую и научную литературу по проблеме исследования;

2. уточнить сущность и содержание понятия «цифровые образовательные ресурсы», решающие актуальные задачи информатизации образования;

3. определить и охарактеризовать этапы моделирования;

4. разработать методические рекомендации по моделированию и применению цифровых образовательных ресурсов.

Научная новизна исследования заключается в том, что: теоретически обоснована целесообразность использования разработанных цифровых образовательных ресурсов учителями информатики, которое является неотъемлемой частью профессиональных умений педагога.

Практическая значимость исследования: разработанная методика моделирования и применения цифровых образовательных ресурсов на уроках информатики.

Структура выпускной квалификационной работы отражает логику, содержание и результаты исследования и состоит из введения, двух глав, заключения, списка литературы и приложений.

Глава 1 Теоретические основы моделирования

1.1 Определение и классификация цифровых образовательных ресурсов

Сегодня мы можем наблюдать очередную смену парадигм: теперь вместо традиционных "электронных книг" нам предлагается разрабатывать и использовать электронные ресурсы. ЦОР представляет собой цифровой образовательный программный продукт? Какие их преимущества по сравнению с программами, которые использовались раньше?

"Официальная" терминология, принятая, в частности, в документах НФПК (Национального фонда подготовки кадров) при организации различных грантовых программ и тендеров на разработку программных средств образовательного назначения, предусматривает более узкие и жесткие рамки понимания этого названия. Согласно этой терминологии, в настоящее время предлагается к разработке и применению в учебном процессе три категории подобных средств:

ЦОРы – "цифровые содержательные модули", поддерживающие изучение конкретного фрагмента соответствующей учебной темы, жестко привязанной к конкретному учебнику в этой теме и сопровождается соответствующим методическим обеспечением;

Тестирование ("инновационные учебно-методические комплексы") – совокупность электронного компонента (обязательно покрывающего весь спектр тем в рамках базовой учебной программы для соответствующего возрастного уровня, реализующего все требуемые функции (от предоставления учебного материала до контроля знаний), содержащий "инновационный" потенциал, чтобы радикально улучшить учебный процесс) и "бумажного" методического сопровождения;

ИИСС (информационные источники сложной структуры) – своего рода аналог рубрики «разное», куда могут быть занесены разные информационные объекты, затрагивающие лишь часть тем базового стандарта, расширяющие их, предоставляющие дополнительный справочный материал, часто – носящие комплексный, интегративный характер и не обязательно привязанные к учебникам [15].

Классификация Цифровых образовательных ресурсов

Информационное содержательное обеспечение в Информационно-телекоммуникационном сопровождений. (ИТС) включает две группы ЦОР:

1) Информационные источники:

оригинальные тексты (хрестоматии; тексты из специальных словарей и энциклопедий; тексты из научной, научно-популярной, учебной, художественной литературы и публицистики….) не повторяющие стабильные учебники;

статические изображения (галереи портретов ученых соответствующей предметной области; «плакаты» – изображения изучаемых объектов и процессов и пр.);– динамические изображения (изучаемые процессы и явления в пространственно-временном континиуме – кино- и видеофрагменты, анимационные модели на CD, DVD);

мультимедиа среды (информационно-справочные источники. практикумы (виртуальные конструкторы), тренажеры и тестовые системы, программированные учебные пособия («электронные учебники», виртуальные экскурсии и пр.).

2) информационные инструменты – это информационные средства, обеспечивающие работу с информационными источниками.

Как правило, информационные источники включают отдельные информационные объекты (элементарные информационные объекты), которые при возможности их выделения могут самостоятельно использоваться в рамках ИТС.

Элементарные информационные объекты могут рассматриваться:

как органичный компонент традиционного учебного процесса, не заменяющий, а дополняющий и расширяющий возможности традиционных, методически целесообразные средства обучения, повышая тем самым эффективность, качество обучения;

как объекты проектирования учебно-информационной среды в рамках педагогического дизайна с использованием инструментальных средств, что позволит повысить эффективность использования ИТС в учебном процессе.

Законченными полными источниками информации цифровых продуктов, охватывающих весь курс или раздел (тема) будет рассматриваться как существенный компонент, который определяет их основные функции.

Информационно-образовательные конечные оцифрованные продукты (оригинальные тексты, не повторяющие стабильные учебники) рассматриваются как дополнительные к основным.

Виды ЦОР по образовательно-методическим функциям.

1) Электронные учебники:

Прототипы традиционных учебников; оригинальные электронные учебники; предметные обучающие системы; предметные обучающие среды.

2) Электронные учебные пособия:

Репетиторы; тренажеры; обучающие; обучающие – контролирующие; игровые; интерактивные; предметные коллекции; справочники, и словари; практические и лабораторные.

3) Электронные учебно-методические комплексы (УМК):

Предметные миры; программно-методические комплексы; предметные учебно-методические среды; инновационные УМК.

4) Электронные издания контроля:

Тесты; тестовые задания; методические рекомендации по тестированию; инструментальные средства.

Классификация ЦОР по типу информации

1) ЦОР с текстовой информацией:

Учебники и учебные пособия; первоисточники и хрестоматии; книги для чтения; задачники и тесты; словари; справочники; энциклопедии; периодические издания; нормативно-правовые документы; числовые данные; программно- и учебно – методические материалы.

2) ЦОР с визуальной информацией:

a. Коллекции: иллюстрации; фотографии; портреты; видеофрагменты процессов и явлений; демонстрации опытов; видеоэкскурс.

b. Модели: 2-3 –х мерные статические и динамические; объекты виртуальной реальности; интерактивные модели.

c. Символьные объекты: схемы; диаграммы; формулы.

d. Карты для предметных областей.

3) ЦОР с комбинированной информацией:

Учебники; учебные пособия; первоисточники и хрестоматии; книги для чтения; задачники; энциклопедии; словари; периодические издания.

4) ЦОР с аудио информацией:

Звукозаписи выступлений; звукозаписи музыкальных произведений; звукозаписи живой природы; звукозаписи неживой природы; синхронизированные аудио объекты.

5) ЦОР с аудио и видео информацией:

Аудио – видео объекты живой и неживой природы; предметные экскурсии; энциклопедии.

6) Интерактивные модели:

Предметные лабораторные практикумы; предметные виртуальные лаборатории.

7) ЦОР со сложной структурой:

Учебники; учебные пособия; первоисточники и хрестоматии; энциклопедии.

Педагогические инструменты цифровых образовательных ресурсов:

Интерактив (взаимодействие) – поочередные высказывания (от выдачи информации до произведенного действия) каждой из сторон. Причем каждое высказывание производится с учетом как предыдущих собственных, так и высказываний другой стороны.

Мультимедиа - представление ресурсов и процессов не традиционном текстовым описанием, а с помощью фото, видео, графики, анимации, звука.

Моделинг - моделирование реальных ресурсов и процессов с целью их исследования.

Коммуникативность - возможность непосредственного общения, оперативность предоставления информации, контроль за состоянием процесса.

Производительность - автоматизация нетворческих, рутинных операций, отнимающих у человека много сил и времени. Быстрый поиск информации по ключевым словам в базе данных, доступ к уникальным изданиям справочно-информационного характера [16].

1.2 Современные цифровые образовательные ресурсы в обучении

На сегодняшний день Россия находится на 45-м месте по использованию информационных и коммуникационных технологий в ключевых сферах жизни общества [25]. В международных индексах готовности к электронному развитию наша страна занимает одно из последних мест среди индустриальных стран по такому показателю, как обучение с использованием ИКТ [26] . Даже имеющиеся в школах ресурсы используются крайне неэффективно.

Школа, как и любая сложная система, состоит из многих подсистем. Среди них выделяется основная – сам процесс обучения, являющийся главной и определяющей частью всей системы.

Компьютеризация процесса обучения- это процесс оснащения образовательных учреждений средствами современной вычислительной техники. Компьютеризация – это технический процесс.

Информатизация процесса обучения - это процесс, направленный на оптимальное использование информационного обеспечения процесса обучения с помощью компьютера. Компьютер дает возможность по- новому построить информационное обеспечение и повысить качество образования. Главная задача - извлечь из этого оборудования максимальную пользу.

Одна из задач проекта ИСО – обеспечить школы необходимыми ей цифровыми ресурсами создать условия для их активного использования в учебной деятельности.

Для обеспечения качества создаваемых учебных материалов большое внимание уделяется их апробации непосредственно в условиях учебного процесса.

Какие изменения вносит ЦОР в учебный процесс.

Повышает эффективность учебного процесса за счёт внесения разнообразия на разных этапах урока.

Даёт богатый дополнительный материал для подготовки к уроку учителю и учащимся

Позволяет показать некоторые процессы в динамике (видеофрагменты, анимация).

Усиливает наглядность

Вместо старых таблиц - «культурное» изображение.

Показ объектов, которые другим способом показать нельзя.

Качественное закрепление и отработка навыков у большого числа учащихся при использовании локальной сети.

Повышает интерес учащихся, особенно интерактивные объекты.

Но на данный момент не все ЦОРы имеют нужное качество.

По результатам апробации нужна доработка продуктов с учетом замечаний и предложений учителей - апробаторов.

Кроме готовых образовательных ресурсов в школах учителя-предметники создают собственные.

Учитель, чтобы не отставать от времени, должен постоянно учиться, обмениваться опытом, видеть опыт работы своих коллег. В этом окажет помощь национальная коллекция цифровых образовательных ресурсов. В итоге каждый учитель сможет получить дидактические материалы к каждому уроку - вместе с методическими рекомендациями по их использованию. Работа по созданию коллекции начата, ЦОРы появились по отдельным предметам, но, к сожалению в них допускаются ошибки.

Назрела необходимость появления в школах учебно - методических комплексов. Это позволит учителю работать творчески и применять новые современные средства. [17].

Использование ЦОР, как активной формы работы на уроках

Что даёт применение электронных пособий для учителя и учащихся?

Для учителя, конечно же, дополнительную нагрузку. Но это с одной стороны. С другой стороны, это вносит разнообразие в урок. Он становится насыщеннее формами и интереснее. Для учащихся – это интерес к предмету, возможность отработать пробелы, исправить отметки. Для слабых и замкнутых ребят работа на компьютере иногда полезнее работы с сильным одноклассником: он спокойнее, никто его не торопит, не насмехается. Со временем такие дети становятся увереннее в себе и преодолевают барьер в общении.

