1. Добрый день уважаемая комиссия и все, присутствующие сегодня здесь.
2. Цель моей работы – разработка и реализация программного каркаса для создания спрайтовой анимации на HTML5.

Для достижения данной цели было необходимо:

* Ознакомится с программным интерфейсом 2d-context.
* Ознакомиться с существующими решениями и технологией создания интерактивных графических веб-приложений.
* Выявить требования к каркасу.
* Спроектировать и реализовать программный каркас.

1. **Что же такое 2d-context HTML5?**

2d-context является веб-технологией позволяющей создавать интерактивные графические веб-приложения рамках HTML5.

С программной точки зрения 2d-context является контекстом тега <canvas> и представляет интерфейс для создания двумерной растровой графики. Данный контекст 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. Данная технология реализует непосредственный режим графики, то есть программист должен сам заботится о перерисовки каждого кадра, реализации необходимых графических алгоритмов и т.д.

1. **Рассмотрим спецификацию на данную технологию:**

Спецификация на технологию это определение и перечень специфических особенностей данной технологии.

В случае с 2d-context, спецификация в первую очередь определяет программный интерфейс.

Приводить весь текст спецификации не имеет смысла из-за его большого объема, приведу общее описание функций интерфейса:

В ходе анализа интерфейса, методы и атрибуты 2d-context можно условно разделить на:

1. Методы работы с путями.
2. Методы и атрибуты для визуализации текста.
3. Методы для визуализации графических примитивов:
   * 1. Прямоугольник
     2. Изображение
     3. Линия
     4. Квадратичные кривые
     5. Кубические кривые
     6. Дуга
     7. Градиент
4. Методы для работы с массивом пикселей.
5. Методы сохранения и восстановления контекста.
6. Метод определения принадлежности точки пути.

**Можно заметить, что методы и атрибуты 2d-context позволяют:**

* Программировать изображение, но не позволяют работать с отдельными частями экрана как с объектами.
* Для создания анимации требуются перерисовка экрана для изменения изображения. Это вызвано отсутствием методов для создания анимации.
* Описание несложных сцен требуют достаточно большое количество кода. Это вызвано тем фактом, что изображение создается с помощью минимального набора графических примитивов.

**Трудности связанные с 2d-context**

Определенные особенности использования интерфейса и некоторые проблемы, связанные с технологией приводят к актуальности моей дипломной работы.

Я имею ввиду такие трудности как:

1. Нет возможности работать с частями изображения как с самостоятельными объектами
2. Для создания анимации требуются перерисовка экрана. Анимация это последовательная смена кадров, но так как 2d-context направлен на создание статичного изображения, он не имеет соответствующих методов
3. Описание несложных сцен требуют достаточно большое количество кода. Это вызвано тем фактом, что изображение создается с помощью минимального набора графических примитивов.
4. Неготовый стандарт. Стандарт в стадии тестирования – Консорциум всемирной паутины (World Wide Web Consortium) объявил о планах, согласно которым окончательная версия стандарта HTML5 будет утверждена лишь к 2014 году
5. Отсутствие визуальных сред, вроде Flash Professional CS6. Проблема в отсутствии возможности не программировать интерфейс графического приложения, а описывать в некотором визуальном редакторе
6. Слабое развитие специализированных каркасов, вызванное, скорее всего, незавершенностью стандарта.

**Требования к каркасу**

В ходе анализа существующих решений и анализа процесса создания графических приложений с использованием 2d-context, были выявлены следующие требования:

**Объектное представление примитивов**, **Работа с мышью и клавиатурой**

В мультимедиа приложении, интерактивность достигается за счет взаимодействия пользователя с отдельными элементами на экране, при этом элементы представлены набором графических примитивов.

Можно сделать вывод, что для изменения внешнего вида элемента необходимо изменять свойства каждого примитива входящего в состав элемента, то есть для разработчика важно иметь возможность представлять элемент интерфейса с помощью совокупности графических примитивов, возможность управлять внешними видом совокупности примитивов как одной сущности;

**Call-back функции для событий объектов**

В случае представления некоторой сущности на экране в виде некоторого объекта, нужно понимать, что объект может иметь не только свойства и методы, но и должен иметь возможность реагировать на некоторые асинхронные события, например: реагировать на щелчок мыши;

**Загрузка мультимедиа-зависимостей объектов**

Такие примитивы как изображение или видео перед выводом на экран требуют загрузки данных. Так как с помощью этих примитивов описывается некоторая сущность, то сущность должна быть описана максимально целостно. Сущность должна иметь возможность самостоятельно загрузить необходимые данные и в соответствующем обработчике среагировать на некоторые события связанные с загрузкой;

**Приоритеты очереди прорисовки**

Объекты могут пересекаться на экране и определенные объекты должны находится выше или ниже других по этому нужен способ указывать приоритет объекта на прорисовку;

**Графический цикл**

Так как 2d-context представляет так называемый непосредственный графический режим, то программист имеет в распоряжении минимальный набор методов для создания некоторой картинки, но если есть потребность в изменяющемся динамичном изображении программист должен сам думать о очистке экрана, перед новым кадром, организации смены кадров и т.д.

