# ТИТУЛЬНЫЙ ЛИСТ

# ЗАДАНИЕ (ТЗ)

# РЕФЕРАТ

Дипломная работа на тему «Программный каркас для создания спрайтовой анимации на HTML5» решает проблему сложности создания интерактивных приложений с двумерной графикой с применением 2d-context HTML5. Исходя из поставленной задачи, в работе подробно рассмотрена область компьютерной графики в целом и проблема сложности использования 2d-context в частности. В работе предлагается метод решения упомянутой проблемы с помощью создания специализированного каркаса.

Работа представляет интерес для специалистов, работающих в области создания мультимедийных приложений(# игр), а также для всех, кому интересна данная тематика. Работа содержит 2 рисунков и 2 приложения. Общий объем работы составляет 51 страницу. Структура работы представлена списком определений, введением, десятью главами, заключением, списком литературы, а также приложениями

# СОДЕРЖАНИЕ

[ТИТУЛЬНЫЙ ЛИСТ 0](#_Toc355909764)

[ЗАДАНИЕ (ТЗ) 1](#_Toc355909765)

[РЕФЕРАТ 2](#_Toc355909766)

[СОДЕРЖАНИЕ 3](#_Toc355909767)

[ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 5](#_Toc355909768)

[ВВЕДЕНИЕ 6](#_Toc355909769)

[1. ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ И ПОСТАНОВКА ЗАДАЧИ 7](#_Toc355909770)

[1.1. HTML5 7](#_Toc355909771)

[1.2. SVG, 2d-context, WebGL-context, DOM – как возможности HTML5 для создания графических приложений или игр 8](#_Toc355909772)

[1.3. Описание 2d-context 10](#_Toc355909773)

[2. СЛОЖНОСТИ И ПРОБЛЕМЫ РАЗРАБОТКИ ПРИ ПОМОЩИ 2D-CONTEXT 10](#_Toc355909774)

[3. ВАРИАНТЫ РЕШЕНИЯ ПРОБЛЕМ И СЛОЖНОСТЕЙ РАЗРАБОТКИ ПРИ ПОМОЩИ 2D-CONTEXT 10](#_Toc355909775)

[4. ТРЕБОВАНИЯ К ПРОГРАММНОМУ КАРКАСУ 11](#_Toc355909776)

[5. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ 11](#_Toc355909777)

[6. АРХИТЕКТУРА КАРКАСА 11](#_Toc355909778)

[6.1. Проектирование архитектуры 11](#_Toc355909779)

[6.2. Парадигма программирования 11](#_Toc355909780)

[6.3. UML – диаграммы 11](#_Toc355909781)

[6.4. Используемые шаблоны проектирования 13](#_Toc355909782)

[7. ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ 14](#_Toc355909783)

[8. СРЕДСТВА РАЗРАБОТКИ 14](#_Toc355909784)

[8.1. Интегрированная среда разработки 14](#_Toc355909785)

[8.2. Дебагер 16](#_Toc355909786)

[8.3. Профайлер 16](#_Toc355909787)

[8.4. Система контроля версий 16](#_Toc355909788)

[9. Принятые стандарты кодирования 16](#_Toc355909789)

[10. Ход работы 16](#_Toc355909790)

[10.1. Выбор сторонних вспомогательных каркасов\библиотек 16](#_Toc355909791)

[10.2. Интеграция классов каркаса 16](#_Toc355909792)

[РЕЗУЛЬТАТ 16](#_Toc355909793)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc355909794)

[ПРИЛОЖЕНИЯ 16](#_Toc355909795)

[ТЗ 16](#_Toc355909796)

[Руководство программиста (ГОСТ 19.504) 16](#_Toc355909797)

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

1. *RIA* – (от англ. Rich Internet application) — это веб-приложение, доступное через Интернет, насыщенное функциональностью традиционных настольных приложений.
2. *OpenGL ES 2.0* — это подмножество графического интерфейса OpenGL, разработанное специально для встраиваемых систем — мобильных телефонов, карманных компьютеров, игровых консолей.
3. *Шейдер* – это программа, которая используется в трёхмерной графике для определения окончательных параметров изображения или объекта.
4. *DOM* (от англ. Document Object Model) — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.
5. *SVG* (от англ. Scalable Vector Graphics) — язык разметки масштабируемой векторной графики, созданный Консорциумом Всемирной паутины (W3C) и входящий в подмножество расширяемого языка разметки XML, предназначен для описания двумерной векторной и смешанной векторно/растровой графики в формате XML.

