МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
БУРЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Институт математики и информатики
Кафедра прикладной математики

Выпускная квалификационная работа

Кроссплатформеное приложение для 3D-визуализации 2D-модели плана помещения методом бросания лучей

Выполнил : студент 4 курса группы 05230
Шорников Александр Евгеньевич
Научный руководитель: к.ф.-м.н., ст. преп.
Трунин Дмитрий Олегович
Научный консультант: асс. каф. ИТ
Брагин Александр Фёдорович

Улан-Удэ
2017

Оглавление

Стр.

ВВЕДЕНИЕ

Интерактивные планы помещений - доступный и современный способ разобраться в незнакомом здании. Сегодня интерактивная карта в отличие от обычной плоской в виде изображения отличается тем, что ее элементами можно управлять. Пользователь, находясь на странице, может свободно перемещаться по карте, находить объекты, схему прохода и просматривать информацию. При необходимости элементы карты могут включать в себя помимо реальных физических объектов (дома, улицы, парки, дороги и т.д.) дополнительно текстовую информацию, видеозаписи и ссылки на сайты.

К таким интерактивным планам предъявляются весьма суровые системные требования: поскольку они предназначены для самых разных платформ, и должны запускаться не только на можных десктопах, на и на мобильных устройствах, слабых нодах типа Raspberry Pi, встраивымх системах типа Arduino, а так же крутиться на сайтах. Не на всех из вышеперечисленых платформ может запускаться стандартная для современных 3D-визуализаций библиотека OpenGL(или WebGL в случае сайтов).

Для построения трехмерной визуализации выбран метод бросания лучей. Данный метод, являясь одним из простейших для трехмерной отрисовки, имеет ряд свойств согласующихся с задачами проекта: малую вычислительную сложность и простую реализацию без использования готовых библиотек трехмерной графики. Плюсами данного метода так же является то, что метод работает с плоской двумерной моделью помещения. То есть для визуализации помещения не надо строить полностью трёхмерную карту для рендеринга - достаточно дать плоскую карту.

Цель: создание кроссплатформеного псевдотрёхмерного движка для

3D визуализации здания по 2D плану.

Задачи:
–Модификация алгоритма рейкастинга для вещественных координат;
–Разработка математической модели для рейкастового рендерера;
–Изучение и освоение технологии кросс-компиляции Cheerp;

Объектом исследования является задача построения псевдотрёхмерной картинки.

Предметом исследования является изучение основных принципов построения псевдотрёхмерной картинки методом бросания лучей.

Глава 1

Постановка задачи

Итак, перед нами встала задача: создание интерактивного плана помещения методом бросания лучей. Поскольку современный человек предпочитает универсальные механизмы для любых платформ, мы так же озадачили себя кроссплатформеностью одного и того же кода, причём на таких казалось бы несовместимых платформах как web и нативная платформа(десктоп и мобильное приложение). Для достижения кроссплатформы на любой десктопной операционной системы, в разработке был применён язык $C++$‘14 с фреймворком Qt 5.8.0. Использование именно этой среды и языка обеспечило полную совместимость как в $*nix$ системах (в частности, в $MacOS\ X$, системах на основе ядра $Linux$, BSD - системах и тд), так и в среде $Windows$. Так же фреймворк Qt позволяет с лёгкостью перекомпилировать тот же самый код без особых изменений на любое $Android$ - устройство и устройство с iOs, например $iPhone$, что весьма актуально ввиду всеобщей распространённости таких гаджетов.

Однако, с платформой Web для встраивания на сайты пришлось повозиться. Любая web-страничка в Интернете представляет собой совокупность документа $HTML$, каскадную страницу стилей CSS, скриптовую часть на языке $JavaScript$ и внутренее содержание в виде текста, документов содержащих в себе картинки, аудиоокнтент, видеоконтент, мультимедиа, апплеты и гиперссылки на другие страницы. И для реализации на сайте нашего приложения необходимо было создать скриптовый апплет. Ввиду абсолютной несовместимости web-платформы с языком $C++$ в качестве скриптового языка для встраиваемых апплетов, необходимо было перенести реализацию на язык $JavScript$ с сохранением работоспособности десктопной версии.

Для решения этой проблемы мы разделили кодовую базу, сделав общей часть непосредственно решающую задачу, и отделив код специфичный для каждой из платформ. Код решающий задачу оперирует абстрактными классами канвы ($Core :: Canvas$) и буфера ($Core :: ImageBuffer$), которые в свою очередь конкретизируются для каждой платформы.

В итоге, в общую часть у нас вошло: Алгоритм рейкастинга, Представление карт уровней, Абстракции канвы и буфера кадров. И части для каждой платформ вошли конкретные реализации канвы и буфера кадров. Сейчас специальная часть для платформ занимают лишь 30% от общей кодовой базы, и нет причин для увеличения этой части с ростом проекта.

