АДАПТАЦИЯ WEBGL STREAMING ДЛЯ СОЗДАНИЯ МНОГОПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ

В.В. Козлов, Г.А. Приставка

Самарский государственный технический университет, Самара, Россия

Обоснование. Quick WebGL — это подключаемый модуль платформы, который обеспечивающий однопользовательский удаленный доступ путем потоковой передачи пользовательских интерфейсов Qt Quick по сети. Пользовательский интерфейс отображается в клиентском браузере с поддержкой WebGL. Любое приложение Qt Quick можно запустить с помощью подключаемого модуля платформы webgl. Вариантом использования служит удаленное обучение или удаленное обслуживание в промышленном сценарии, когда из браузера можно удаленно отображать и управлять указателем мыши на встроенном устройстве HMI. Основной недостаток подключаемого модуля webgl заключается в том, что в рамках одного процесса возможно запускать свое приложение только удаленно или локально с подключением только одного клиента.

Цель — показать возможность применения плагина Qt WebGL Streaming для построения многопользовательских веб-приложений.

Методы. WebGL Streaming [1] — это плагин, который позволяет транслировать приложения Qt по сети, и, таким образом, они могут быть отображены на HTML5. На практике это означает, что вы можете запустить приложение на удаленном хосте и отобразить его графический интерфейс в локальном веб-браузере. Для этого не нужно вносить какие-либо изменения в исходный код, следует только запустить приложение при помощи специальной команды.

Само приложение не запускается внутри веб-браузера. Браузер отображает только графический интерфейс. Таким образом, это не потоковое видео и не зеркалирование. Речь идет о «развязке» графического интерфейса приложения и его отображении.

Поскольку данный плагин предназначен только для OpenGL (ES) [2], потоковая передача WebGL не работает с виджетами или любыми другими элементами, не относящимися к OpenGL.

Основные недостатки плагина:

– высокая требовательность к пропускной способности интернет-подключения;
– возможность запустить только для одного пользователя.

В связи с выявленными недостатками были поставлены следующие задачи:

– рассмотреть возможные способы использования плагина в локальной сети;
– рассмотреть возможные реализации многопользовательского режима;
– разработать приложение, позволяющее расширить функционал WebGl.

В ходе исследования был проведен сравнительный анализ веб-приложения и WebGl Streaming и определены ситуации для предпочтительного использования WebGl Streaming.

Поскольку все дело в веб-браузере, можно просто взять обычный веб-сервер и создать веб-приложение — результат будет почти таким же: бэкэнд размещается на удаленном устройстве, а графический интерфейс на основе HTML отображается в веб-браузере.

Действительно, в некоторых случаях достаточно просто иметь простой REST API, особенно если вам нужно получить только некоторые значения текстовых данных. Поэтому вполне вероятно, что приложение на основе Qt с потоковой передачей WebGL будет излишним для такой цели.

Однако, в более сложных сценариях, в Qt-приложения с WebGL-потоковый интерфейс может соответствовать лучше, потому что таким образом вы получаете мощный бэкэнд (c++/Qt), и привлекательный и производительный frontend. Его реализация проще, чем в случае с HTML/CSS или JavaScript.

Наиболее очевидный вариант использования потоковой передачи — возможность использовать приличный графический интерфейс для некоторых бюджетных устройств с ограниченной вычислительной мощностью, без графического процессора и довольно часто вообще без дисплея. Например, это распространенный сценарий для домена промышленной автоматизации, где вы можете установить множество «безголовых» устройств по всему заводу: они могут быть распределены по довольно значительной площади или даже

установлены в местах с опасной средой. Следовательно, возможность удаленного управления настройки этих устройств довольно удобна.

Другой возможный вариант использования — мера по борьбе с пиратством. Допустим, вы хотите защитить свое программное обеспечение от «взлома» или «пиратства».

Очевидно, что если на клиенте ничего не работает, то нечего взламывать, поскольку ваши пользователи имеют только графический интерфейс в своих браузерах, а само приложение работает на вашем сервере. В отличие от веб-приложений, где часть работы выполняется на клиенте (запросы к серверу).