# ВВЕДЕНИЕ

Еще недавно самым популярным средством создания интерактивных графических веб-приложений, насыщенных анимацией, трехмерной графикой и т.д., была технология Flash от фирмы Adobe[10], что требовало покупки среды разработки Flash Professional и изучения разработчиком языка программирования Acton Script, а так же требовало от пользователя установки в браузер средства для воспроизведения(Flash Player) Flash-роликов. Долгое время Flash был по сути был единственной технологией для создания игр, мультфильмов, интерактивны веб-приложений. Для всех основных платформ существовал и поддерживался Flash-player, но с развитием открытой технологии HTML5, постепенно, Adobe отказалось от поддержки Flash-player для операционных систем семейства Linux[11], а затем и от поддержки для мобильных платформ [12], так же ряд крупных фирм отказались от поддержки Flash:

* Apple в пользу HTML5 для iPhone, iPod touch и iPad;
* Microsoft в пользу Silverlight для Windows Phone 7 и HTML5 для Windows 8;
* Oracle в пользу JavaFX.

С появлением поддержки нового, предоставляющего схожие с Flash графические возможности, но еще не готового стандарта HTML(HTML5), ведущими производителями браузерных ядер (WebKit (Chrome), Gecko(FireFox), Presto(Opera)), перед разработчиком интерактивных веб-приложений встает необходимость использования графических возможностей HTML5.

В данной работе решается проблема сложности разработки графических веб-приложений с помощью 2d-context HTML5.

# ВВЕДЕНИЕ В ПРЕДМЕТНУЮ ОБЛАСТЬ И ПОСТАНОВКА ЗАДАЧИ

# HTML5

HTML5 (*англ. HyperText Markup Language, version 5*) – это язык разметки гипертекстовых документов являющийся одной из самой популярной технологией в интернете. Это пятая версия HTML-стандарта пришедшая на смену HTML4. Стандарт HTML5 находится в стадии тестирования – W3C объявил о планах, согласно которым окончательная версия стандарта HTML5 будет утверждена лишь к 2014 году[6].

HTML5 по сравнению с HTLM4 вводит новые интерфейсы:

* Медиа теги video[2] и audio[3], позволяющие воспроизводить видео и музыку на странице;
* Интерфейс позволяющий создавать веб-приложения, для работы, которых интернет соединение необходимо только на этапе загрузки[4];
* Drag&Drop интерфейс;
* Интерфейс 2d-context[5] для создания растровой графики;
* Интерфейс webgl-context[6] для создания трехмерной графики;
* Интерфейс к истории посещения[7];
* Интерфейс к хранилищу данных типа ключ–значение на стороне клиента[8].

Создание графических приложений в HTML возможно с помощью следующих технологий:

1. 2d-context – контекст тега canvas;
2. Webgl-context – контекст тега canvas;
3. SVG – язык описания векторных изображений;
4. Манипуляции DOM-моделью HTML документа.

# SVG, 2d-context, WebGL-context, DOM – как возможности HTML5 для создания графических приложений или игр

Выбрав HTML5 для создания графического приложения, разработчику доступны:

1. 2d-context – Это контекст тега <canvas> предоставляющего интерфейс для создания двумерной растровой графики. 2d-context позволяет манипулировать изображением на плоской двумерной системе координат с центром в левом верхнем углу экрана. Интерфейс представлен набором методов и свойств, определяющих графические примитивы их внешний вид и аффинные преобразования над экранной плоскостью.

Данный контекст можно считать простой альтернативой технологии Flash и вероятно будет подходить для создания простых игр, анимированных интерфейсов и т.д.

1. Webgl-context – Это контекст тега <canvas> предоставляющего интерфейс для создания трехмерной графики. Интерфейс является производным от OpenGL ® ES 2.0 и имеет схожие возможности, включая работу с вершинными и пиксельными шейдерами. Webgl-context позволяет задействовать вычислительные мощности видеокарты, что может подвергать пользователя риску, через открытие доступа к привилегированному режиму видеокарты и оборудования[13];
2. SVG – язык описания векторных изображений, построенный на языке XML.
3. Манипуляции DOM-моделью HTML документа – С самого появления языка HTML, HTML документ состоял из набора тегов определявших внешний вид и назначение различных элементов на странице. Манипуляция тегами, т.е. структурными элементами DOM–модели, можно рассматривать как возможность создания спрайтовой анимации. Этот метод мало применим для достаточно сложной игры из-за «подрагивания» или «мелькания» при достаточно частом обновления страницы. Манипуляции с DOM-моделью не предназначены для частых динамичных изменений.

Для сравнения 2d-context и SVG, можно использовать тест приведенный по ссылке[14]: тест состоял в отслеживании числа кадров в секунду (FPS) при изменении количества объектов (движущиеся с максимальной скоростью квадраты) на экране при использовании 2d-context и при использовании SVG. Результаты теста на ОС Windows 7, приведены на рисунке1.