Сложная, но возможная, форма применения ЦОРов - использование их в интерактивных, инновационных методах обучения: создание учебных мини-проектов, рациональный поиск информации в Интернет, использование материалов ЦОРов для подтверждения выдвинутых учебных гипотез. Рассмотрим применение ЦОР на различных этапах урока:

ЦОР для актуализации знаний

Электронные тесты

ЦОР на этапе объяснения нового материала.

В основе деятельности лежит личностное включение учащегося в процесс, когда компоненты деятельности им самим направляются и контролируются. Стимул к обучению реализуется через внесение элемента новизны, который отвлекает детей от трудностей, увлекая и пленяя их своей

необычностью, использованием своеобразных средств. Такими элементами новизны являются, например:

Электронные учебники;

Мультимедийные презентации;

Учебные видеофильмы.

ЦОР для контроля и оценки знаний, умений и навыков.

Проектная деятельность

Программы тренажёры.

Для аттестации учащихся используется, как традиционная форма, так и компьютерный вариант (с использованием подготовленных при помощи системы ЦОР контрольных работ и тестов).

ЦОР для подготовки домашнего задания

Творческие задания

Рефераты, доклады

Презентации

Самообучение

Нельзя рассматривать ЦОР только как новые образовательные возможности. Они формируют новые умения и навыки. У учеников появилась возможность использовать другие материалы для подготовки к уроку и самоподготовки. Именно образовательный процесс с применением ЦОР изменяет школьника. Результаты процесса выражены в достижениях (учебных и личностных) ученика. Прежде всего, скорей всего происходит не процесс приобретения новых знаний, а процесс формирования новых умений и навыков. Именно на такой результат и должны быть ориентированы уроки с применением ЦОРов.

Использования ЦОР приводит к изменению в содержании образования, технологии обучения и отношениях между участниками образовательного процесса [18].

1.3 Общее представление о моделях и моделировании

Понятие «модель» в повседневной жизни чаще ассоциируется с «макетом», который обладает внешним или функциональным сходством с определенным объектом. Макеты, модели создаются для того, чтобы, не имея реального объекта, рассмотреть, как он выглядит и не имея возможности манипулировать с реальным объектом, пробовать производить какие-либо действия с объектом, имитирующим его. В результате наблюдений модели и манипуляций с ней можно получить новые знания о реальном объекте. Если это уже известные человечеству сведения, то модель использовалась для обучения. Если новое знание получено впервые, то совершается акт познания мира человечеством. В результате познания человечество, как правило, приходит к более совершенной модели изучаемого объекта, точнее соответствующей реальному объекту.

Объект, в общенаучном смысле, – «определенная часть окружающей нас реальной действительности (предмет, процесс, явление)» [5], или «некоторая часть окружающего нас мира, которая может быть рассмотрена как единое целое» [6]. Заметим, что эта трактовка понятия «объект» избавляет от необходимости в многочисленных высказываниях, связанных с объектами, перечислять триаду «предмет, процесс, явление», как это делается в большинстве учебников. Объект это то, на что направлено внимание познающего субъекта. Это то, что может быть вычленено в окружающем мире.

«Процесс – последовательная смена состояний объекта в результате произведенных действий» [6]. Но процесс сам по себе может быть объектом рассмотрения, частью окружающего мира, т.к. мир существует как в пространстве, так и во времени. Поэтому перечисление «объекты и процессы» [7] отождествляет понятия «объект» и «предмет».

Явление это обнаружение объекта, внешней формы его существования. Школьные учебники информатики ничего не объясняют по этому поводу, но можно предположить, что здесь подразумеваются физические, химические, биологические, социальные и прочие явления. Явление может быть обнаружено, если его можно отличить от других явлений. Для этого необходимо сравнивать параметры, признаки и свойства все тех же предметов и процессов, т.е. объектов. Явление, будучи вычлененным из окружающего мира и рассматриваемое как единое целое, тоже может быть названо объектом. Таким образом, будем считать излишним перечисление в определениях и рассуждениях, относящихся к объекту вообще, таких объектов как предмет, процесс, явление.

Моделирование – процесс создания модели, точнее – исследование какого-либо объекта путем построения и изучения его модели; использование моделей для определения или уточнения характеристик и рационализации способов построения вновь конструируемых объектов.

Формальная система, эквивалентная реальному объекту, является моделью этого объекта. Формализация – процесс построения формальной системы – один из методов моделирования.

1.4 Общая классификация видов и этапов моделирования

Каждая модель создается для конкретной цели и, следовательно, уникальна. Однако наличие общих черт позволяет сгруппировать все их многообразие в отдельные классы, что облегчает их разработку и изучение. В теории рассматривается много признаков классификации, и их количество не установилось. Тем не менее, наиболее актуальны следующие признаки классификации:

характер моделируемой стороны объекта;

характер процессов, протекающих в объекте;

способ реализации модели.

Классификация моделей и моделирования по признаку "характер моделируемой стороны объекта"

В соответствии с этим признаком модели могут быть:

функциональными (кибернетическими);

структурными;

информационными.

Функциональные модели отображают только поведение, функцию моделируемого объекта. В этом случае моделируемый объект рассматривается как "черный ящик", имеющий входы и выходы. Физическая сущность объекта, природа протекающих в нем процессов, структура объекта остаются вне внимания исследователя, хотя бы потому, что неизвестны. При функциональном моделировании эксперимент состоит в наблюдении за выходом моделируемого объекта при искусственном или естественном изменении входных воздействий. По этим данным и строится модель поведения в виде некоторой математической функции.

Структурное моделирование - это создание и исследование модели, структура которой (элементы и связи) подобна структуре моделируемого объекта. Как мы выяснили ранее, подобие устанавливается не вообще, а относительно цели исследования. Поэтому она может быть описана на разных уровнях рассмотрения. Наиболее общее описание структуры - это топологическое описание с помощью теории графов.

Классификация моделей и моделирования по признаку "характер процессов, протекающих в объекте"

По этому признаку модели могут быть детерминированными или стохастическими, статическими или динамическими, дискретными или непрерывными или дискретно-непрерывными.

Детерминированные модели отображают процессы, в которых отсутствуют случайные воздействия.

Стохастические модели отображают вероятностные процессы и события.

Статические модели служат для описания состояния объекта в какой-либо момент времени.

Динамические модели отображают поведение объекта во времени.

Дискретные модели отображают поведение систем с дискретными состояниями.

Непрерывные модели представляют системы с непрерывными процессами.

Дискретно-непрерывные модели строятся тогда, когда исследователя интересуют оба эти типа процессов.

Очевидно, конкретная модель может быть стохастической, статической, дискретной или какой-либо другой.

Классификация моделей и моделирования по признаку "способ реализации модели"

Согласно этому признаку модели делятся на два обширных класса:

абстрактные (мысленные) модели;

материальные модели.

Нередко в практике моделирования присутствуют смешанные, абстрактно - материальные модели.

Абстрактные модели представляют собой определенные конструкции из общепринятых знаков на бумаге или другом материальном носителе или в виде компьютерной программы.

Абстрактные модели, не вдаваясь в излишнюю детализацию, можно разделить на:

- символические;
- математические.

Символическая модель - это логический объект, замещающий реальный процесс и выражающий основные свойства его отношений с помощью определенной системы знаков или символов. Это либо слова естественного языка, либо слова соответствующего тезауруса, графики, диаграммы и т. п.

Символическая модель может иметь самостоятельное значение, но, как правило, ее построение является начальным этапом любого другого моделирования.

Математическое моделирование - это процесс установления соответствия моделируемому объекту некоторой математической конструкции, называемой математической моделью, и исследование этой модели, позволяющее получить характеристики моделируемого объекта.

Математическое моделирование - главная цель и основное содержание изучаемой дисциплины.

Математические модели могут быть:

аналитическими;

имитационными;

смешанными (аналитико-имитационными).

Аналитические модели - это функциональные соотношения: системы алгебраических, дифференциальных, интегро-дифференциальных уравнений, логических условий. Уравнения Максвелла - аналитическая модель электромагнитного поля. Закон Ома - модель электрической цепи.

Преобразование математических моделей по известным законам и правилам можно рассматривать как эксперименты. Решение на основе аналитических моделей может быть получено в результате однократного просчета безотносительно к конкретным значениям характеристик ("в общем виде"). Это наглядно и удобно для выявления закономерностей. Однако для сложных систем построить аналитическую модель, достаточно полно отражающую реальный процесс, удается не всегда. Тем не менее, есть процессы, например, марковские, актуальность моделирования которых аналитическими моделями доказана практикой.

Имитационное моделирование. Создание вычислительных машин обусловило развитие нового подкласса математических моделей - имитационных.

Имитационное моделирование предполагает представление модели в виде некоторого алгоритма - компьютерной программы, - выполнение которого имитирует последовательность смены состояний в системе и таким образом представляет собой поведение моделируемой системы.

Процесс создания и испытания таких моделей называется имитационным моделированием, а сам алгоритм - имитационной моделью.

В чем заключается отличие имитационных и аналитических моделей?

В случае аналитического моделирования ЭВМ является мощным калькулятором, арифмометром. Аналитическая модель решается на ЭВМ.

В случае же имитационного моделирования имитационная модель - программа - реализуется на ЭВМ.

Имитационные модели достаточно просто учитывают влияние случайных факторов. Для аналитических моделей это серьезная проблема. При наличии случайных факторов необходимые характеристики моделируемых процессов получаются многократными прогонами (реализациями) имитационной модели и дальнейшей статистической обработкой накопленной информации. Поэтому часто имитационное моделирование процессов со случайными факторами называют статистическим моделированием.

Если исследование объекта затруднено использованием только аналитического или имитационного моделирования, то применяют смешанное (комбинированное), аналитико-имитационное моделирование. При построении таких моделей процессы функционирования объекта декомпозируются на составляющие подпроцессы, и для которых, возможно, используют аналитические модели, а для остальных подпроцессов строят имитационные модели.

Материальное моделирование основано на применении моделей, представляющих собой реальные технические конструкции. Это может быть сам объект или его элементы (натурное моделирование). Это может быть специальное устройство - модель, имеющая либо физическое, либо геометрическое подобие оригиналу. Это может быть устройство иной физической природы, чем оригинал, но процессы в котором описываются аналогичными математическими соотношениями. Это так называемое аналоговое моделирование. Такая аналогия наблюдается, например, между колебаниями антенны спутниковой связи под ветровой нагрузкой и колебанием электрического тока в специально подобранной электрической цепи.

