МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «КубГУ»)

Физико-технический факультет

Кафедра теоретической физики и компьютерных технологий

К У Р С О В О Й П Р О Е К Т

ТЕХНОЛОГИИ ТЮНИНГА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА В БРАУЗЕРНЫХ И НАСТОЛЬНЫХ ПРИЛОЖЕНИЯХ

Работу выполнила _____ Бутенко Надежда Сергеевна

Курс 3

Направление 09.03.02 Информационные системы и технологии

Научный руководитель

старший преподаватель _____ В. Н. Значко

Нормоконтролер инженер _____ Г. Д. Цой

Краснодар 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цели данного курсового проекта: разработка демонстрационного приложения, позволяющего выполнять настройку любых графических элементов, а также изучение технологий QSS и CSS и стандартных способов их применения. Задачи: изучение технологий QSS и CSS и стандартных способов их применения, а также разработка демонстрационного приложения для настройки дизайна интерфейса.

Актуальность данной темы крайне высока, ввиду того, что в настоящее время таким аспектам, как User Interface и User Experience уделяется всё больше и больше внимания. Приложение, написанное в рамках данного курсового проекта, являет собой полезный инструмент для дизайнеров.

1 CSS

В 1989 году участник европейского совета по ядерным исследованиям Тим Бернерс-Ли предложил проект World Wide Web (Всемирная паутина), сейчас этот проект более известен как Интернет. При работе над данным проектом был применен язык разметки HTML. Первый сайт в Интернете был написан на чистом HTML. В первых стандартах HTML такого понятия как веб-дизайн еще не было. Не имея средств для изменения внешнего вида, все старания разработчиков были направлены на доступность текста на любых устройствах, способных отображать текст. То есть, первоначально разработчики управляли разметкой только логическими тегами, тегами заголовков, подзаголовков, списками, абзацами, цитатами [1].

Впоследствии, для подержания конкуренции стали появляться теги для облагораживания проектов. Когда же код HTML стал выглядеть перегруженным, и удобство его изменений вызывало сомнения, появился первый стандарт CSS.

1.1 История CSS

Cascading Style Sheets (CSS) - каскадные таблицы стилей.

Концепция каскадных таблиц стилей был разработана в те времена, когда в распоряжении у веб-дизайнеров ничего более изысканного, чем таблицы, не было. А тег <h1> использовался как способ сделать текст крупнее, так же для добавление фона применяли теги <table>, пример на листинге 1. В те «темные времена» норвежский учетный и специалист в области информатики Хокон Виум Ли предложил первую версию CSS [2].

```
<font face=Arial color=#0000FF size=6>
<br>
<h1>Big Font</h1>
<table width=200 height=300 border=5
bordercolor=navy bgcolor=lightblue>
<td background=clouds.jpg>
<img src=spacer.gif width=4 height=20>
<img align=left border=2>
<br clear=left>
```

Листинг 1 - Пример кода

В основу его предложения была заложена мысль отделения дизайна от основной части сайта, чтобы программисты имели возможность изменять части сайта независимо друг от друга. Но все же самый первый CSS стандарт не подразумевал под собой ничего сверхъестественного, с помощью всего нескольких сточек нельзя было управлять стилями многостраничного сайта. Но это помогло убрать из HTML-кода теги <font> и
, на листинге 2 пример некорректного кода. Первоначально CSS не был полноценным инструментом для разметки сайта. Он просто дал возможность вынести одноименные атрибуты тегов HTML width и height, так же свойство float, заменил атрибут align элементов <img> и <table>.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset= "utf-8">
<title> Лабораторная работа </title>
</head>
```

```
<body vlink="#2F4F4F"
background="85426c71e99731b94312922d4dccb2ff (1).png"
bgcolor="#FFE4E1" text="#000000">
<a href= "mailto:volha.black@gmail.com"> Электронный
ящик </a>
<br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br>
<br><br><br><br><br><br><br><br><br><br><br><br><br>
<p> Случайный текст </p>
</body>
```

Листинг 2 - Пример кода

Хоть и по тем временам это считалось прорывом, это никому не было нужно. Тем более, что тот вариант CSS не предназначался для размещения блоков, а только давал возможность работать с обтекаемыми текстом «врезками». Работу с картинками проводили через таблицы.

В ходе развития веб-программирования не было создано определенных рамок и ограничений. Это привело к разнообразным подходам, а так же методам отображения в браузерах, так как каждый браузер создавал свою каскадную таблицу стилей. В конечном итоге был создан так называемый Консорциум Всемирной паутины W3C. Организация на добровольных правах создавала стандарты, которые устраивали бы всех. В ходе стандартизаций был добавлен и изменен функционал CSS, то есть был выпущен второй стандарт. Новый стандарт частично воплотил идею автора о независимой от использованных тегов верстке, и с помощью селектора display давал возможность использовать в верстке более семантическую разметку. Списки и навигацию можно было отобразить в виде вкладок, а выравнивание столбца осуществлялось намного проще, всего в одну строку, это продемонстрировано на листинге 3.

```
.sidebar, main {display: table-cell; }
```

Листинг 3 - Пример кода

Несмотря на удобство CSS, популярные на тот момент кросс-платформенные браузеры отображали его некорректно, что перечеркивало весь их смысл. Но пользователи, получив совет к использованию таблиц консорциумом, стали применять более сложные методы, например сложную связку абсолютного позиционирования и float-элементов, это продемонстрировано на листинге 4.