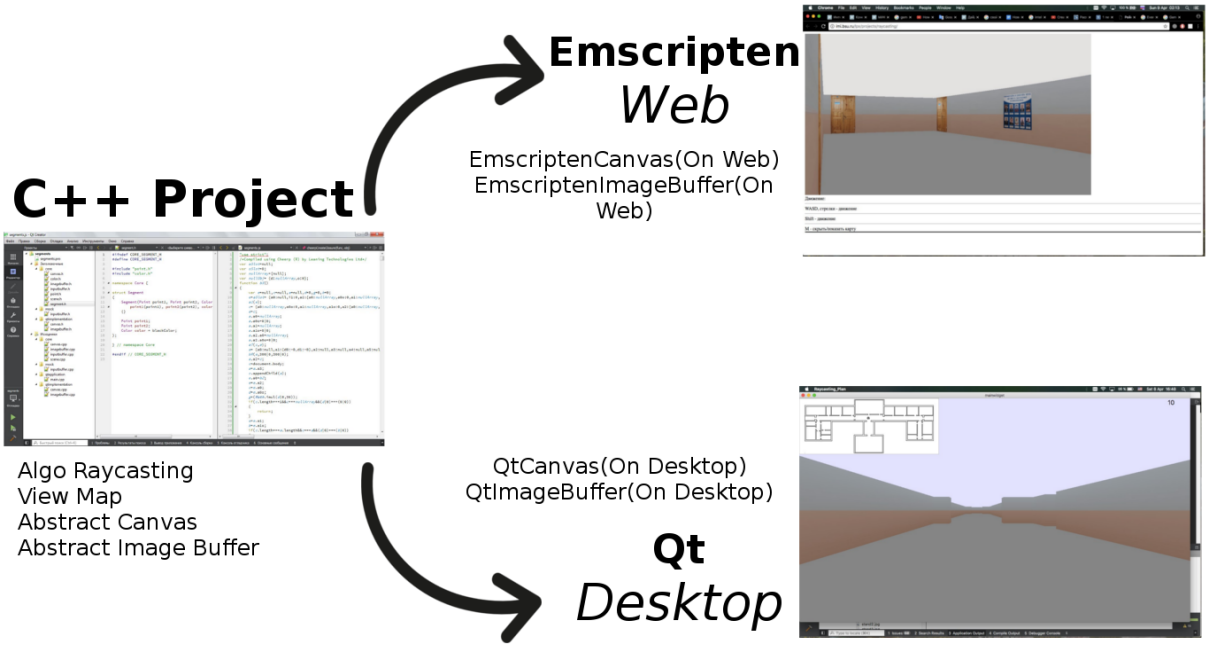

Рис. 1.1: Разделение конкретных реализаций

Для переноса общей части на *JavaScript* был использован кросс-компилятор $C++$ в *JavaScript* под названием *Emscripten*. *Emscripten* — ком-

пилятор из $LLVM$ байт-кода в $JavaScript$. $C/C++$ код может быть скомпилирован в $LLVM$ байт-код с помощью компилятора $Clang$. Некоторые другие языки так же имеют компиляторы в $LLVM$ байт-код. $Emscripten$ на основе байт-кода генерирует соответствующий $JavaScript$-код, который может быть выполнен любым интерпретатором $JavaScript$, например современным браузером. $Emscripten$ предоставляет: $emconfigure$ – утилита настройки окружения и последующего запуска $./configure$; $emmake$ – утилита для настройки окружения и последующего запуска $make$; $emcc$ – компилятор $LLVM$ в $JavaScript$.

Для автоматической сборки проекта и компиляции использована утилита $CMake$. $CMake$ - это универсальная кроссплатформенная утилита для автоматической сборки программы из исходных кодов. При этом сама $CMake$ непосредственно сборкой кода не занимается, а выступает в качестве front-end'а для back-end компилятора. И в итоге для общей сборки проекта необходим скрипт сборки для $CMake$, который выглядит следующим образом:

```
cmake -G"MinGW Makefiles" -DCMAKE_TOOLCHAIN_FILE=
C:\cheerp\share\cmake\Modules\CheerpToolchain.cmake
../segments && mingw32-make
```

Подобного рода скрипты позволяют скомпилировать проект и сразу в нативную версию, и в версию для web-приложения с условием готового платформозависимого канваса и буффера кадров для странички $HTML$. Выходной $JavaScript$-файл встраивается на заранее установленную страничку автоматически.

Глава 2

Математическая модель

2.1. Способ представления карты

Посколь