Рисунок 1 – Сравнительный тест технологий 2d-context и SVG

Как можно видеть на графике (Рисунок 1), 2d-contex более производителен при большом количестве объектов и так как основными задачами, решаемыми с помощью графических возможностей HTML5 и Flash, чаще всего являются реализация спрайтовой анимация и работа с изображениями, можно сделать вывод, что для замены технологии Flash больше всего подходит 2d-context ведь он, как и Flash, реализует работу с растровой графикой и позволяет работать с большим числом объектов. Очевидно, что выбирая между 2d-context и SVG для реализации приложения, которое должно генерировать динамичное изображение состоящее из большого количества объектов, следует выбрать 2d-context.

# Описание 2d-context

# СЛОЖНОСТИ И ПРОБЛЕМЫ РАЗРАБОТКИ ПРИ ПОМОЩИ 2D-CONTEXT

Использование 2d-context влечет за собой следующие трудности:

1. Неготовый стандарт. Стандарт в стадии тестирования – W3C объявил о планах, согласно которым окончательная версия стандарта HTML5 будет утверждена лишь к 2014 году[6];

2. Отсутствие визуальных сред, вроде Flash Professional CS6;

3. Слабое развитие специализированных каркасов, вызванные, скорее всего, незавершенностью стандарта;

4. Разработка без использования специализированного каркаса требует большой объем кода для реализации несложной графики.

Но тем не менее, не смотря на трудности, HTML5 перспективная технология, т.к. не зависит от какой то одной компании, имеет открытый стандарт и рекомендуется такими компаниями как Apple и Microsoft.

# ВАРИАНТЫ РЕШЕНИЯ ПРОБЛЕМ И СЛОЖНОСТЕЙ РАЗРАБОТКИ ПРИ ПОМОЩИ 2D-CONTEXT

Одним из вариантов решения описанных выше проблем может быть создание каркаса, сравнимого по гибкости с разработкой на API и позволяющего уменьшать количество кода за счет готовой реализации некоторых программных решений (# оптимизация по времени прорисовки, готовый цикл анимации и т.д.). Каркас призван облегчить труд разработчика и дать возможность сосредоточить свое внимание на реализации конкретного приложения.