```
float: left;
position: absolute;
.clearfix
```

Листинг 4 - Пример кода

В то же время были готовы некоторые функции третьего стандарта, например типы селекторов и цветов. Но разночтения в браузерах, ошибки и не реализованные функции так же пестрили ошибками, что делало кросс-браузерную верстку весьма неприятным занятием.

В 2002 году разработчики прекратили активную разработку нового стандарта и занялись исправлением ошибок и улучшением кросс-браузерности своего продукта. Так же было уделено немалое внимание по созданию мануалов. В то же время с помощью тестовой страницы Acid2 разработчики браузеров смогли улучшить поддержку CSS. За это время разработчики создали крепкую основу для стандартов CSS3.

К концу работ над CSS2.1, что случилось в конце 2000-х годов, была возобновлена работа над следующим стандартом. В мае 2011 года после полной

остановки работ над вторым стандартом рабочая группа в полную силу приступила к созданию следующей версии.

Третий стандарт воплотил основную задумку автора. Теперь CSS стал модульным, то есть он был разделен на части, которые развиваются в отдельности друг от друга.

В итоге CSS3 можно разделить на четыре основных направления.

Первое направление - обработка селекторов и медиа-запросов. Здесь важна техническая поддержка параметров различных устройств, на которых требуется загружать те или иные стили. Но медиа-запросы позволяют не только определять размеры используемого аппарата, но и помогают разработчикам создавать адаптивные макеты.

Второе направление – оформление. Это основная и очень важная часть CSS. Новый стандарт расширил возможности разработчиков, предоставив те возможности, о которых в 1994 даже не мечтали.

По статистике 95% сайтов несут в себе текстовый контент. Топографика и интернационализация играют немаловажную роль в создании сайта. С помощью шрифтового оформления можно задавать настроение и располагать к себе аудиторию. Так же немалое значение имеют расположение, переносы и выравнивание текста – на этом всём зиждется третье направление.

Четвертое направление о компоновке. Компоновка определяет местоположение блоков относительно друг друга, что является очень сложным и важным процессом.

1.2 Стандарты CSS

Как было сказано ранее, CSS прошел долгий путь и три стандарта, расширяя и дополняя язык разметки. Тем не менее, откинув историю, можно увидеть весь масштаб работы, проделанной разработчиками, и понять, с чем приходилось работать ранее, а что современные программисты имеют сейчас.

Самый первый стандарт CSS1 хоть и имел множество проблем с реализацией в браузерах, но предоставлял функции, которые стали фундаментом для более поздних версий [3]. Не обращаясь к функционалу HTML программист, мог:

-управлять способами отображения информации на странице;

-запрещать или разрешать обтекание элементов;

-управлять размерами элементов;

-управлять внешними и внутренними отступами;

-управлять вертикальным выравниванием в таблицах;

-управлять границами элементов: цвет, стиль или ширина;

-форматировать нумерованные и ненумерованные списки: изменять типы маркеров, обтекание маркеров текстом, а также применять в качестве маркеров ненумерованных списков изображения;

-задавать цвет текста и цвет фона;

-задавать в качестве фона изображение, а также изменять его параметры;

-управлять параметрами шрифтов;

-управлять параметрами текста, например изменять регистр текста;

-управлять междустрочным интервалом и расстоянием между словами.

Ко второй версии были частично исправлены ошибки прошлого стандарта, но, тем не менее, CSS2 хоть и был одобрен консорциумом W3C, по-прежнему имел свои ошибки, а также разночтения в браузерах. Тем не менее, предоставлял следующие возможности:

-блочная вёрстка: появились относительное, абсолютное и фиксированное позиционирование, а также возможность управлять размещением элементов на странице без табличной вёрстки;

-определение типа носителей позволяет устанавливать разные стили для разных носителей;

-звуковые таблицы стилей: для определения голоса, громкости и т. д. для звуковых носителей;

-страничные носители: для установки разных стилей для элементов на чётных и нечётных страницах при печати;

-расширенный механизм селекторов;

-были добавлены указатели;

-генерируемое содержимое: позволяет добавлять содержимое, которого нет в исходном документе, до или после нужного элемента.

Если вспомнить историю CSS, разработчики остановили разработку следующего стандарта из-за множества ошибок в самом стандарте и в браузерном отображении. В связи с этим были выпущены дополнения, исправляющие все, что было упущено из вида. CSS2.1 давал такие возможности:

-задавать направление текста: слева-направо или справа-налево;

-задавать видимую область элемента и обрезать все остальное;

-изменять внешний вид курсора;

-реализация «слоистой» структуры;

-задавать минимальные и максимальные размеры элементов;

-указывать расстояние между ячейками в таблице;

-изменять обводку элементов;

-отдельно указывать параметры для стилистического изменения каждой грани таблицы;

-определять фиксированные размеры элементам таблицы;

-управлять внешним видом кавычек, в которые оборачиваются цитаты;

Третий стандарт CSS3 был не такой масштабный, как прошлый, но имел блочный характер разработки. Построенный на основе CSS2 и CSS2.1, дополнил прошлые функции и добавил новые:

-оформление множественных фонов;

-скругление углов;

-добавление рамок и градиентов;

-возможность добавлять вертикальный текст;

-добавление пользовательских шрифтов;

-переносы строк;

-выравнивание текста;

-компоновка страницы;

-создание анимированных элементов без использования JavaScript;

-поддержка линейных и радиальных градиентов;

-возможность создавать тени.

С 29 сентября 2011 года консорциумом W3C ведется разработка следующего, четвертого стандарта CSS. Модули CSS4 разрабатываются на основе CSS3, но имеют пока только черновой вариант.

1.3 Преимущества CSS

Многие могут заметить, что оформление сайта можно сделать и средствами HTML, но таким образом на разработку сайта будет затрачено много времени, и это может привести к неприятным последствиям в бедущем. Поэтому давайте рассмотрим главные преимущества CSS перед HTML.

Первым преимуществом CSS является его простота в изучении. Так же большой плюс каскадных таблиц стилей - это его принцип отделения оформления от функционала и контента [4].

Также, благодаря CSS можно создать несколько вариантов дизайна страниц проекта, например разные фоны, размеры рамок или текстовой стилизации. А можно настроить изменение дизайна под разные размеры гаджетов – это называют адаптивной верстkой. Или же можно настроить проект так, чтобы при копировании или печати сразу с сайта не печатались элементы управления.

Огромный плюс для пользователей вследствие разделения HTML от CSS - быстрота загрузки страниц web-сайта. Это происходит за счёт того, что браузер загружает только структуру документа, данные хранимые на странице, а представление этих данных загружается лишь раз и кэшируется, что способствует не только быстрой загрузке страницы, но и уменьшению трафика и нагрузки на сервер.

Для разработчика невероятным плюсом является упрощение настройки дизайна. Каскадные таблицы стилей позволяют на все страницы с HTML кодом написать один файл с описанием дизайна. Поэтому, допустив ошибку в стиле,

необязательно перерывать множество строк кода HTML-страниц. Достаточно просто изменить соответствующий CSS-файл. Данная особенность веб-программирования упрощает не только исправление ошибок, но и изменение дизайна с нуля.

CSS дает возможность делать со страницей все что угодно. Достаточно прописать правильное значение селектора, что невозможно сделать атрибутами тегов HTML.

И самое важное преимущество - это повышение совместимости с разными интернет-платформами за счёт использования web-стандартов, прописанных консорциумом W3C.

1.4 Почему следует избегать атрибут style

Как было сказано ранее, CSS и HTML прошли достаточно большой путь, и неудивительно, что во время всего этого пути появлялись те, кто пытался их объединить. Одним таким способом является атрибут style. Он используется для стилизации элементов непосредственно в HTML-коде. Данный атрибут style предполагался как простой способ изменения внешнего вида практически любого HTML-элемента, это продемонстрировано в листинге 5