Нередко создаются материально-абстрактные модели. Та часть операции, которая не поддается математическому описанию, моделируется материально, остальная - абстрактно. Таковы, например, командно-штабные учения, когда работа штабов представляет собой натурный эксперимент, а действия войск отображаются в документах.

Классификация по рассмотренному признаку - способу реализации модели - показана на рис. 1.2.

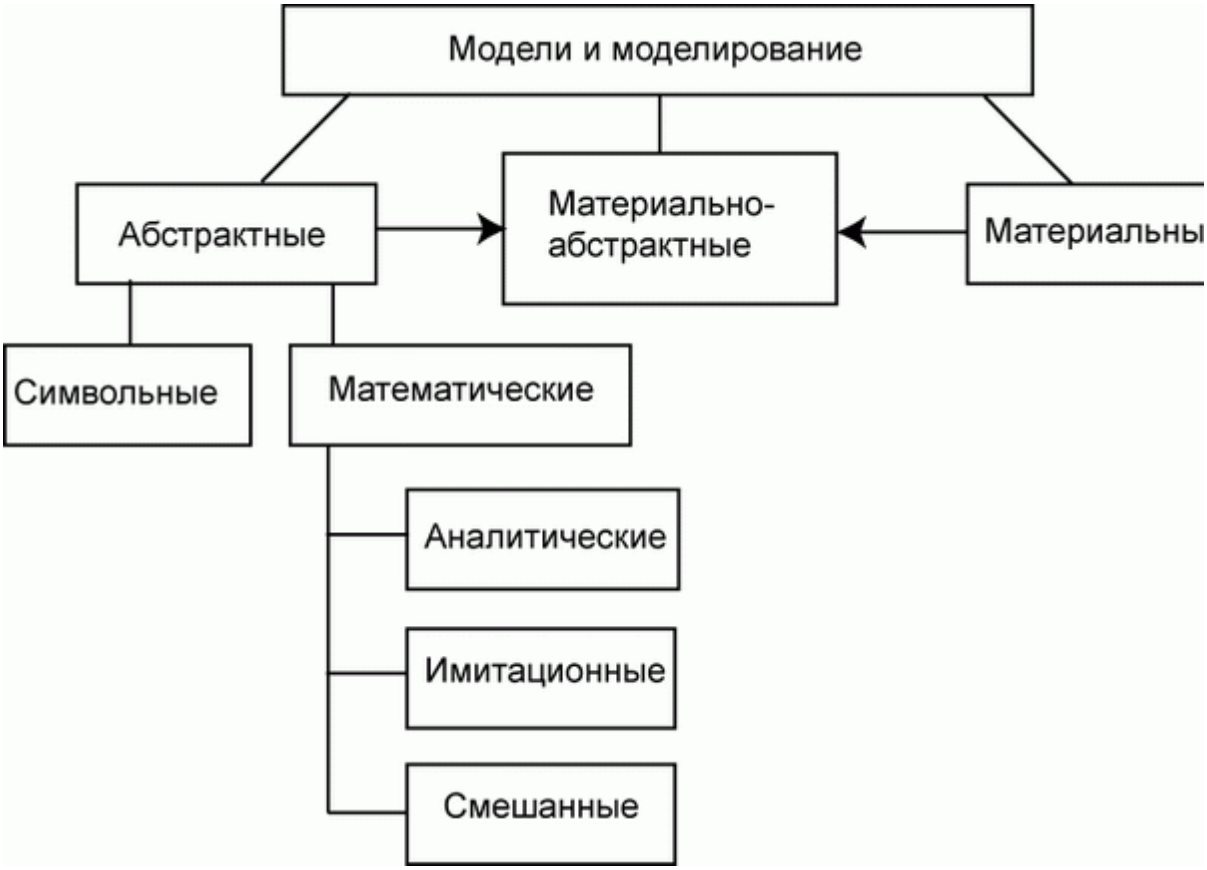

Illustration 1: Классификация по способу реализации

Этапы моделирования

Математическое моделирование как, впрочем, и любое другое, считается искусством и наукой. Известный специалист в области

имитационного моделирования Роберт Шеннон так назвал свою широко известную в научном и инженерном мире книгу: "Имитационное моделирование - искусство и наука". Поэтому в инженерной практике нет формализованной инструкции, как создавать модели. И, тем не менее, анализ приемов, которые используют разработчики моделей, позволяет усмотреть достаточно прозрачную этапность моделирования.

Первый этап: уяснение целей моделирования. Вообще-то это главный этап любой деятельности. Цель существенным образом определяет содержание остальных этапов моделирования. Заметим, что различие между простой системой и сложной порождается не столько их сущностью, но и целями, которые ставит исследователь.

Обычно целями моделирования являются:

прогноз поведения объекта при новых режимах, сочетаниях факторов и т. п.;

подбор сочетания и значений факторов, обеспечивающих оптимальное значение показателей эффективности процесса;

анализ чувствительности системы на изменение тех или иных факторов;

проверка различного рода гипотез о характеристиках случайных параметров исследуемого процесса;

определение функциональных связей между поведением ("реакцией") системы и влияющими факторами, что может способствовать прогнозу поведения или анализу чувствительности;

уяснение сущности, лучшее понимание объекта исследования, а также формирование первых навыков для эксплуатации моделируемой или действующей системы.

Второй этап: построение концептуальной модели. Концептуальная модель (от лат. conception) - модель на уровне определяющего замысла, который формируется при изучении моделируемого объекта. На этом этапе исследуется объект, устанавливаются необходимые упрощения и аппроксимации. Выявляются существенные аспекты, исключаются второстепенные. Устанавливаются единицы измерения и диапазоны изменения переменных модели. Если возможно, то концептуальная модель представляется в виде известных и хорошо разработанных систем: массового обслуживания, управления, авторегулирования, разного рода автоматов и т. д. Концептуальная модель полностью подводит итог изучению проектной документации или экспериментальному обследованию моделируемого объекта.

Результатом второго этапа является обобщенная схема модели, полностью подготовленная для математического описания - построения математической модели.

Третий этап: выбор языка программирования или моделирования, разработка алгоритма и программы модели. Модель может быть аналитической или имитационной, или их сочетанием. В случае аналитической модели исследователь должен владеть методами решения.

В истории математики (а это, впрочем, и есть история математического моделирования) есть много примеров тому, когда необходимость моделирования разного рода процессов приводила к новым открытиям. Например, необходимость моделирования движения привела к открытию и разработке дифференциального исчисления (Лейбниц и Ньютон) и соответствующих методов решения. Проблемы аналитического моделирования устойчивости кораблей привели академика Крылова А. Н. к созданию теории приближенных вычислений и аналоговой вычислительной машины.

Результатом третьего этапа моделирования является программа, составленная на наиболее удобном для моделирования и исследования языке - универсальном или специальном.

Четвертый этап: планирование эксперимента. Математическая модель является объектом эксперимента. Эксперимент должен быть в максимально возможной степени информативным, удовлетворять ограничениям, обеспечивать получение данных с необходимой точностью и достоверностью. Существует теория планирования эксперимента, нужные нам элементы этой теории мы изучим в соответствующем месте дисциплины.

Результат четвертого этапа - план эксперимента.

Пятый этап: выполнение эксперимента с моделью. Если модель аналитическая, то эксперимент сводится к выполнению расчетов при варьируемых исходных данных. При имитационном моделировании модель реализуется на ЭВМ с фиксацией и последующей обработкой получаемых данных. Эксперименты проводятся в соответствии с планом, который может быть включен в алгоритм модели. В современных системах моделирования такая возможность есть.

Шестой этап: обработка, анализ и интерпретация данных эксперимента. В соответствии с целью моделирования применяются разнообразные методы обработки: определение разного рода характеристик случайных величин и процессов, выполнение анализов - дисперсионного, регрессионного, факторного и др. Многие из этих методов входят в системы моделирования (GPSS World, AnyLogic и др.) и могут применяться автоматически. Не исключено, что в ходе анализа полученных результатов модель может быть уточнена, дополнена или даже полностью пересмотрена.

После анализа результатов моделирования осуществляется их интерпретация, то есть перевод результатов в термины предметной области.

Это необходимо, так как обычно специалист предметной области (тот, кому нужны результаты исследований) не обладает терминологией математики и моделирования и может выполнять свои задачи, оперируя лишь хорошо знакомыми ему понятиями.

На этом рассмотрение последовательности моделирования закончим, сделав весьма важный вывод о необходимости документирования результатов каждого этапа. Это необходимо в силу следующих причин.

Во-первых, моделирование процесс итерактивный, то есть с каждого этапа может осуществляться возврат на любой из предыдущих этапов для уточнения информации, необходимой на этом этапе, а документация может сохранить результаты, полученные на предыдущей итерации.

Во-вторых, в случае исследования сложной системы в нем участвуют большие коллективы разработчиков, причем различные этапы выполняются различными коллективами. Поэтому результаты, полученные на каждом этапе, должны быть переносимы на последующие этапы, то есть иметь унифицированную форму представления и понятное другим заинтересованным специалистам содержание.

В-третьих, результат каждого из этапов должен являться самоценным продуктом. Например, концептуальная модель может и не использоваться для дальнейшего преобразования в математическую модель, а являться описанием, хранящим информацию о системе, которое может использоваться как архив, в качестве средства обучения и т. д. [8].

Трёхмерная графика

Трёхмерная графика - раздел компьютерной графики, совокупность приемов и инструментов (как программных, так и аппаратных), предназначенных для изображения объёмных объектов. Больше всего

применяется для создания изображений на плоскости экрана или листа печатной продукции в архитектурной визуализации, кинематографе, телевидении, компьютерных играх, печатной продукции, а также в науке и промышленности.

Трёхмерное изображение на плоскости отличается от двумерного тем, что включает построение геометрической проекции трёхмерной модели сцены на плоскость (например, экран компьютера) с помощью специализированных программ. При этом модель может как соответствовать объектам из реального мира (автомобили, здания, ураган, астероид), так и быть полностью абстрактной (проекция четырёхмерного фрактала).

Для получения трёхмерного изображения на плоскости требуются следующие шаги:

1. моделирование - создание трёхмерной математической модели сцены и объектов в ней;

2. рендеринг (визуализация) - построение проекции в соответствии с выбранной физической моделью;

3. вывод полученного изображения на устройство вывода - дисплей или принтер.