Можно сделать вывод, что в каркасе должен быть реализовани графический цикл, то есть программист не должен забоится о организации смены кадров;

**Слои**

Под слоями подразумевается набор тегов <canvas>, расположенных друг над другом. Одни тег – один слой. Визуальные объекты распределяются по слоям, что позволяет разделить объекты по некоторой логике. Так же с помощью слоев можно уменьшить время прорисовки кадра если вынести на отдельный слой объекты, редко меняющее свое изображение. Так как объекты не меняются, то нет смысла их перерисовывать;

**Модульность**

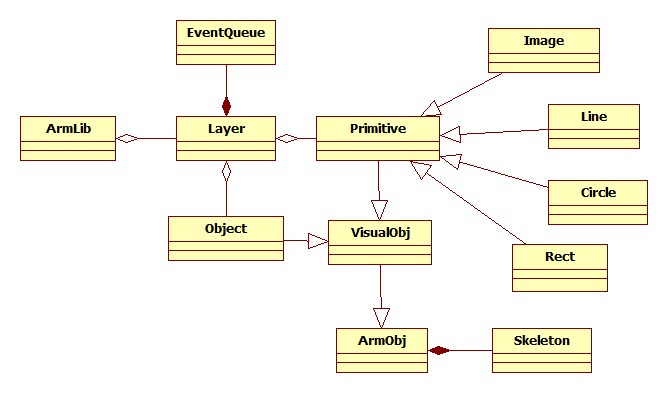
Необходимо учитывать, что размер каркаса влияет на общее время загрузки и старта приложения, а так как при создании достаточно универсального каркаса, в разных приложениях могут быть задействованы далеко не все возможности, необходимо реализовать модульности каркаса. Должна быть возможности собирать каркас для конкретного приложения, из необходимых для конкретного приложения, модулей;

**Вызовы методов каркаса цепочкой**

При анализе существующих каркасов была отмечена такая функция как возможность вызывать методы объекта не только из самого объекта, но и сразу после вызова методов объекта не возвращающих значение. Данная функциональность используется и в крупных библиотеках и каркасах не связанных с графикой, например в jQuery и позволяет сделать код более читаемым;

**Спроектировать и реализовать программный каркас**

Структуру разработанного каркаса можно представить следующей диаграммой:



Классом высшего уровня является класс ArmLib, данный класс содержит основные методы управления каркасом. Класс ArmLib содержит в себе объекты класса Layer.

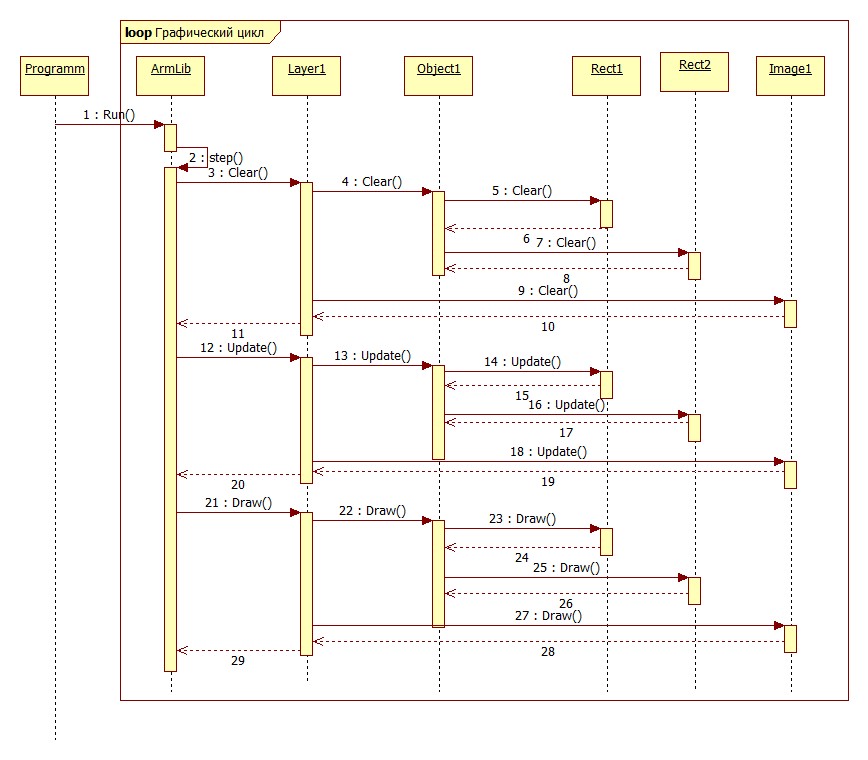
Класс Layer описывает слой и содержит в себе объекты классов наследующих от класса VisualObj. VisualObj является общим описанием визуального объекта. Класс Primitive является общим описанием графических примитивов.

Класс EventQueue данный класс описывает очеред сообщений объекта класса ArmLib объектам класса Layer. Примером сообщения может служить событие щелчка мыши или нажатие кнопок клавиатуры.

Каждый графический примитив представлен классом наследником класса Primitive. Класс Skeleton описывает полигон точек описывающих очертания определенного визуального объекта.

Класс Object описывает объект-контейнер содержащий в себе объекты классов производных от класса VisualObj. Данный класс позволяет ассоциировать набор визуальных объектов с некторым одним объектом.

На следующей диаграмме приведен процесс работы главного цикла каркаса:



Для разработки каркаса были использованы следующие инструменты:

* Netbeans в качестве IDE;
* Браузеры Chrome, Firefox, Opera для тестирования каркаса;
* Статический анализатор кода JSLint;
* Библиотека Gizmo
* Отладчик и профайлер браузера Chrome;
* Система контроля версий Git;
* Язык программирования JavaScript.

**В ходе работы были достигнуты следующие результаты:**

* Разработан и реализован программный каркас для упрощения разработки графического веб-приложения с помощью технологии 2d-context HTML5.
* Написана статья на тему: «Программный каркас для создания спрайтовой анимации на HTML5». Данная статья получила третье место в секции «Технологии разработки и проектирования информационных систем» на конференции «Технологии Microsoft в теории и практике программирования».
* Получен реальный опыт использования системы контроля версий
* Улучшено знание языка JavaScript

В дальнейшем работа по данной теме будет продолжена. В частности планируется существенно пересмотреть архитектуру каркаса с целью повышения гибкости разработки с использованием каркаса.