```
<html>
<body style="background-color:ivory;">
<h1>Посмотрите на цвета и стили</h1>
<p style="font-family:verdana;color:red;">
Этот текст написан красным цветов с помощью шрифта
Verdana.
</p>
<p style="font-family:times;color:green;">
```

```
Этот текст написан зеленым цветом с помощью шрифта
Times.
</p>
<p style="font-size:30px;">Этот текст имеет размер 30
пикселей.</p>
</body>
</html>
```

Листинг 5 – Пример кода

Если обратить внимание на листинг, то можно заметить, что с помощью атрибута style мы задаем CSS-стиль внутри HTML-элементов, данный способ задания стилей называется встроенным стилем. Встроенный стиль, в отличие от простого CSS, определяет стиль элемента, в котором он был описан, а так же стиль его дочерних элементов.

Давайте рассмотрим подробнее методы задания тех или иных элементов дизайна.

Задний фон является основным способом графического изменения сайта, задается с помощью CSS-свойства background-color, которое в качестве значения может принимать любое доступное значение цвета. После свойства обязательно должно идти двоеточие и значение, после значения обязательно ставится точка с запятой. Эти правила распространяются на все свойства атрибута style. Если нужно задать несколько свойств стиля для одного элемента, то каждое последующее свойство записывается после « ; », это продемонстрировано на листинге 6.

```
<html>
<head>
<title>HTML фон</title>
</head>
<body style="color:Yellow;>
```

```
<h1>Заголовок 1-го уровня</h1>
</body>
</html>
```

Листинг 6 - Пример кода

Цвет текста элемента задается, как и в CSS, с помощью свойства color. Так можно задать любой цвет для текстового содержимого. В качестве значения свойство color может принимать имена цветов, RGB значения или шестнадцатеричные коды.

Рассмотрим наиболее простой способ задания цвета и в атрибуте style, и в CSS. Это продемонстрировано на листинге 7.

```
<html>
<head>
<title>HTML фон</title>
</head>
<body style="color:Yellow; background-color:#66cc66">
<h1>Заголовок 1-го уровня</h1>
<p>Первый параграф</p>
</body>
</html>
```

Листинг 7 - Пример кода

Таким образом, чтобы задать цвет, нужно просто указать его название в качестве значения свойства. При этом, не имеет значения, пишите вы названия строчными или прописными буквами, поэтому можно написать yellow, Yellow или YELLOW, и все это будет работать.

Система RGB использует три числа, которые описывают относительное количество красного, зеленого и синего цветов, которые смешаны вместе для получения любого оттенка. Числа могут варьироваться от 0 до 255 для числовых значений или от 0% до 100%

Чаще всего значения красного, зеленого и синего цветов выражают числами от 0 до 255. Например, вместо 80% красного, 40% зеленого и 0% синего можно написать 204 красного, 102 зеленого и 0 синего, это продемонстрировано на листинге 8

```
<html>
<head>
<title>HTML фон</title>
</head>
<body style="color:Yellow; background-color:#66cc66">
<h1>Заголовок 1-го уровня</h1>
<p>Первый параграф</p>
<p>Второй параграф</p>
<p style="color: rgb(204, 102, 0)">Третий параграф</p>
</body>
</html>
```

Листинг 8 - Пример кода

Теперь рассмотрим чаще всего используемый и самый удобный метод задавания цвета - шестнадцатеричный код. На листинге 9 изображено, как это будет выглядеть.

```
<html>
<head>
```

```
<title>HTML фон</title>
</head>
<body style="color:Yellow; background-color:#66cc66">
<h1>Заголовок 1-го уровня</h1>
<p>Первый параграф</p>
<p>Второй параграф</p>
<p style="color:# cc6600">Третий параграф</p>
</body>
</html>
```

Листинг 9 - Пример кода

Все три метода определения цветов в CSS подходят ко всем свойствам, которые в качестве значений могут принимать цвета. Все доступные названия цветов вы можете посмотреть в таблице цветов, там сможете подобрать цвет в RGB или шестнадцатеричном формате.

Шрифты могут очень сильно влиять на дизайн страниц. В CSS они разделены на семейства, в которых вы можете выбрать, какой шрифт лучше подходит для определенного элемента на странице, это продемонстрировано на листинге 10

```
<html>
<head>
<title>HTML фон</title>
</head>
<body>
<h1>Заголовок 1-го уровня</h1>
<p font-family: Verdana, Geneva, Arial, sans-serif;>Первый
параграф</p>
```

```
<p>Второй параграф</p>
<p>Третий параграф</p>
</body>
</html>
```

Листинг 10 - Пример кода

Свойство font-family дает возможность задать список предпочтительных шрифтов, которые указываются через запятую. Если имя шрифта состоит из нескольких слов, то такое название следует заключить в двойные кавычки, например: "Courier New".

Размер шрифта очень сильно влияет на дизайн веб-страницы и читабельность ее текста. В CSS есть несколько единиц измерения, с помощью которых можно задавать размер для текста, все они подробно описаны в разделе «Единицы измерения CSS». Также имеется возможность задавать размер с помощью ключевых слов, это продемонстрировано на листинге 11

```
<html>
<head>
<title>HTML фон</title>
</head>
<body>
<h1>Заголовок 1-го уровня</h1>
<p>Первый параграф</p>
<p font-size: medium>Второй параграф</p>
<p>Третий параграф</p>
</body>
</html>
```

Листинг 11 - Пример кода

Все доступные ключевые слова, задающие размер, вы можете посмотреть в справочнике по CSS в описании свойства font-size.

Выравнивание текста в HTML-документе задаётся с помощью свойства text-align, которое позволяет выровнять текст по правой или левой стороне, а также задать выравнивание текста по ширине. Свойство text-align работает только с блочными элементами, выравнивая все строчные элементы внутри блочного, это продемонстрировано на листинге 12