Однако, в связи с попытками создания 3D-дисплеев и 3D-принтеров, трёхмерная графика не обязательно включает в себя проецирование на плоскость.

Компьютерное моделирование

Сцена (виртуальное пространство моделирования) включает в себя несколько категорий объектов:

1. Геометрия (построенная с помощью различных техник модель, например здание).

2. Материалы (информация о визуальных свойствах модели, например цвет стен и отражающая/преломляющая способность окон).

3. Источники света (настройки направления, мощности, спектра освещения).

4. Виртуальные камеры (выбор точки и угла построения проекции).

5. Силы и воздействия (настройки динамических искажений объектов, применяется в основном в анимации).

6. Дополнительные эффекты (объекты, имитирующие атмосферные явления: свет в тумане, облака, пламя и пр.).

Задача трёхмерного моделирования - описать эти объекты и разместить их в сцене с помощью геометрических преобразований в соответствии с требованиями к будущему изображению.

Рендеринг

На этом этапе математическая (векторная) пространственная модель превращается в плоскую (растровую) картинку. Если требуется создать фильм, то рендерится последовательность таких картинок -- кадров. Как структура данных, изображение на экране представлено матрицей точек, где каждая точка определена по крайней мере тремя числами: интенсивностью красного, синего и зелёного цвета. Таким образом рендеринг преобразует трёхмерную векторную структуру данных в плоскую матрицу пикселов. Этот шаг часто требует очень сложных вычислений, особенно если требуется создать иллюзию реальности. Самый простой вид рендеринга - это построить контуры моделей на экране компьютера с помощью проекции. Обычно этого

недостаточно и нужно создать иллюзию материалов, из которых изготовлены объекты, а также рассчитать искажения этих объектов за счёт прозрачных сред (например, жидкости в стакане). Существует несколько технологий рендеринга, часто комбинируемых вместе. Например:

1. Z-буфер (используется в OpenGL и DirectX 10).

2. Сканлайн (scanline) - он же Ray casting («бросание луча», упрощенный алгоритм обратной трассировки лучей) - расчёт цвета каждой точки картинки построением луча из точки зрения наблюдателя через воображаемое отверстие в экране на месте этого пиксела «в сцену» до пересечения с первой поверхностью. Цвет пиксела будет таким же, как цвет этой поверхности (иногда с учётом освещения и т. д.).

3. Трассировка лучей (рейтрейсинг, англ. raytracing) - то же, что и сканлайн, но цвет пиксела уточняется за счёт построения дополнительных лучей (отражённых, преломлённых и т. д.) от точки пересечения луча взгляда. Несмотря на название, применяется только обратная трассировка лучей (то есть как раз от наблюдателя к источнику света), прямая крайне неэффективна и потребляет слишком много ресурсов для получения качественной картинки.

4. Глобальное освещение (англ. global illumination, radiosity) -- расчёт взаимодействия поверхностей и сред в видимом спектре излучения с помощью интегральных уравнений.

Грань между алгоритмами трассировки лучей в настоящее время практически стёрлась. Так, в 3D Studio Max стандартный визуализатор называется Default scanline renderer, но он считает не только вклад диффузного, отражённого и собственного (цвета самосвечения) света, но и сглаженные тени. По этой причине, чаще понятие Raycasting относится к обратной трассировке лучей, а Raytracing - к прямой.

Вследствие большого объема однотипных вычислений рендеринг можно разбивать на потоки (распараллеливать). Поэтому для рендеринга весьма актуально использование многопроцессорных систем. В последнее время активно ведётся разработка систем рендеринга использующих GPU вместо CPU, и уже сегодня их эффективность для таких вычислений намного выше. К таким системам относятся:

1. Refractive Software Octane Render.
2. AAA studio FurryBall.
3. RandomControl ARION (гибридная).

Многие производители систем рендеринга для CPU также планируют ввести поддержку GPU (LuxRender, YafaRay, mental images iray).

Самые передовые достижения и идеи трёхмерной графики (и компьютерной графики вообще) докладываются и обсуждаются на ежегодном симпозиуме SIGGRAPH, традиционно проводимом в США.

Программное обеспечение

Программные пакеты, позволяющие создавать трёхмерную графику, то есть моделировать объекты виртуальной реальности и создавать на основе этих моделей изображения, очень разнообразны. Последние годы устойчивыми лидерами в этой области являются коммерческие продукты: такие как 3D Studio Max, Maya, Lightwave 3D, Softimage, Sidefx Houdini, Maxon Cinema 4D и сравнительно новые Rhinoceros 3D, Nevercenter Silo или ZBrush. Кроме того, существуют и открытые продукты, распространяемые свободно, например, пакет Blender (позволяет создавать 3D модели, с последующим рендерингом (компьютерной визуализацией)), K-3D и Wings3D.

Для чего задумали трехмерную графику? Прежде всего, она создана для более реального изображения предметов, для более яркого представления реального мира, для изображения предметов, объектов, которые максимально будут соответствовать реальным.

Создание трехмерного изображения (естественно с помощью специальных программ) включает в себя основных два этапа: моделирование и непосредственно визуализацию. На этапе моделирования происходит проектирование модели (основная цель моделирования, есть то, что проектируются объекты и в дальнейшем редактируется с помощью геометрических преобразований, для создания более реальной модели с определенными требованиями), а на последующем этапе выполняется построение проекции, и в дальнейшем оживление созданной модели с помощью разных методов и приемов. Трехмерная графика и анимация занимает сейчас важную нишу, и в дальнейшем планирует свое все большее развитие и внедрение во многих областях [9].

Часть 2 Практическое применение

2.1 Визуализация загружаемой готовой модели

Архитектура html страницы

Для начала рассмотрим архитектуру HTML страницы [10].

Итак, приступим к формированию страницы. Сначала следует тег <! doctype>. В спецификации HTML5 тег <!doctype> был упрощен: вам достаточно запомнить его атрибут —html. Это не только облегчает ввод этого тега, но и улучшает его защиту от ошибок. Обратите внимание, что атрибут имеет вид html, а не html5. Вне зависимости от количества версий HTML, тег <!doctype> всегда сможет иметь атрибут html.

Тег <html> содержит все остальные HTML-теги за исключением тега <! doctype>. Каждый из остальных тегов должен быть размещен между тегом <html> и тегом </html>.

Пример тега <!doctype>:

```
<!doctype html>
<html lang="en">
```

После указания атрибута html и английского языка следует тег <head>, который может содержать скрипты, информацию о поддерживаемых браузерах, ссылки на таблицу стилей, метаинформацию и другие инициализационные функции. В разделе head можно использовать следующие теги.

1. <base>;
2. <link>;
3. <meta>;
4. <script>;

5. <style>;
6. <title>.

Тег <title> содержит фактический заголовок документа и является обязательным тегом раздела <head>. Этот заголовок можно увидеть в верхней части браузера при просмотре страницы. Тег <link> указывает CSS3-таблицу стилей, которая будет использоваться для отображения данной HTML5-страницы. Эта таблица стилей имеет имя main-stylesheet.css

Пример тега <head>:

```
<head>
     <title>HTML5 Fundamentals Example</title>
     <link rel="stylesheet" href="main-stylesheet.css"  />
</head>
```

Затем мы используем тег <body>, за которым следуют теги <header> и <hgroup>, описанные ранее. Область <h1> в данном примере содержит название вымышленной компании (Acme United), а область <h2> содержит подзаголовок «A Simple HTML5 Example» (простой пример на HTML5).

Пример тегов <body> и <header>:

```
<body >
     <header>
          <hgroup>
               <h1>Acme United</h1>
               <h2>A Simple HTML5 Example</h2>
          </hgroup>
     </header>
```

Структура страницы и используемые библиотеки

```
<!DOCTYPE html>
<html lang="en" >
    <head>
        <meta charset="utf-8" />
```

```
            <meta name="author" content="Script Tutorials" />
            <title>Hard Disk</title>
                  <meta  name="viewport"  content="width=device-width,
initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />

        </head>
        <body>
            <script src="js/three.min.js"></script>
            <script src="js/OBJLoader.js"></script>
            <script src="js/THREEx.WindowResize.js"></script>
            <script src="js/OrbitControls.js"></script>
            <script src="js/stats.min.js"></script>
            <script src="js/script1.js"></script>
        </body>
    </html>
```

Как мы видим, всего несколько строчек html кода и получаем результат, показаный на рис. 2.

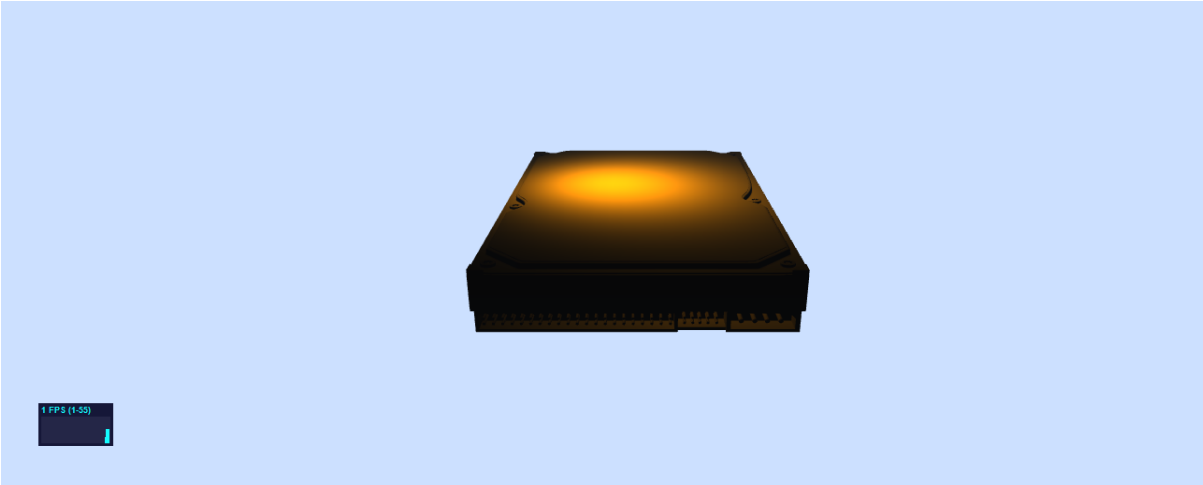

Illustration 2: Модель винчестера

Теперь мы можем с помощью этих библиотек скомпоновать решение следующих задач:

1. Загрузка произвольного трёхмерного объекта.