Классическое веб-приложение выполняется по клиент-серверной архитектуре с браузером в роли универсального клиента. При этом предполагается, что одно серверное приложение обслуживает множество клиентов в режиме «запрос-ответ». Проблемой использования плагина Qt WebGL Streaming в многопользовательском режиме становится его изначальная ориентация на режим «одно приложение – один пользователь».

Единственная найденная концепция, не сильно загружающая систему — это запуск приложений на различных портах сервера. Для реализации предложенной концепции модуль Qt Network предлагает классы, позволяющие писать TCP/IP-клиенты и серверы, такие как QTcpSocket, QTcpServer и QUdpSocket, которые представляют концепции сети низкого уровня. Перед началом передачи данных необходимо установить TCP-соединение с удаленным хостом и портом. После установления соединения IP-адрес и порт однорангового узла доступны через определенные команды. В любое время одноранговый узел может закрыть соединение, и передача данных немедленно прекратится. QTcpSocket работает асинхронно и выдает сигналы для сообщения об изменениях состояния и ошибках.

Изначально прописывается пул портов, которые следует использовать для потоковой передачи.

Таким образом, каждый процесс должен обслуживать один порт с потоковой передачей Qt WebGL. Запуск приложения на портах осуществляется автоматически при открытии новой вкладки (подключении нового клиента) и останавливается через короткий промежуток времени после закрытия. Ниже приведен фрагмент кода, демонстрирующий данный алгоритм.

```
void ProxyServer::incomingConnection(qintptr handle) {
    clientHTTP *c=new clientHTTP(this);
    if(!c->init(handle)) c->deleteLater();
}
```

Метод «init» осуществляет запуск прописанного приложения для каждого порта:

```
this->port=port;
QString s1=QString("%1/%2").arg(port->path).arg(port->app);
QString s2="-platform";
QString s3=QString("webgl:port=%1").arg(port->port);
app.start(s1,
                { s2, s3 },
                QIODevice::NotOpen);
bool isStarted=app.waitForStarted();
if(!isStarted) return false;
internal.connectToHost("127.0.0.1",port->port);
isInternalConnected=internal.waitForConnected();
if(!isInternalConnected) {
     app.terminate();
     return false;
}
```

Результаты. Была создана реализация многопользовательского режима для плагина WebGl Streaming, основанная на запуске параллельных процессов. Программа включает в себя также:

– работу с кэшем приложения;
– работу с Cookie;
– архивирование посылаемых данных.

В прошлом решение о разработке приложения, которое может работать в веб-браузере, обычно означало использование принципиально веб-приложения. Предполагая, что плагин WebGL со временем будет

развиваться и использоваться чаще, такой подход станет стабильным и полным, и в определенных моментах не будет уступать веб-приложениям по ресурсозатратности.

Выводы. В данной работе была доказана возможность применения плагина Qt WebGL Streaming для построения многопользовательских веб-приложений, обладающих малой ресурсозатратностью и высоким быстродействием.

Ключевые слова: веб-приложение; плагин; WebGL Streaming; браузер; TCP/IP; клиент; порт.

Список литературы

1. resources.qt.io [Электронный ресурс]. Remote UIs with WebGL and WebAssembly // The Qt Company [дата обращения: 01.04.2022]. Доступ по ссылке: https://resources.qt.io/industry-solution-automation/remote-uis-with-webgl-and-webassembly
2. khronos.org [Электронный ресурс]. OpenGL ES Overview // The Khronos Group [дата обращения: 01.04.2022]. Доступ по ссылке: https://www.khronos.org/api/opengles

Сведения об авторах:

Глеб Алексеевич Приставка — студент, группа ГИП-119, институт автоматики и информационных технологий; Самарский государственный технический университет, Самара, Россия. E-mail: pristavka.gleb@yandex.ru

Вячеслав Васильевич Козлов — научный руководитель, кандидат технических наук; заместитель заведующего кафедрой информационных технологии, Самара, Россия. E-mail: vco2005@mail.ru