```
<html>
    <body style="background-color: DarkGray; color: white;">
        <h1 style="font-family: verdana;">Заголовок</h1>
        <p style="font-size: 10px;">Очень маленький размер
                текста.</p>
        <p style="text-align: right;">Этот текст будет выровнен
                по правому краю.</p>
        </body>
    </html>
```

Листинг 12 - Пример кода

Подведем итоги. Атрибут style является некой заменой CSS, имеющий такие же правила написания кода, как и каскадная таблица стилей. Но, тем не менее, есть ряд причин, по которым данный атрибут стараются не использовать при написании сайтов.

Данный атрибут замедляет загрузку сайтов. Браузер грузит сразу структуру документа, хранимые данные, а так же стилевое оформление, что нагружает сервер и понижает посещаемость сайта.

Для поддержания web-ресурса на верхних позициях поисковой выдачи необходимо проходить индексацию, а наличие блока лишнего текста в HTML

документе усложняет этот процесс. Этого можно избежать, вынеся все стили в отдельный документ.

Так же при использовании style страдает «закоженность», отношение размера текста к общему коду страницы. Сопровождение и обновление сайта будет занимать больше времени, так как при отказе от разделения контента от стилей придется каждый раз внутри HTML-документа прописывать стилистическое оформление. Что, опять же, мешает поисковым роботам проводить индексацию страницы.

И самый большой недостаток, отпугивающий от этого способа оформления сайтов - удобочитаемость кода. Так как коммерческие проекты в большинстве своем несут в себе достаточно большой объем HTML-кода, в вопросе выбора между атрибутом style и каскадными таблицами стилей разработчики делают выбор в пользу CSS.

2 QSS

QSS в значительной части был вдохновлён каскадными таблицами стилей CSS для HTML, вследствие чего имеет похожий синтаксис.
В частности, как и в CSS, в QSS можно изменять форму, цвет, прозрачность элементов, а также визуальную реакцию на события. Стили можно применять как к отдельному виджету, так и ко всему классу, что делает QSS удобным для разработки интерфейсов приложений.

Также, Qt Designer предоставляет возможность интеграции стилей, что упрощает их тестирование и разработку. Также, при запуске приложения на Qt, можно применить к нему таблицу стилей, Механизм QSS позволяет полностью отделить визуальный дизайн от разработки приложения и привлечь к стилизации приложения веб-дизайнеров.

2.1 История QSS

Историю QSS невозможно рассматривать в отрыве от истории Qt, поэтому следует начать с самого начала. Как средство разработки Qt впервые стал известен общественности в мае 1995 года. Первоначально он разрабатывался исполнительным директором Хаарвардом Нордом и призедентом Айком Чеймб-Ингом компании Trolltech [5].

Интерес Хаарварда к графическим пользовательским интерфейсам начался с привлечения его к разработки интерфейсов на C++. А спустя несколько лет эти два норвежских программиста разрабатывали базы данных для ультразвуковых изображений, данная разработка велась под такими операционными системами как Unix, Macintosh, Windows и облегчала работу с этими системами, ибо она предоставляла графический интерфейс. Работа над данным проектом дала еще один толчок для создания объектно-ориентированной межплатформенной си-

стемы разработки графического пользовательского интерфейса. А вечные дискуссии друзей-программистов стали интеллектуальной основой для дальнейшей работы.

В 1991 году началась полномасштабная разработка классов, которые стали костяком молодого проекта. Через год была реализована еще одной гениальная идея: Айриком была предложена простая и мощная парадигма программирования GUI - использование «сигналов и слотов» - которая используется и сейчас. В 1993 году было разработано первое графическое ядро Qt, что способствовало созданию собственных виджетов.

В 1994 году была предпринята попытка выхода на рынок, но так как они не имели не законченного продукта ни заказчиков, данная попытка была неудачной. И все же 4 марта этого же года была зарегистрирована компания, которая называлась «Quasar Technologies», далее они не раз меняли имя компании, но после покупки фирмой Nokia, она была переименована в «Qt Software».

Можно так же обратить внимание на название. Заглавная буква «Q» была выбрана в качестве префикса, поскольку эта буква имела красивое начертание в шрифте Emacs, которым пользовался Хаарвард. Была добавлена буква «t», означающая «toolkit»(инструментарий).

В апреле 1995 года через посредничество одного университетского профессора, знакомого Хаарварда, норвежская компания «Metis» заключила с ними контракт на разработку программного обеспечения на основе Qt. Примерно в это же время «Trolltech» приняла на работу Арнта Гулдбрансена, которые в течение своих шести лет работы в этой компании продумал и реализовал оригинальную систему документирования, а также внёс определённый вклад в программный код Qt.

20 мая 1995 года Qt 0.90 был установлен на сайте «sunsite.unc.edu». Спустя шесть дней о выпуске этой версии было объявлено на «comp.os.linux.announce». Это была первая публичная версия Qt. Qt можно было использовать в разработках как Windows, так и Unix, причём программный интерфейс был одинаковый

на обеих платформах. С первого дня предусматривались две лицензии применения Qt: коммерческая лицензия предназначалась для коммерческих разработок, и свободно распространяемая версия предназначалась для разработок Open-source проектов. Контракт с «Metis» сохранил компанию на плаву, хотя в течение долгих 10-ти месяцев не было продано ни одной коммерческой лицензии Qt.

В марте 1996 года Европейское управление космических исследований стало вторым заказчиком Qt, которое приобрело десять коммерческих лицензий. Верящие в удачу Айрик и Хаарвард приняли на работу ещё одного разработчика. Qt 0.97 был выпущен в конце мая, и 24 сентября 1996 года вышла версия Qt 1.0. К концу этого года вышла версия Qt 1.1; восемь заказчиков — все их разных стран — приобрели в общей сложности 18 лицензий. В этом году был также основан Маттиасом Эттричем проект KDE.

Принятое Маттиасом решение по применению Qt для построения KDE помогло Qt стать фактическим стандартом по разработке на C++ графического пользовательского интерфейса в системе Linux. Маттиас присоединился к «Trolltech» в 1998 году, и последняя значимая версия Qt первого выпуска, 1.40, появилась в сентябре того же года. Qt 2 имела новую лицензию для открытого исходного кода — Q Public License(QPL), которая соответствовала определению открытого исходного кода. В августе 1999 года Qt выиграла премию журнала «Linux World» за лучшую библиотеку или инструментальное средство. Примерно в это же время была образована компания «Trolltech Pty Ltd»(Австралия).

Qt 3.0 была выпущена в 2001 году. Qt теперь работала в системах Windows, Mac Os X, Unix. Qt 3.0 содержала 42 новых класса, и объем её программного кода превышал 500 000 строк. Qt 3 представляла собой важный шаг вперёд по сравнению с Qt 2, которая, в частности, значительно улучшила поддержку локализации и кодировки Unicode, ввела совершенно новые виджеты по просмотру и редактированию текста и класс регулярных выражений, аналогичных применяемым языкам Perl. Qt 3.0 была удостоена премии «Software Development Times» в категории «Высокая продуктивность» в 2002 году.

Летом 2005 года была выпущена Qt 4.0. Имея около 500 классов и более

9000 функций, Qt 4 оказалась больше и богаче любой предыдущей версии; она была разбита на несколько библиотек, чтобы разработчики могли использовать только нужные им части Qt. Версия Qt 4 представляет собой большой шаг вперёд по сравнению с предыдущими версиями; она содержит полностью новый набор эффективных и простых в применении классов-контейнеров, усовершенствованную функциональность архитекторы модель/представление, быстрый и гибкий фреймворк графики 2D и мощные классы для просмотра и редактирования текста в кодировке Unicode, не говоря уже о тысячах небольших улучшений по всему спектру классов Qt. Qt 4 является первой версией Qt, доступной на всех поддерживаемых платформах, как для коммерческой разработки, так и для разработки с открытым исходным кодом.

После того как в 2011 году Nokia заключает договор с Microsoft об использовании Windows в создании мобильных телефонов, Nokia продает часть своих акций компании Digia, и уже через год полностью отказывается от Qt. Но уже в 2016 году The Qt Company выходит из состава Digia и становиться самостоятельной компанией.

На данный момент уже вышла версия Qt 5.9 она является самой новой, но, несмотря на нее новизну, она уже достаточно стабильна. В функции, используемые разработчиками, добавились новые свойства, расширены функции старых модулей, так же добавили новый модуль, способный отображать вход контролера-геймпада в приложение C++. Расширение модулей привело так же к изменению самой платформы: была добавлена поддержка новых операционных систем, а так же возможность оптимизировать размеры. Так же были дополнены модули для предварительного просмотра технологии, и были определены устаревшие модули.

В заключении надо отметить, что, со дня образования компании популярность Qt постоянно росла, и она продолжает расти в наши дни. Этот успех является отражением качества Qt, так и того удовольствия, которые разработчик получает при его использовании. За последнюю декаду Qt превратилась из «секретного» программного продукта, известного только группе профессионалов, в

продукт, которым пользуются по всему миру тысячи коммерческих заказчиков и десятки тысяч разработчиков приложений с открытым исходным кодом.

2.2 Синтаксис и таблицы стилей

Как было отмечено ранее, QSS очень похож синтаксически на CSS, поэтому, имея опыт работы с каскадными таблицами стилей освоить стили Qt несложно. Но тем не менее о методах написания стилей необходимо упомянуть.

QSS состоит из последовательности правил стилей. В свою очередь правило стиля состоит из селектора и декларации. Селекторы описывают, над каким объектом будет производиться действие, а в декларации описаны те самые действия. Например, чтобы задать QPushButton цвет фона красным необходимо написать «QPushButton {color:red}» [6].

Также, QSS не чувствителен к регистру, исключением является имена классов, объектов и имена свойств. Так же в этом случае можно использовать несколько имен классов одновременно, перечислив их через запятую, например: «QPushButton , QlineEdit , QcomboBox {color:red}»

Технически это будет равно коду из листинга 13.