2. Рисование объекта с аппаратным ускорением на странице браузера.
3. Взаимодействие с объектом.

Множество замечательных примеров вы можете увидеть по этой ссылке [12].

Теперь рассмотрим назначение каждой из использованных библиотек:

1. three.min.js - кроссбраузерная библиотека JavaScript для технологии WebGL [13];

   .

2. OBJLoader.js - библиотека обеспечивающая возможность загрузки объектов;
3. THREEx.WindowResize.js - для обновления средств визуализации и камеры при изменении размеров окна;
4. OrbitControls.js — управление;
5. stats.min.js - этот класс обеспечивает простое информационное окно, которое поможет вам отслеживать эффективность кода;
6. script1.js - пользовательская библиотека, содержимое которой собственно и есть то, что в результате мы увидим на экране браузера (смотреть приложение А) [14].

В результате выполнения всех выше описанных действий, мы получаем сцену, освещение, камеру и загруженный трёхмерный объект.

Для управления камерой (вращения, масштабирования, перемещения) нам достаточно только менять координаты и углы в точке 10.

2.2 Руководство по анимации модели

Для примера создадим анимированную 3D-сцену, на которой мы увидим летящий над морем самолёт с помощью Three.js - JavaScript библиотеки, упрощающей работу с WebGL.

HTML и CSS

Первое, что нужно сделать - добавить между открывающим и закрывающим тегом body блок, в котором мы будем рендерить нашу WebGL-сцену:

```
<div id="world" class="world"></div>
```

Оформите блок так, чтобы он занимал все окно браузера. Фон нашей страницы сделаем ввиде градиента, который заменит собой небо.

```
world {
    position: absolute;
    width:100%;
    height:100%;
    overflow: hidden;
    background:linear-gradient(#e4e0ba, #f7d9aa);
}
```

Теперь нам нужно подключить библиотеку Three.js к веб-странице. Для этого сразу перед закрывающим тегом body напишите следующее:

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

Мы закончили с HTML и CSS и настал час JavaScript!

Пишем код на JavaScript

Three.js очень прост в использовании, если вы хоть немного знаете JavaScript. Разобьем наш код на несколько частей, чтобы проще его было понять.

Цветовая палитра

Для начала определимся с цветами в нашей будущей сцене. Эти цвета будут постоянно использоваться в нашей игре на Three.js, поэтому вынесем HEX-значения этих цветов в объект Colors:

```
var Colors = {
    red: 0xf25346,
    white: 0xd8d0d1,
    brown: 0x59332e,
    pink: 0xF5986E,
    brownDark: 0x23190f,
    blue: 0x68c3c0,
};
```

Структура кода

Несмотря на то, что нам предстоит написать большое количество JavaScript-кода, его структура довольно проста. Все основные функции нам необходимо запустить при инициализации объекта window с помощью функции init():

```
window.addEventListener('load', init, false);

function init() {
    // настройка сцены, камеры и рендера
    createScene();
```

```
    // добавление освещения сцены
    createLights();

    // добавление объектов на сцену
    createPlane();
    createSea();
    createSky();

    // запускаем цикл для обновления объектов
    // на сцене и покадровый перерасчет сцены
    loop();
}
```

Настройка 3D-сцены

Для создания нашей 3D игры на Three.js нам понадобятся:

- Сцена (некоторая область, в которую мы поместим наши 3D-объекты для последующего рендера).
- Камера (в нашей игре мы будем использовать камеру с перспективой).
- Рендеринг, который будет отображать нашу сцену с ипользованием WebGL.
- Объекты для рендера (самолет, море и облака).
- Источники света (мы будем использовать полушарный свет для атмосферы и направленный свет для теней).

Сцену, камеру и рендеринг мы запустим в функции createScene():

```
var scene, camera, fieldOfView, aspectRatio, nearPlane,
    farPlane, HEIGHT, WIDTH, renderer, container;

function createScene() {
```

```
// Получаем ширину и высоту сцены и используем
// их для установки размера и пропорций камеры,
// а также для размера финального рендера
HEIGHT = window.innerHeight;
WIDTH = window.innerWidth;

// Создание сцены
scene = new THREE.Scene();

// Добавим эффект тумана на нашу сцену (его
// цвет возьмем из наших таблиц стилей, а не
// из объекта Colors
scene.fog = new THREE.Fog(0xf7d9aa, 100, 950);

// Создание камеры
aspectRatio = WIDTH / HEIGHT;
fieldOfView = 60;
nearPlane = 1;
farPlane = 10000;
camera = new THREE.PerspectiveCamera(
    fieldOfView,
    aspectRatio,
    nearPlane,
    farPlane
);

// Задаем позицию камеры в 3D пространстве
camera.position.x = 0;
camera.position.z = 200;
camera.position.y = 100;

// Создаем рендер
renderer = new THREE.WebGLRenderer({
// Разрешаем прозрачноть для сцены чтобы
// увидеть наше градиентное небо
alpha: true,
```

```
// Активируем сглаживание. Это может снизить
// производительность. К счастью, наш проект
// состоит из низкополигональных моделей
antialias: true
});

// Задаем размер рендера (в нашем случае
// он равен размеру сцены)
renderer.setSize(WIDTH, HEIGHT);

// Включаем рендер теней
renderer.shadowMap.enabled = true;

// Обратимся к DOM-элементу с идентификатором world,
// для рендера в него сцены.
container = document.getElementById('world');
container.appendChild(renderer.domElement);

// Отслеживаем изменения размера экрана.
// В случае изменения запускаем функцию
// handleWindowResize(), чтобы обновить камеру и рендер.
window.addEventListener('resize', handleWindowResize, false);
}
```

Так как размер окна браузера может меняться пользователем, необходимо обновлять сцену, рендер и камеру при изменении размера окна:

```
function handleWindowResize() {
  // обновим высоту, ширину камеры и рендера
  HEIGHT = window.innerHeight;
  WIDTH = window.innerWidth;
  renderer.setSize(WIDTH, HEIGHT);
  camera.aspect = WIDTH / HEIGHT;
  camera.updateProjectionMatrix();
```

```
}
```

Освещение сцены

Освещение задает настроение всей сцене и должно применяться осмотрительно. Для начала мы добьемся того, чтобы наши объекты было видно на сцене.

```
var hemisphereLight, shadowLight;

function createLights() {
  // Полушарный свет - это градиентный свет
  // Первый параметр - цвет неба, второй - цвет земли,
  // а третий - интенсивность света
                        hemisphereLight          =          new
THREE.HemisphereLight(0xaaaaaa,0x000000,.9)

   // Направленный свет светит в определенном направлении из
точки,
  // (как солнце). Это значит что продуцируемые лучи параллельны.
  shadowLight = new THREE.DirectionalLight(0xffffff, .9);

  // Устанавливаем направление света
  shadowLight.position.set(150, 350, 350);

  // Разрешаем отбрасывание теней

  shadowLight.castShadow = true;

  // Определяем видимую область теней
  shadowLight.shadow.camera.left = -400;
  shadowLight.shadow.camera.right = 400;
  shadowLight.shadow.camera.top = 400;
  shadowLight.shadow.camera.bottom = -400;
  shadowLight.shadow.camera.near = 1;
  shadowLight.shadow.camera.far = 1000;
```

```
    // Задаем разрешение теней. Учтите, что чем
    // оно больше, тем ниже производительность.
    shadowLight.shadow.mapSize.width = 2048;
    shadowLight.shadow.mapSize.height = 2048;

    // Добавляем освещение на сцену
    scene.add(hemisphereLight);
    scene.add(shadowLight);
}
```

Как видите, для создания света требуется задать приличное число параметров. Поэкспериментируйте с цветом освещения, интенсивностью, количеством источников света и, возможно, вы сделаете сцену еще лучше.

Создание 3D-объектов с помощью Three.js

Если вы имеете навыки 3D-моделирования, то вы можете создавать свои объекты и импортировать их в проект на Three.js. Мы же с вами поступим следующим образом: создадим наши объекты используя примитивы, доступные в Three.js. Библиотека Three.js имеет немалое количество готовых к использованию примитивов, например, куб, сфера, цилиндр и др. Комбинируя эти примитивы и создадим наши объекты.

Создание моря в Three.js

Самым простым объектом в нашей игре является море. С него то мы и начнем. Чтобы не усложнять, мы представим море как обычный голубой цилиндр в нижней части экрана. Пока этого достаточно, а позже мы сделаем море более реалистичным.

```
// Определяем JS-объект нашего моря
Sea = function(){

  // Создаем геометрию цилиндра со следующими параметрами:
  // верхний радиус, нижний радиус, высота, количество
  // сегментов по окружности и по вертикали
  var geom = new THREE.CylinderGeometry(600,600,800,40,10);

  // Поворачиваем наш объект по оси x
  geom.applyMatrix(new THREE.Matrix4().makeRotationX(-Math.PI/2));

  // создаем материал для нашего объекта
  var mat = new THREE.MeshPhongMaterial({
    color:Colors.blue,
    transparent:true,
    opacity:.6,
    shading:THREE.FlatShading,
  });

  // Для создания объекта в Three.js нужно создать меш, который
  // представляет собой совокупность созданных ранее геометрии и материала
  this.mesh = new THREE.Mesh(geom, mat);

  // Разрешаем морю отбрасывать тени
  this.mesh.receiveShadow = true;
}

// Инициализируем море и добавляем его на сцену
var sea;

function createSea(){
  sea = new Sea();

  // Подвинем объект в нижнюю часть нашей сцены
  sea.mesh.position.y = -600;
```

```
    // Добавляем финальный меш на сцену
    scene.add(sea.mesh);
  }
```

Повторим еще рав в каком порядке необходимо создавать 3D-объект в Three.js:

- Создаем геометрию.
- Создаем материал.
- Объединяем геометрию и материал в меш.
- Добавляем меш на сцену.

Выполняя эти действия, мы можем создавать различные примитивы. Комбинируя их, мы сможем получать более сложные формы. Давайте попробуем.