# ТРЕБОВАНИЯ К ПРОГРАММНОМУ КАРКАСУ

# ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

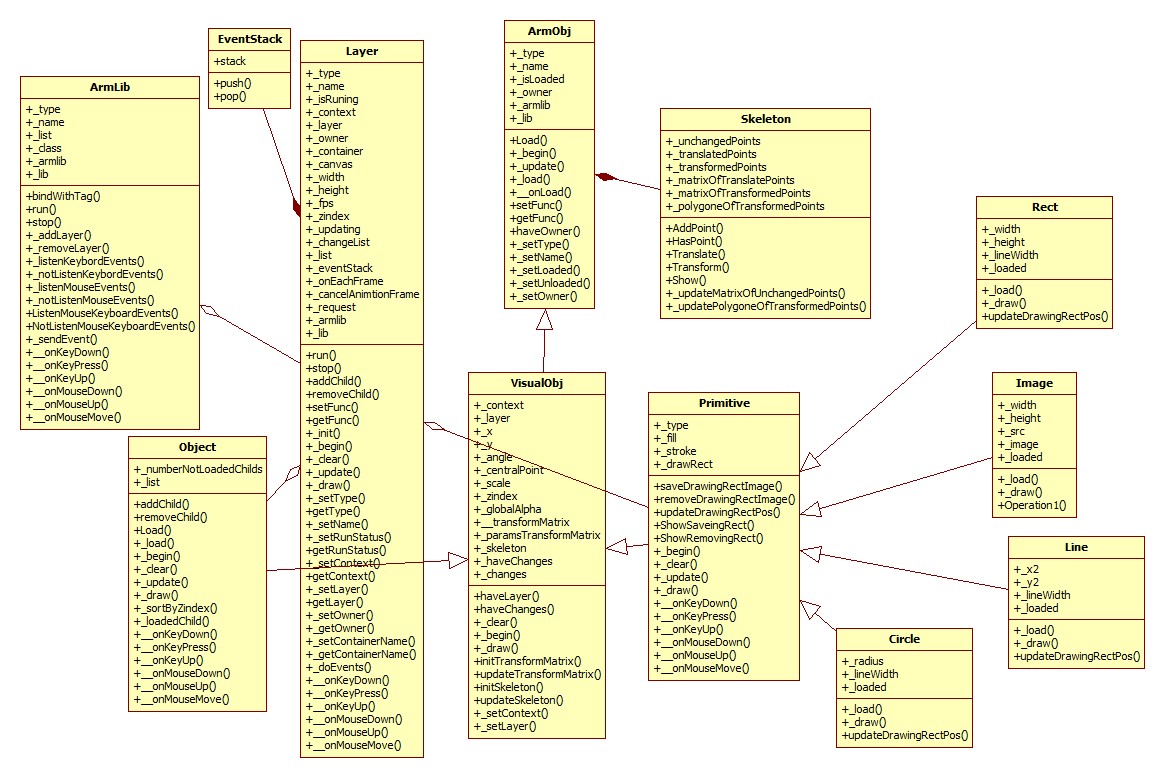
# АРХИТЕКТУРА КАРКАСА

# Проектирование архитектуры

# Парадигма программирования

# UML – диаграммы

Перед началом разработки приложения была создана диаграмма классов (Рисунок 2).

****

**Рисунок 2 – UML диаграмма классов**

Данная диаграмма (Рисунок 2) отображает отношения между классами каркаса. Класс ArmLib, является менеджером каркаса, т.е. через его методы происходит управление всем каркасом. Можно видеть, что класс Layer и Object агрегирует в себе объекты классов Object и классы производные от класса Primitive. Данное отношение между классами позволяет ввести абстракцию объектов, т.е. структурные части приложения представлены не просто набором классов, функций и т.д., а набором объектов классов Object, Image, Circle, Line, Rect и производными от этих классов. Данный подход позволяет уменьшить сопряжение[15] и сделать структурные части каркаса более выраженными. Классы Rect, Line, Circle, Text, Image описывают графические примитивы, с помощью которых описываются визуальные объекты приложения.

# Используемые шаблоны проектирования

В ходе разработки сложного программного обеспечения важно создавать качественный, готовый к сопровождению другими программистами, код. Одной из методик уменьшения сложности сопровождения, является использование, так называемых, шаблонов проектирования программного обеспечения. Шаблоны проектирования описывают повторимые архитектурные конструкции, представляющие собой решение проблемы проектирования в рамках некоторого часто возникающего контекста. Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться. Использование широко известных шаблонов, делает код более «читаемым».

В каркасе были использованы ряд шаблонов, а именно:

**Порождающие шаблоны:**

* «Одиночка», данный шаблон использован в классе ArmLib, для гарантирования существования только одного объекта класса ArmLib;
* «Фасад», данный шаблон использован в классах графических примитивов: Rect, Line, Circle, Image. В данных классах данный шаблон дает возможность более удобно работать с API контекста рисования HTML5;
* «Компоновщик», данный шаблон использован в классах ArmLib, Layer и Object для создания древовидной структуры отношений примитивов и объектов сцены.

**Фундаментальные шаблоны**

* «Делегирование», данный шаблон является базовым для приложения созданного в объектно-ориентированном стиле. Данный шаблон используется почти во всех классах каркаса.

**Поведенческие шаблоны**

* «Наблюдатель», данный шаблон используется в классах ArmLib, Layer и Object для оповещения объектов класса Layer в классе ArmLib, объектов класса Object, Image, Rect, Line в класах Object и Layer о событиях клавиатуры и мыши.

# ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ

# СРЕДСТВА РАЗРАБОТКИ

# Интегрированная среда разработки

«По некоторым оценкам до 40% рабочего времени программист тратит

на редактирование исходного кода» [16]

Среда разработки является инструментом призванным упрощать разработку программного продукта, за счет интеграции таких инструментов как: текстовый редактор, компилятор, средства оптимизации сборки, система контроля версий и т.д, в одном продукте.

Выбор среды разработки, является критичным как с точки зрения времени выполнения проекта, так и с точки зрения качества проекта. Грамотный выбор подходящей среды разработки может позволять: уменьшать время кодирования и количество ошибок, за счет:

* Подсветки синтаксиса используемого языка программирования;
* Наглядном представлении файлов проекта;
* Интеграции с системами контроля версий;
* Интеграция с системами тестирования;
* Использовании различных плагинов, например jsLint и т.д.

Для разработки была выбрана интегрированная среда разработки Netbeans. Выбор Netbeans обусловлен:

* Стоимостью, среда бесплатна;
* Большим комюнити, в случае возникновения проблем проще найти решение;
* Поддержкой синтаксиса языка JavaScript;
* Поддержка распределенной системы контроля версий Git.
* Возможность использовать утилиту jsLint в виде плагина, что позволяет использовать данную утилиту совместно со средой разработки.

# Дебагер

# Профайлер

# Система контроля версий

# Принятые стандарты кодирования

# Ход работы

# Выбор сторонних вспомогательных каркасов\библиотек

# Интеграция классов каркаса

# РЕЗУЛЬТАТ

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# ПРИЛОЖЕНИЯ

# ТЗ

# Руководство программиста (ГОСТ 19.504)