```
QPushButton {color:red};
QLineEdit {color:red};
QComboBox {color:red}
```

Листинг 13 – Пример кода

То есть если обратить пристальное внимание к правилам написания стилей, то можно проследить схожесть двух таблиц стилей, и главное сходство здесь в оформлении кода: «Класс {селектор:описание}»

Во всех примерах используется самый простой тип селекторов – селекторы типов. Так как вдохновителем QSS была CSS, то все типы селекторов, что есть во второй, есть и в первой таблице стилей, самые распространенные селекторы продемонстрированы в таблице 1.

Таблица 1 – Типы селекторов

Селектор	**Пример**	**Пояснение**
Универсальный селектор	*	Соответствует всем виджетам.
Селектор типа	QPushButton	Совпадает элементом QPushButton и его подклассами.
Селектор свойств	QPushButton[flat="false"]	Совпадает с не плоскими QPushButton. Этот селектор используется для проверки любого свойства Qt, которое поддерживает QVariant :: toString (). Кроме того, специальное свойство класса поддерживается для имени класса. Этот селектор используют для проверки динамических свойств.
Селектор классов	.QPushButton	Соответствует QPushButton , но не относится к его подклассам, то есть он эквивалентен *[class~="QPushButton"].
ID селектора	QPushButton#okButton	Соответствует всем QPushButton которые называются okButton.
Селектор потомков	QDialog QPushButton	Соответствует всем QPushButton, которые являются потомками QDialog .
Выбор ребенка	QDialog > QPushButton	Соответствует всем экземплярам QPushButton, которые являются прямыми дочерними элементами

Зачастую приходится работать со сложными, составными виджетами. Для стилизации оных нужно получить доступ к суб-элементам управления виджета, таким как раскрывающаяся кнопка (QComboBox) или стрелки вверх и вниз (QSpinBox). Селекторы могут содержать субконтроллеры, благодаря которым можно ограничить применение правила. Например: QComboBox::drop-down { image: url(dropdown.png) } [7].

В этом правиле стилизована кнопка раскрывающегося списка всех QComboBox. Хоть синтаксис и напоминает псевдоэлементы CSS3, суб-элементы Qt концептуально отличны от них и имеют иные каскадные семантики.

Суб-элементы управления всегда позиционируются относительно других элементов, называемых эталонными. Это могут быть виджеты или другие суб-элементы управления. Например, раскрывающийся список QComboBox :: по умолчанию помещается в верхнем правом углу прямоугольника. Этот прямоугольник может быть изменен с использованием свойства subcontrol-origin. Например, если мы хотим поместить раскрывающийся список в прямоугольник поля QComboBox вместо того начального прямоугольника по умолчанию, мы можем использовать код, аналогичный написанному в листинге 14:

```
QComboBox {
    margin-right: 20px;
}
QComboBox::drop-down {
    subcontrol-origin: margin;
}
```

Листинг 14 – Пример кода

Выравнивание выпадающего списка в прямоугольнике поля изменяется с помощью свойства subcontrol-position .

Схема относительного позиционирования (position : relative) позволяет смещать положение суб-элемента управления с его исходного положения. Например, когда нажата кнопка раскрывающегося списка QComboBox , нам может потребоваться смещение стрелки внутрь, чтобы получить «нажатый» эффект. Для этого мы можем использовать код как на листинге 15:

```
QComboBox::down-arrow {
    image: url(down_arrow.png);
}
QComboBox::down-arrow:pressed {
    position: relative;
    top: 1px; left: 1px;
}
```

Листинг 15 - Пример кода

Абсолютная схема позиционирования (position : absolute) позволяет изменять положение и размер суб-элемента управления относительно эталонного элемента [8].

Селекторы могут содержать псевдосостояния, они ограничивают применения правил в зависимости от состояния виджета. Псевдосостояния указываются в конце селекторов с двоеточием. Например: QPushButton:hover { color: white }

Псевдосостояния можно отменять с помощью восклицательного знака. Например, следующее правило применяется, когда мышь не зависает над QRadioButton : QRadioButton:!hover { color: red }

Псевдосостояния могут быть связаны логическим «И». Например, следующее правило применяется, когда мышь наводится над отмеченными галочками QCheckBox: QCheckBox:hover:checked { color: white } [9].

При необходимости логическое «ИЛИ» может быть выражено с помощью запятой: QCheckBox:hover, QCheckBox:checked { color: white }.

Псевдо-состояния могут появляться в сочетании с суб-элементами управления. Например: QComboBox::drop-down:hover { image: url(dropdown_bright.png) }.

Конфликты возникают, когда несколько правил стиля определяют одни и те же свойства с разными значениями. Рассмотрим таблицу стилей на листинге 16:

```
QPushButton#okButton { color: gray }
QPushButton { color: red }
```

Листинг 16 – Пример кода

Оба правила соответствуют вызываемым экземплярам QPushButton и для свойства «цвет» есть конфликт. Чтобы его разрешить, должно учитывать специфику селекторов. В приведенном выше примере QPushButton#okButton считается более конкретным, чем QPushButton потому, что он обычно относится к одному объекту, а не ко всем экземплярам класса.

Аналогично, селекторы с псевдосостояниями более специфичны, чем те, которые без них. Таким образом, в таблице стилей на листинге 17 указано, что QPushButton должен иметь белый текст, когда мышь курсирует над ним, в противном же случае – красный.

```
QPushButton:hover { color: white }
QPushButton { color: red }
```

Листинг 17 – Пример кода

Рассмотрим пример кода из листинга 18:

```
QPushButton { color: red }
QAbstractButton { color: gray }
```

Листинг 18 – Пример кода

Оба правила применяются к экземплярам QPushButton (поскольку QPushButton наследует QAbstractButton) и существует конфликт для свойства цвета. Поскольку QPushButton наследует QAbstractButton, может возникнуть соблазн предположить, что QPushButton специфичнее, чем QAbstractButton. Однако в этом случае все селекторы типов имеют одинаковую специфичность, и правило, которое появляется последним, имеет приоритет. Другими словами, все кнопки будут серыми. Для иного результата надо менять порядок правил.

Для определения специфичности правила таблицы стилей Qt следуют спецификации CSS2:

Специфика селектора рассчитывается следующим образом:

-подсчитайте количество атрибутов идентификатора в селекторе, это будет переменная «a»;

-подсчитайте количество других атрибутов и псевдоклассов в селекторе, это переменная «b»;

-подсчитайте количество имен элементов в селекторе, это «c».

Совмещение этих трех переменных даст необходимую специфичность. Примеры представлены в таблице 2.

Таблица 2 – Примеры определения специфичности

#x34y {}	a=1	b=0	c=0	specificity = 100
H1 + *[REL=up] {}	a=0	b=1	c=1	specificity = 11

Таблицы стилей могут быть для QApplication, родительских виджетов и для дочерних виджетов. Итоговая таблица стилей произвольного виджета получается путем слияния таблиц стилей, установленных на предках виджетов, а также таблицы стилей, установленной в QApplication.

Для рассмотрения распределения приоритетов рассмотрим следующий пример.

Сперва пишем: qApp-> setStyleSheet ("QPushButton {color: white}");

Затем мы устанавливаем таблицу стилей объекта QPushButton: myPushButton-> setStyleSheet ("* {color: blue}");

Таблица стилей на QPushButton заставляет QPushButton (и любой дочерний виджет) иметь синий текст, несмотря на более конкретный набор правил, предоставляемый таблицей стилей всего приложения.

Результат был бы таким же, если бы мы написали: myPushButton-> setStyleSheet («цвет: синий»); за исключением того, что если у QPushButton были потомки (что маловероятно), таблица стилей бы их не изменила.

Все же есть одно весовое отличие QSS от CSS. В таблицах стилей Qt нет наследования. То есть если в CSS можно лишний раз не прописывать название шрифта или цвета, они автоматически наследуются у родителей, то в Qt если не прописать шрифт он будет системным. Следовательно, что бы произошло наследование необходимо написать qApp - > setStyleSheet ("QGroupBox, QGroupBox * {color: red;}") [10].

3 Концепция и логическая структура программы

В ходе выполнения курсового проекта была написана программа, в первую очередь предназначенная для демонстрации вариантов настройки индивидуального пользовательского интерфейса. Данный программный продукт способен при нажатии на кнопку менять свой внешний вид на ранее заданный стиль, что значительно упростит работу веб-дизайнерам и людям, представляющим программы заказчикам. Ведь благодаря возможности сменить стиль в одно нажатие можно предоставить несколько вариантов интерфейса, а так же примеры внешнего вида определенного программного обеспечения для слабовидящих или же людей с другими отклонениями. Данная демонстрационная программа имеет открытый код, что позволяет менять ее под нужды пользователей.

Для разработки программы был использован Qt Creator версии 4.4.1 основанный на Qt 5.9.2.

В программе на форме находиться стандартные графические элементы. Все представленные ниже элементы графического интерфейса статические. Но, тем не менее, с каждым из элементов на рисунке 1 можно взаимодействовать.

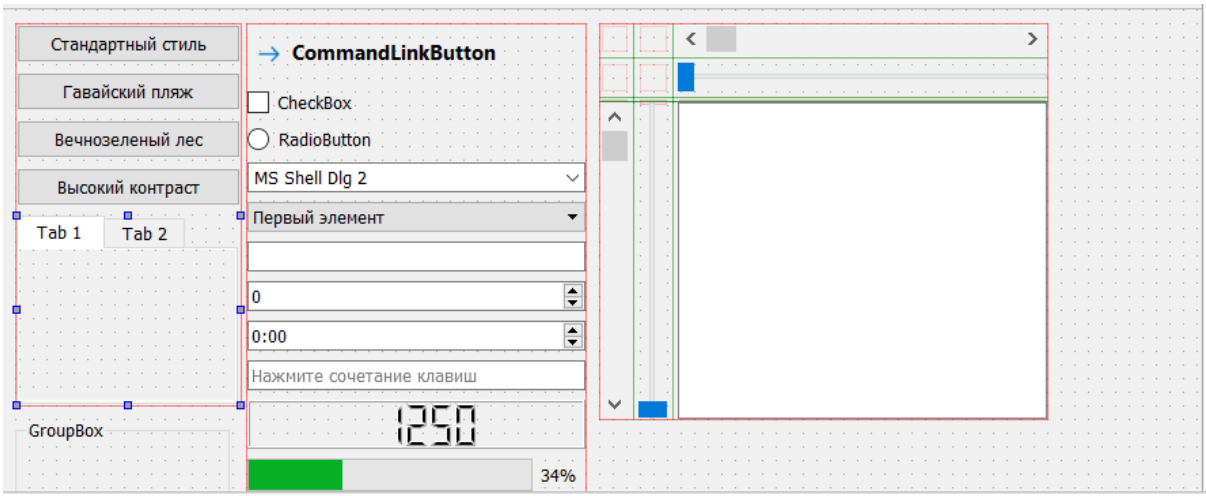

Рисунок 1 - Изображение формы с используемыми элементами

Все элементы, представленные выше, по умолчанию уже имеют свой стандартный стиль, это видно на рисунке 2. И, в принципе, ранее этого хватало для того что бы пользователь был доволен. Сейчас же, когда пользователь избалован красотой интерфейсов, проект со стандартными стилями не выигрышен.

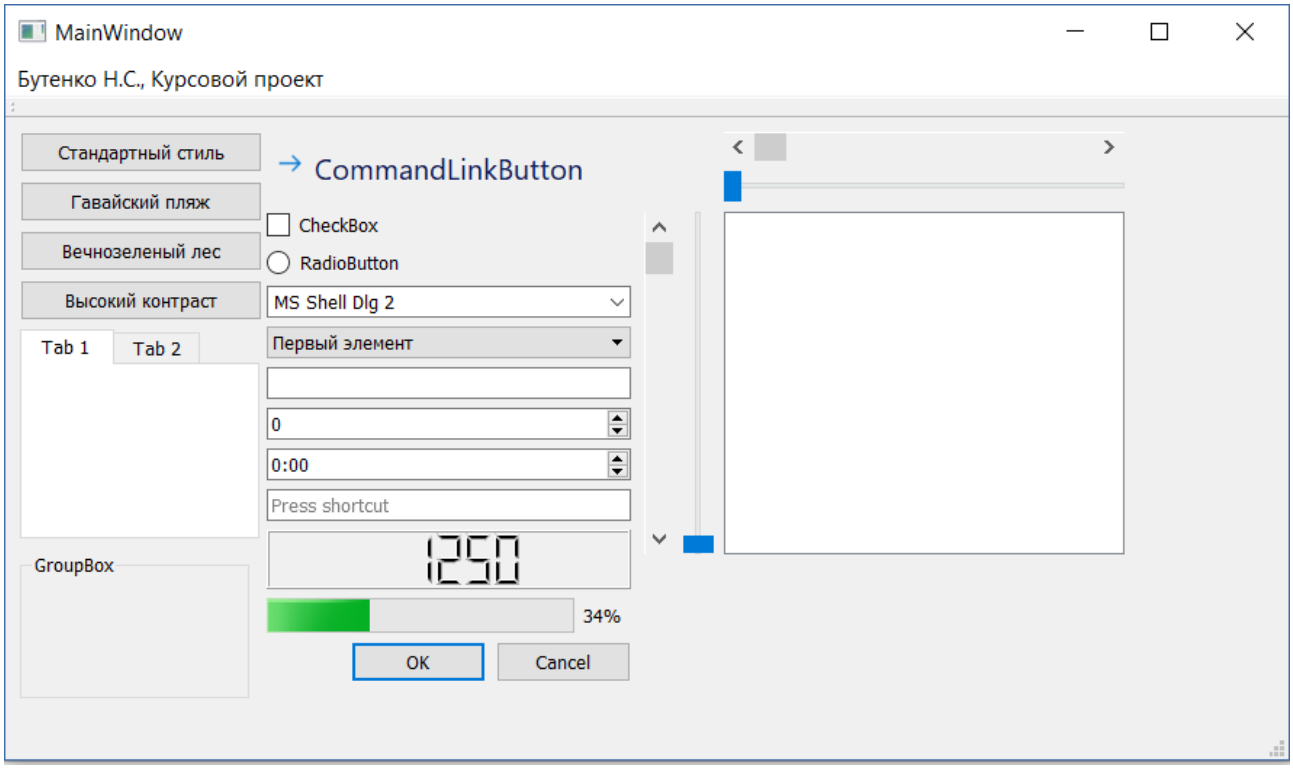

Рисунок 2 - Стандартные стили QSS

Чтобы сделать проект более красочным и интуитивно понятным, необходимо добавить стилей. Присваивать стили можно как прикладывая к каждому элементу свой qss-файл, так и одним махом – единственным стилевым файлом для всей программы. В данном курсовом проекте представлен способ определения стиля для всех элементов одним файлом.

Так как был выбран самый удобный способ изменения стилей, разобраться в логической структуре приложения не составит труда. Само присвоение стиля элементу можно увидеть на листинге 19.

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "QFile"
#include "QString"
#include "QLatin1String"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
 void MainWindow::on_pushButton_2_clicked()
{
    QFile styleF("Untitled-2.qss");
    styleF.open(QFile::ReadOnly);
    QString styleSheet2 = QLatin1String(styleF.readAll());
    qApp->setStyleSheet(styleSheet2);
     }
```

Листинг 19 – Пример кода

В выше представленном листинге можно увидеть часть кода файла main-window.cpp. В нем реализуется подробный набор действий для присвоения кнопке под названием pushButton_2 стилевого файла с именем Untitled-2.qss.