Объединение кубов в облака

С облаками нам придется повозиться немного больше, поскольку они будут состоять из нескольких кубов, случайным образом расположеных в одном объекте.

```
  Cloud = function(){
    // Создаем пустой контейнер, содержащий
    // составные части облаков (кубики)
    this.mesh = new THREE.Object3D();

    // Создаем геометрию кубика. Этот кубик мы
```

```
    // будем дублировать для создания облаков
    var geom = new THREE.BoxGeometry(20,20,20);

    // Создадим простой материал для нашего кубика
    var mat = new THREE.MeshPhongMaterial({
      color:Colors.white,
    });

    // Продублируем кубик произвольное число раз
    var nBlocs = 3+Math.floor(Math.random()*3);
    // Применим цикл для каждого куба и поместим их в наш меш
    for (var i=0; i<nBlocs; i++ ){
      // Создадим меш путем клонирования кубика циклом
      var m = new THREE.Mesh(geom, mat);

      // зададим случайную позицию и ротацию каждому кубику
      m.position.x = i*15;
      m.position.y = Math.random()*10;
      m.position.z = Math.random()*10;
      m.rotation.z = Math.random()*Math.PI*2;
      m.rotation.y = Math.random()*Math.PI*2;

      // Зададим произвольный размер кубикам
      var s = .1 + Math.random()*.9;
      m.scale.set(s,s,s);

      // Пусть кубики отбрасывают и преломляют тени
      m.castShadow = true;
      m.receiveShadow = true;

      // Поместим кубик в контейнер, созданный в начале этой функции
      this.mesh.add(m);
    }
  }
```

Теперь нам ничто не мешает создать небо из нескольких облаков, расположенных случайным образом на оси Z:

```
// Создадим объект неба
Sky = function(){
  // Создадим пустой контейнер
  this.mesh = new THREE.Object3D();

  // Количество облаков на небе будет равно 20
  this.nClouds = 20;

  // Чтобы распределить облака равномерно рассчитаем
  // величину угла для каждого следующего облака
  var stepAngle = Math.PI*2 / this.nClouds;
```

```
  // Создадим облака для неба с помощью цикла
  for(var i=0; i<this.nClouds; i++){
    var c = new Cloud();

    // Зададим угол поворота и расстояние
    // между центром оси и облаком
    var a = stepAngle*i;
    var h = 750 + Math.random()*200;

    // Конвертируем полярные координаты (угол,
       // расстояние) в Декартовые координаты (x, y)
    c.mesh.position.y = Math.sin(a)*h;
    c.mesh.position.x = Math.cos(a)*h;

    // Поворачиваем облако относительно его позиции
    c.mesh.rotation.z = a + Math.PI/2;

    // Для большего реализма разместим облака
    // на случайной глубине на нашей сцене
    c.mesh.position.z = -400-Math.random()*400;

    // Установим случайный размер для каждого облака
```

```
      var s = 1+Math.random()*2;
      c.mesh.scale.set(s,s,s);

      // Добавим каждое облако в наш контейнер
      this.mesh.add(c.mesh);
    }
  }

  // Инициализируем небо и размещаем его внизу экрана
  var sky;

  function createSky(){
    sky = new Sky();
    sky.mesh.position.y = -600;
    scene.add(sky.mesh);
  }
```

Создание самолета

Фигура самолета более сложная, чем облака и море, но мы уже изучили все необходимое, чтобы это реализовать! Мы по прежнему будем комбинировать и инкапсулировать наши объекты.

Создание игры на Three.js

```
  var AirPlane = function() {

    this.mesh = new THREE.Object3D();

    // Создаем кабину
    var geomCockpit = new THREE.BoxGeometry(60,50,50,1,1,1);
    var matCockpit = new THREE.MeshPhongMaterial({color:Colors.red,
shading:THREE.FlatShading});
    var cockpit = new THREE.Mesh(geomCockpit, matCockpit);
```

```
    cockpit.castShadow = true;
    cockpit.receiveShadow = true;
    this.mesh.add(cockpit);
```

```
    // Создаем двигатель
    var geomEngine = new THREE.BoxGeometry(20,50,50,1,1,1);
    var matEngine = new THREE.MeshPhongMaterial({color:Colors.white,
shading:THREE.FlatShading});
    var engine = new THREE.Mesh(geomEngine, matEngine);
    engine.position.x = 40;
    engine.castShadow = true;
    engine.receiveShadow = true;
    this.mesh.add(engine);
```

```
    // Создаем хвост
    var geomTailPlane = new THREE.BoxGeometry(15,20,5,1,1,1);
    var matTailPlane = new
THREE.MeshPhongMaterial({color:Colors.red,
shading:THREE.FlatShading});
    var tailPlane = new THREE.Mesh(geomTailPlane, matTailPlane);
    tailPlane.position.set(-35,25,0);
    tailPlane.castShadow = true;
    tailPlane.receiveShadow = true;
    this.mesh.add(tailPlane);

    // Создаем крыло
    var geomSideWing = new THREE.BoxGeometry(40,8,150,1,1,1);
    var matSideWing = new THREE.MeshPhongMaterial({color:Colors.red,
shading:THREE.FlatShading});
    var sideWing = new THREE.Mesh(geomSideWing, matSideWing);
```

```
    sideWing.castShadow = true;
    sideWing.receiveShadow = true;
    this.mesh.add(sideWing);
```

```
    // Создаем пропеллер
    var geomPropeller = new THREE.BoxGeometry(20,10,10,1,1,1);
    var matPropeller = new
THREE.MeshPhongMaterial({color:Colors.brown,
shading:THREE.FlatShading});
    this.propeller = new THREE.Mesh(geomPropeller, matPropeller);
    this.propeller.castShadow = true;
    this.propeller.receiveShadow = true;

    // Создаем лопасть
    var geomBlade = new THREE.BoxGeometry(1,100,20,1,1,1);
    var matBlade = new
THREE.MeshPhongMaterial({color:Colors.brownDark,
shading:THREE.FlatShading});

    var blade = new THREE.Mesh(geomBlade, matBlade);
    blade.position.set(8,0,0);
    blade.castShadow = true;
    blade.receiveShadow = true;
    this.propeller.add(blade);
    this.propeller.position.set(50,0,0);
    this.mesh.add(this.propeller);
  };
```

Этот самолет выглядит весьма схематично, но ничего страшного. В дальнейшем мы поработаем над его детализацией, а пока что инициализируем его и добавим на сцену:

```
  var airplane;

  function createPlane(){
    airplane = new AirPlane();
    airplane.mesh.scale.set(.25,.25,.25);
    airplane.mesh.position.y = 100;
```

```
    scene.add(airplane.mesh);
  }
```

Рендеринг сцены в Three.js

Мы создали несколько объектов и добавили их на сцену. Но, если вы запустите нашу игру, то ничего не увидите. Это все потому, что мы не просчитали рендер нашей сцены. К счастью дело поправимо, причем всего лишь одной строчкой кода:

renderer.render(scene, camera);

Анимация объектов в Three.js

Давайте оживим нашу игру путем перемещения облаков и моря, а также вращая пропеллер самолета. Для этого нам понадобится бесконечный цикл:

```
  function loop(){
    // Вращаем пропеллер, море и небо
    airplane.propeller.rotation.x += 0.3;
    sea.mesh.rotation.z += .005;
    sky.mesh.rotation.z += .01;

    // рендерим сцену
    renderer.render(scene, camera);

    // снова вызываем фунцию loop
    requestAnimationFrame(loop);
  }
```

Как видите, мы перенесли вызов рендера в функцию loop(). Это потому, что при изменении того или иного объекта нам необходимо пересчитать рендер сцены и увидеть эти самые изменения в объектах.

Добавляем реакцию на перемещение мыши

Сейчас наш самолет расположен в центре сцены. Давайте заставим его следовать за курсором мыши. Для этого, когда страница загрузилась в браузере, необходимо повесить обработчик события на document и отслеживать перемещение курсора миши. Для этого немного изменим функцию init():

```
function init(event){
  createScene();
  createLights();
  createPlane();
  createSea();
  createSky();

  //добавляем слушатель перемещения курсора мыши
  document.addEventListener('mousemove', handleMouseMove, false);

  loop();
}
```

Кроме того, нам нужно будет создать функцию, обрабатывающую передвижение курсора мыши:

```
var mousePos={x:0, y:0};

// Обрабатываем наше событие
```

```
function handleMouseMove(event) {
  // Конвертируем полученные значение положения мыши
  // в нормализолованное значение между -1 и 1;
  // вот формула для горизонтальной оси:
  var tx = -1 + (event.clientX / WIDTH)*2;

  // Для вертикальной оси нам необходимо инвертировать
  // нашу формулу, так как в 2D ось Y идет в
  // противоположном направлении, в отличии от оси Y в 3D
  var ty = 1 - (event.clientY / HEIGHT)*2;
  mousePos = {x:tx, y:ty};
}
```

Имеем нормализованное значение положения мыши по осям X и Y, мы можем перемещать наш самолет в соответствии с этими значениями. Для этого слегка модифицируем функцию loop() и добавим новую функцию для изменения позиции самолета:

```
function loop(){
  sea.mesh.rotation.z += .005;
  sky.mesh.rotation.z += .01;

  // Обновляем наш самолет в каждом кадре
  updatePlane();

  renderer.render(scene, camera);
  requestAnimationFrame(loop);
}

function updatePlane(){

  // Давайте перемещать наш самолет в промежутке
  // от -100 до 100 по горизонтали и от 25 до 175
```

```
    // по вертикали, в зависимости от позиции курсора мыши,
    // у которого разброс значений между -1 и 1 в обоих
    // направлениях. Чтобы добиться этого, мы используем
    // функцию нормализации (normalize)

    var targetX = normalize(mousePos.x, -1, 1, -100, 100);
    var targetY = normalize(mousePos.y, -1, 1, 25, 175);

    // Обновляем позицию нашего самолета
    airplane.mesh.position.y = targetY;
```

```
    airplane.mesh.position.x = targetX;
```

```
    airplane.propeller.rotation.x += 0.3;
  }

  function normalize(v,vmin,vmax,tmin, tmax){
    var nv = Math.max(Math.min(v,vmax), vmin);
    var dv = vmax-vmin;
    var pc = (nv-vmin)/dv;
    var dt = tmax-tmin;
    var tv = tmin + (pc*dt);
    return tv;

  }
```

Мы сделали чтобы самолет реагировал на движения мыши!

Ранее мы создали очень простой самолетик и теперь знаем как создавать объекты и объединять их. Теперь нам предстоит узнать как изменять примитивы, чтобы подогнать их под наши потребности.

Куб, например, может быть изменен путем перемещения его вершин. В нашем случае мы хотим сделать его более похожим на кабину. Давайте

взглянем на кабину летчика и подумаем как можно сделать ее более узкой сзади.

Изменение геометрии в Three.js

```
  // Кабина летчика
  var geomCockpit = new THREE.BoxGeometry(80,50,50,1,1,1);
  var matCockpit = new THREE.MeshPhongMaterial({color:Colors.red,
shading:THREE.FlatShading});

  // мы можем получить доступ к определенной вершине формы через
  // массив vertices, а затем изменить ее x, y и z свойства:
  geomCockpit.vertices[4].y-=10;
  geomCockpit.vertices[4].z+=20;
  geomCockpit.vertices[5].y-=10;
  geomCockpit.vertices[5].z-=20;
  geomCockpit.vertices[6].y+=30;
  geomCockpit.vertices[6].z+=20;
  geomCockpit.vertices[7].y+=30;
  geomCockpit.vertices[7].z-=20;

  var cockpit = new THREE.Mesh(geomCockpit, matCockpit);
  cockpit.castShadow = true;
  cockpit.receiveShadow = true;
  this.mesh.add(cockpit);
```

Это отличный пример того как можно манипулировать формой, чтобы настроить ее для своих нужд. Как можно видеть, в этом нет ничего сложного. Попробуйте поиграть со значениями и получить свою собственную версию самолета.

Кто летит самолете?

Добавление пилота в наш самолет так же просто, как добавление пары ящиков. Но нам нужен не просто пилот, нам нужен классный пилот с развивающимися от ветра волосами! Поскольку мы работаем на низкополигональной сцене, - это не такая уж и сложная задача.

Анимирование волос в Three.js

Давайте посмотрим на код:

```
var Pilot = function(){
    this.mesh = new THREE.Object3D();
    this.mesh.name = "pilot";

    // angleHairs - свойство, используемое для анимации волос
    this.angleHairs=0;

    // Тело пилота
    var bodyGeom = new THREE.BoxGeometry(15,15,15);
    var bodyMat = new THREE.MeshPhongMaterial(
      {color:Colors.brown, shading:THREE.FlatShading}
    );
    var body = new THREE.Mesh(bodyGeom, bodyMat);
    body.position.set(2,-12,0);
    this.mesh.add(body);

    // Лицо пилота
    var faceGeom = new THREE.BoxGeometry(10,10,10);
    var faceMat = new
THREE.MeshLambertMaterial({color:Colors.pink});
    var face = new THREE.Mesh(faceGeom, faceMat);
    this.mesh.add(face);

    // Элемент волос
    var hairGeom = new THREE.BoxGeometry(4,4,4);
```

```
      var hairMat = new
THREE.MeshLambertMaterial({color:Colors.brown});
      var hair = new THREE.Mesh(hairGeom, hairMat);
      // Присвоить форму волос к нижней границе,
      // которая облегчит задачу с подбором размеров
      hair.geometry.applyMatrix(new
THREE.Matrix4().makeTranslation(0,2,0));

      // создать контейнер для волос
      var hairs = new THREE.Object3D();

      // создать контейнер для волос в верхней части
      // головы (те из них, которые будут анимированы)
      this.hairsTop = new THREE.Object3D();

      // создать волосы в верхней части головы
      // и расположите их на сетке 3 x 4
      for (var i=0; i<12; i++){
          var h = hair.clone();
          var col = i%3;
          var row = Math.floor(i/3);
          var startPosZ = -4;
          var startPosX = -4;
          h.position.set(startPosX + row*4, 0, startPosZ + col*4);
          this.hairsTop.add(h);
      }
      hairs.add(this.hairsTop);

      // создать волосы на лице (борода)
      var hairSideGeom = new THREE.BoxGeometry(12,4,2);
      hairSideGeom.applyMatrix(new THREE.Matrix4().makeTranslation(-
6,0,0));
      var hairSideR = new THREE.Mesh(hairSideGeom, hairMat);
      var hairSideL = hairSideR.clone();
      hairSideR.position.set(8,-2,6);
      hairSideL.position.set(8,-2,-6);
```

```
      hairs.add(hairSideR);
      hairs.add(hairSideL);

      // создать волосы на задней части головы
      var hairBackGeom = new THREE.BoxGeometry(2,8,10);
      var hairBack = new THREE.Mesh(hairBackGeom, hairMat);
      hairBack.position.set(-1,-4,0)
      hairs.add(hairBack);
      hairs.position.set(-5,5,0);

      this.mesh.add(hairs);

      var glassGeom = new THREE.BoxGeometry(5,5,5);
      var glassMat = new
THREE.MeshLambertMaterial({color:Colors.brown});
      var glassR = new THREE.Mesh(glassGeom,glassMat);
      glassR.position.set(6,0,3);
      var glassL = glassR.clone();
      glassL.position.z = -glassR.position.z

      var glassAGeom = new THREE.BoxGeometry(11,1,11);
      var glassA = new THREE.Mesh(glassAGeom, glassMat);
      this.mesh.add(glassR);
      this.mesh.add(glassL);
      this.mesh.add(glassA);

      var earGeom = new THREE.BoxGeometry(2,3,2);
      var earL = new THREE.Mesh(earGeom,faceMat);
      earL.position.set(0,0,-6);
      var earR = earL.clone();
      earR.position.set(0,0,6);
      this.mesh.add(earL);
      this.mesh.add(earR);
  }
  // шевеление волос
  Pilot.prototype.updateHairs = function(){
```

```
    // получаем волосы
    var hairs = this.hairsTop.children;

    // обновляем их в соответствии с углом angleHairs
    var l = hairs.length;
    for (var i=0; i<l; i++){
        var h = hairs[i];
        // каждый элемент волос будет масштабироваться
        // с помощью цикла от 75% до 100% от исходного размера
        h.scale.y = .75 + Math.cos(this.angleHairs+i/3)*.25;
    }
    // увеличение угла для следующего кадра
    this.angleHairs += 0.16;
}
```

Анимируем волосы в Three.js

Теперь, чтобы сделать движение волос, добавьте эту строку в функцию loop:

```
airplane.pilot.updateHairs();
```

Создаем волны в Three.js

Вы, наверное, заметили, что море не очень похоже на море, а скорее на поверхность, раздавленую катком. Она нуждается в нескольких волнах. Это может быть сделано путем объединения двух методов, которые мы использовали ранее:

Манипулирование геометрией вершины, как мы это делали с кабиной самолета.

Применение циклического движения к каждой вершине, как мы делали, чтобы переместить волосы пилота.

Чтобы создать волны, мы будем вращать каждую отдельную вершину цилиндра вокруг их изначальных позиций со случайной скоростью и дистанцией (радиусом вращения). И нам также необходимо использовать здесь некоторые тригонометрические функции.

Анимируем воду в Three.js

Внесем некоторые изменения в наше море:

```
  Sea = function(){
      var geom = new THREE.CylinderGeometry(600,600,800,40,10);
      geom.applyMatrix(new THREE.Matrix4().makeRotationX(-
Math.PI/2));

      // важно: путем слияния вершин мы обеспечиваем непрерывность
волн
      geom.mergeVertices();

      // получаем вершины
      var l = geom.vertices.length;

      // создать массив для хранения новых данных, связанных с
каждой вершиной
      this.waves = [];

      for (var i=0; i<l; i++){
          // получаем каждую вершину
          var v = geom.vertices[i];

          // сохраняем некоторые данные, связанные с нею
          this.waves.push({
              y:v.y,
              x:v.x,
              z:v.z,
              // случайный угол
```

```
            ang:Math.random()*Math.PI*2,
            // случайное расстояние
            amp:5 + Math.random()*15,
            // случайная скорость между 0,016 и 0,048 радиан/кадр
            speed:0.016 + Math.random()*0.032
        });
    };
    var mat = new THREE.MeshPhongMaterial({
        color:Colors.blue,
        transparent:true,
        opacity:.8,
        shading:THREE.FlatShading,
    });

    this.mesh = new THREE.Mesh(geom, mat);
    this.mesh.receiveShadow = true;

}
```

Теперь мы создадим функцию, которая будет вызываться в каждом кадре, чтобы обновить положение вершин для имитации волн.

```
Sea.prototype.moveWaves = function (){

    // получаем вершины
    var verts = this.mesh.geometry.vertices;
    var l = verts.length;

    for (var i=0; i<l; i++){
        var v = verts[i];

        // получить данные, связанные с ними
        var vprops = this.waves[i];

        // обновить положение вершины
        v.x = vprops.x + Math.cos(vprops.ang)*vprops.amp;
```

```
            v.y = vprops.y + Math.sin(vprops.ang)*vprops.amp;

            // увеличение угла для следующего кадра
            vprops.ang += vprops.speed;
        }

        // Говорим визуализатору, что геометрия моря изменилась.
        // На самом деле, для того, чтобы поддерживать оптимальный
        // уровень производительности, three.js кэширует геометрию и
        // игнорирует любые изменения, если мы не добавим эту строку
        this.mesh.geometry.verticesNeedUpdate=true;

        sea.mesh.rotation.z += .005;
    }
```

Волны в Three.js, сделанные из цилиндра

Так же, как мы делали для волос пилота, мы добавим эту строку в функцию loop:

```
sea.moveWaves();
```

Теперь мы наслаждаемся волнами!

Доработка освещения сцены в Three.js

Мы уже создали некоторое освещение. Но мы хотели бы сделать сцену повеселее и сделать тени мягче. Для достижения этой цели мы будем использовать рассеянный свет. В функции createLights мы добавим следующие строки:

```
// AmbientLight изменяет глобальный цвет сцены
// и делает тени более мягкими
ambientLight = new THREE.AmbientLight(0xdc8874, .5);
scene.add(ambientLight);
```

Более мягкий полет

Наш маленький самолет уже следует за движениями мыши. Но полет при этом, к сожалению, не выглядит реалистичным. Когда самолет меняет свою высоту, было бы неплохо, чтобы он менял свое положение и ориентацию более плавно.

Самый простой способ сделать то, что мы задумали - заставить самолет двигаться к цели, добавив часть расстояния, которое отделяет его от этой цели в каждом кадре. В основном, код будет выглядеть следующим образом:

```
currentPosition += (finalPosition - currentPosition)*fraction;
```

Для большей реалистичности, вращение самолета также можно изменять в зависимости от направления его движения. Если самолет взлетает слишком быстро, он должен быстро вращаться против часовой стрелки. Если он медленно опускается, то должен медленно вращаться по часовой. Для достижения точности мы можем просто присвоить пропорциональное значение поворота к оставшемуся расстоянию между целью и положением самолета.