В файле с именем Untitled-2.qss хранится стиль под названием «Гавайский пляж». В данном стиле представлены способы создания выпуклых кнопок и закругленными углами. Так же все виджеты имеют контрастную по отношению к самому полю окантовку. То, как выглядит программа с примененным данным стилем, отображено на рисунке 3, а способ написания стиля для некоторых виджетов отображён на листинге 20.

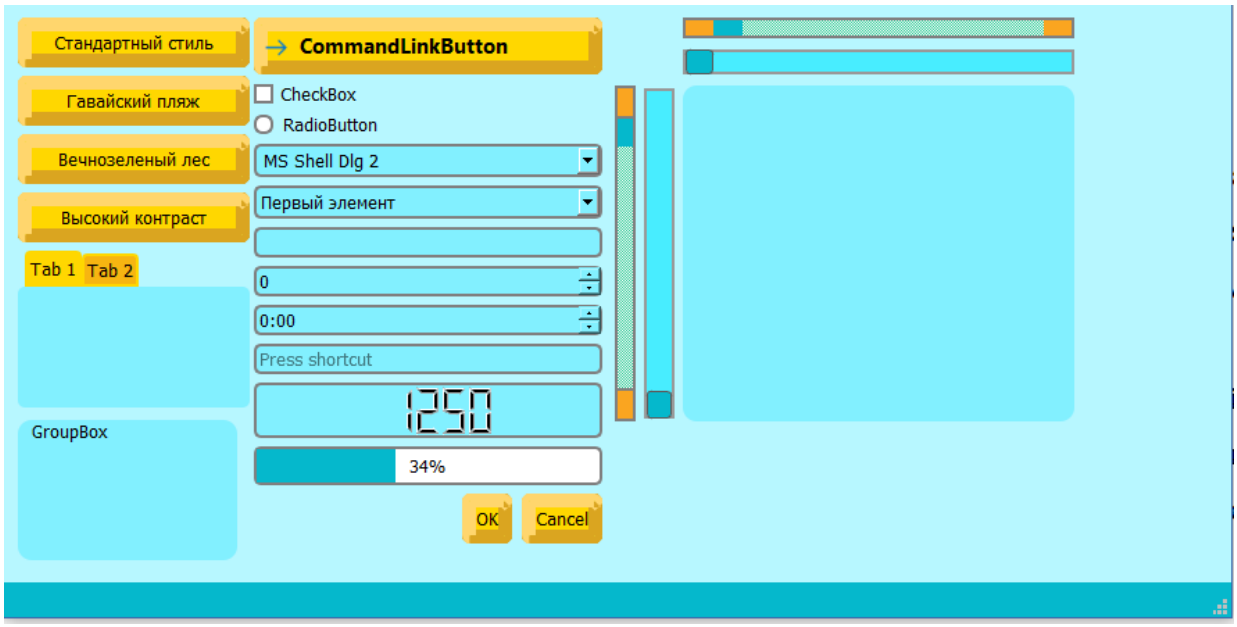

Рисунок 3 – Стиль «Гавайский пляж»

```
QPushButton{
 background-color:#FFD700;
 border-style: outset;
 border-width: 10px;
 border-color: #DAA520;
 border-radius: 6px
}
QPushButton:pressed {
 background-color:#DAA520;
```

```
 border-style: outset;
 border-width: 10px;
 border-radius: 6px;
 border-color: #B8860B
}
QLineEdit {
 background-color:#FFD700
}
#centralWidget{
 background-color:#b7f7ff
}
QProgressBar {
     border: 2px solid grey;
     border-radius: 5px;
     text-align: center;
}
QProgressBar::chunk {
     background-color: #05B8CC;
     width: 20px;
}
 QStatusBar {
     background: #05B8CC;
}
```

Листинг 20 - Пример кода для стиля «Гавайский пляж»

Все остальные стили, названия которых можно увидеть на рисунке 1, присоединяются аналогичным образом, как и «Гавайский стиль».

ЗАКЛЮЧЕНИЕ

В ходе выполнения данного курсового проекта были выполнены все поставленные цели и задачи. А именно: изучены технологии CSS и QSS, стандартные способы их применения и история, а также разработано приложение, предназначенное для демонстрации возможностей тюнинга пользовательского интерфейса.

Один за другим, светила IT-индустрии приходят к выводу, что уделять внимание таким вещам, как User Interface и User Experience важно и нужно, и недостаточно расти лишь наращивая вычислительные мощности. Технологии должны быть доступными и приятными глазу. Рамок технического характера у дизайнеров и художников с каждым годом становится всё меньше, и можно без труда наблюдать, насколько приятнее и удобнее стало взаимодействие со всевозможными интерфейсами сейчас относительно прошлых лет. Перспективы развития данной области ограничены лишь человеческим воображением.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дунаев В. В. HTML, скрипты и стили / В. В. Дунаев. – СПб.: БХВ-Петербург, 2006. – 813 с.

2 Дакетт Д. HTML и CSS. Разработка и дизайн веб-сайтов / Д. Дакетт. – М.: Эксмо, 2017. – 480 с.

3 Макфарланд Д. С. Новая большая книга CSS / Д. С. Макфарланд. – СПб.: Питер, 2017. – 720 с.

4 Хрусталев А. HTML5 + CSS3. Основы современного WEB-дизайна / А. Хрусталев, А. Кириченко. – СПб.: Наука и техника, 2018. – 352 с.

5 Eirik Eng, Matthias Ettrich, Interview by Laurent Rathle // KDE France. – 2003. (Engl). – URL: https://dot.kde.org/2004/04/12/interview-trolltechs-eirik-eng-and-matthias-ettrich.html [12 April 2004]

6 Шлее М. Qt 5.3. Профессиональное программирование на C++ / М. Шлее. – СПб.: БХВ-Петербург, 2015. – 928 с.

7 Матренин П. Введение в кроссплатформенное программирование на C++ в Qt / П. Матренин. – М.: Нобель Пресс, 2013. – 64 с.

8 Саммерфилд М. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++ / М. Саммерфилд. – СПб.: Символ-Плюс, 2018. – 560 с.

9 Бланшет Ж. Qt 4. Программирование GUI на C++ / Ж. Бланшет, М. Саммерфилд. – СПб.: КУДИЦ-Пресс, 2008. – 718 с.

10 QT Official Documentation // Espoo: The QT Company. – 2018. – QT 5.9 Official Documentation. – (Engl.). – URL: http://doc.qt.io/ [31 May 2017].