В нашем коде функция updatePlane должна выглядеть следующим образом:

```
function updatePlane(){
    var targetY = normalize(mousePos.y,-.75,.75,25, 175);
    var targetX = normalize(mousePos.x,-.75,.75,-100, 100);

    // Перемещаем самолет в каждом кадре, добавив часть
оставшегося расстояния
    airplane.mesh.position.y += (targetY-
airplane.mesh.position.y)*0.1;

    // Вращаем самолет пропорционально оставшемуся расстоянию
    airplane.mesh.rotation.z = (targetY-
airplane.mesh.position.y)*0.0128;
```

```
      airplane.mesh.rotation.x = (airplane.mesh.position.y-
targetY)*0.0064;

      airplane.propeller.rotation.x += 0.3;
  }
```

Теперь движение самолета выглядит гораздо более реалистичным. Изменяя значение fraction, вы можете сделать чтобы самолет быстрее или медленнее реагировал на движение мыши.

Теперь вы знаете, как создавать объекты из примитивов, как анимировать их и как устанавливать освещение сцены. Вы также увидели как улучшить внешний вид и движение ваших объектов и как настроить окружающий свет.

Следующим шагом, который выходит за рамки данного руководства, поскольку он включает в себя некоторые более сложные методы, была бы реализовация игры на Three.js столкновений, сбора «точек», а также разных уровней. Загрузите исходный код и посмотрите на реализацию; вы увидите все концепции, которые вы узнали до сих пор, а также некоторые продвинутые, которые вы можете исследовать сами.

Вывод

В данной работе был создан универсальный набор рекомендаций с примерами, по созданию модели. А так же описан способ загрузки уже готовой модели, опять таки созданой самостоятельно с помощью какого либо приложения(blender, 3ds max, и т.д.) или взятой из сети. Добавлена возможность анимации, управления(в данном случае с помощью мыши). На основании этих рекомендаций, учитель сможет самостоятельно создавать неограниченое количество моделей и применять их в процессе своей профессиональной деятельности. В качестве наглядного материала, создания тестовых заданий или других форм контроля. Эту задача выполнена на основе Web-платформы с помощью технологии WebGL и использованием библиотеки three.js.

Этот метод даёт следующие преимущества:

- производительность за счёт GPU;
- кроссплатформенность;
- отсутствие компиляции;
- автоматическое управление памятью;
- открытый стандарт.

Недостатки:

- требуется хотя бы базовый уровень знаний в программировании;
- мало переведённой документации;
- большой объём кода.

Однако перечисленные недостатки не могут стать препятствием для испольвания данной технологии. Достаточно высокий уровень конечного

результата и низкая сложность в его достижении скорей является стимулом для преодоления тех незначительных сложностей, которые могут возникнуть.

Список литературы

[1] Вильданов А.Н.3D-моделирование на WebGL с помощью библиотеки three.js: Учебное пособие", 2014 г., 114 с

[2] Добудько Т.В. Формирование профессиональной компетентности учителя информатики в условиях информатизации образования. –Самара: Изд-во СамГПУ, 1999.

[3] Дэвид Флэнаган JavaScript: The Definitive Guide: Символ-Плюс, 2008 г. -992 стр.

[4] Информатика. 6-7 / Под ред. проф. Н.В.Макаровой — СПб.: Питер Ком, 1998.

[5] Информатика. 9 / Под ред. проф. Н.В.Макаровой — СПб.: Питер Ком, 1999.

[6] Николай Прохорёнок "HTML, JavaScript, PHP и MySQL. Джентельменский набор Web-мастера" // 3-е изд., перераб. и доп., Санкт-Петербург: "БХВ-Петербург", 2010, 912с.

[7] НОУ «ИНТУИТ», [Электронный ресурс] 2003 – 2017. URL: http://www.intuit.ru/studies/courses/643/499/lecture/11351?page=2 (дата обращения: 30.04.2017).

[8] Онлайн-книга по WebGL//http://metanit.com/web/webgl/

[9] Основы HTML5: Часть 1. Приступая к работе [Электронный ресурс]. – Режим доступа: http://www.ibm.com/developerworks/ru/library/wa-html5fundamentals/index.html

[10] Пугач В.И. Технологии обучения будущих учителей информатики в педагогических институтах / Самарский гос. пед. Ин-т, 1994.

[11] Рейтинг стран мира по уровню развития информационно-коммуникационных технологий - [Электронный ресурс]. – Режим доступа: http://gtmarket.ru/ratings/ict-development-index/ict-development-index-info

[12] Советский энциклопедический словарь / Под ред. А.М.Прохорова. – М.: Советская энциклопедия,1982.

[13] Старостина В.П. (С) Вопросы интернет образования № 48

[14] Сидорова Наталья Михайловна, учитель информатики и математики - http://nmsidorova.narod.ru

[15] Трехмерное моделирование [Электронный ресурс]. – Режим доступа: http://knowledge.allbest.ru/programming/2c0a65625a3ad78b5c53a88521206c37_0.html

[16] Теория и методика обучения информатике, Лапчик М.П., Семакин И.Г., Хеннер Е.К., Рагулина М.И., 2008

[17] Усенков Дмитрий Юрьевич. - ООО «СТОиК» [Электронный ресурс]. – Режим доступа: http://www.npstoik.ru].

[18] Хбаровский краевой институт развития образования - [Электронный ресурс]. – Режим доступа: http://resource.ippk.ru/mediawiki]

[19] Хенер Е. К. Формирование ИКТ-компетентности учащихся и преподавателей в системе непрерывного образования

[20] Экспертный центр электронного государства - [Электронный ресурс]. – Режим доступа: http://d-russia.ru/ratings

[21] github [Электронный ресурс]. – Режим доступа: https://github.com/bebraw/jswiki/wiki.

[22] JavaScript: The Definitive Guide (Definitive Guides) Paperback – 13 May 2011

[23] Learn PHP 7: Object Oriented Modular Programming using HTML5, CSS3, JavaScript, XML, JSON, and MySQL Paperback – 28 Dec 2015

[24] mrdoob/three.js [Электронный ресурс]. – Режим доступа: https://github.com/mrdoob/three.js

[25] three.js/examples [Электронный ресурс]. – Режим доступа: http://threejs.org/examples/.

[26] WebGL With Three.js – Lesson 6 [Электронный ресурс]. – Режим доступа: https://www.script-tutorials.com/webgl-with-three-js-lesson-6/

Приложение А

```
var lesson6 = {
     scene: null,
     camera: null,
     renderer: null,
     container: null,
     controls: null,
     clock: null,
     stats: null,

   // инициализация
     init: function() {

       // Создать главную сцену
       this.scene = new THREE.Scene();
       this.scene.fog = new THREE.FogExp2(0xcce0ff, 0.0003);

       var SCREEN_WIDTH = window.innerWidth,
           SCREEN_HEIGHT = window.innerHeight;

       // Подготовить камеру
        var VIEW_ANGLE = 45, ASPECT = SCREEN_WIDTH / SCREEN_HEIGHT, NEAR = 1,
FAR = 2000;
         this.camera = new THREE.PerspectiveCamera( VIEW_ANGLE, ASPECT, NEAR,
FAR);
       this.scene.add(this.camera);
       this.camera.position.set(0, 300, 800);
       this.camera.lookAt(new THREE.Vector3(0,0,0));

       // Подготовить визуализатор
       this.renderer = new THREE.WebGLRenderer({ antialias:true });
       this.renderer.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);
       this.renderer.setClearColor(this.scene.fog.color);
       this.renderer.shadowMapEnabled = true;
       this.renderer.shadowMapSoft = true;

       // Подготовить контейнер
       this.container = document.createElement('div');
       document.body.appendChild(this.container);
       this.container.appendChild(this.renderer.domElement);

       // events
       THREEx.WindowResize(this.renderer, this.camera);

       // Подготовить средства управления (OrbitControls)
                  this.controls    =    new    THREE.OrbitControls(this.camera,
this.renderer.domElement);
       this.controls.target = new THREE.Vector3(0, 0, 0);
       this.controls.maxDistance = 2000;

       // Подготовить часы
       this.clock = new THREE.Clock();

       // Подготовить статистику
       this.stats = new Stats();
       this.stats.domElement.style.position = 'absolute';
       this.stats.domElement.style.left = '50px';
```

```
    this.stats.domElement.style.bottom = '50px';
    this.stats.domElement.style.zIndex = 1;
    this.container.appendChild( this.stats.domElement );

    // добавляем освещение
    var spLight = new THREE.SpotLight(0xffffff, 2.75, 2000, Math.PI / 2);

    //spLight.castShadow = true;
    spLight.position.set(-100, 300, -50);
    this.scene.add(spLight);
    var spLight = new THREE.SpotLight(0xffffff, 2.75, 2000, Math.PI / 2);
    //spLight.castShadow = true;
    spLight.position.set(-100, -300, 300);
    this.scene.add(spLight);

     // загрузка модели
    this.loadModel();
  },
  loadModel: function() {

    // подготовка и загрузка модели
    var oLoader = new THREE.OBJLoader();
    oLoader.load('models/HDD.obj', function(object, materials) {

      // var material = new THREE.MeshFaceMaterial(materials);
      var material2 = new THREE.MeshLambertMaterial({ color: 0xa65e00 });

      object.traverse( function(child) {
        if (child instanceof THREE.Mesh) {

          // пользовательский материал
          child.material = material2;

          // включить отбрасывание тени
          child.castShadow = true;
          child.receiveShadow = true;
        }
      });

        // Точка 10
      object.position.x = 0;
      object.position.y = 0;
      object.position.z = 0;
      object.scale.set(1, 1, 1);
      lesson6.scene.add(object);
    });
  }
};

// анимация сцены
function animate() {
  requestAnimationFrame(animate);
  render();
  update();
}
```

```
// Обновление управления и статистика
function update() {
  lesson6.controls.update(lesson6.clock.getDelta());
  lesson6.stats.update();
}

// визуализация сцены
function render() {
  if (lesson6.renderer) {
    lesson6.renderer.render(lesson6.scene, lesson6.camera);
  }
}

// Инициализировать урок при загрузке страницы
function initializeLesson() {
  lesson6.init();

  animate();
}

if (window.addEventListener)
  window.addEventListener('load', initializeLesson, false);
else if (window.attachEvent)
  window.attachEvent('onload', initializeLesson);
else window.onload = initializeLesson;
```