Министерство образования Республики Беларусь
УО «Полоцкий государственный университет»

Кафедра технологий программирования

О Т Ч Е Т

о прохождении преддипломной практики
студентки 4 курса 15-ИТ-1 группы

Стеняева Андрея Дмитриевича

в период с 25.03.2019г. по 19.04.2019г.

Наименование базы практики: УО «Полоцкий государственный университет»

Руководитель от кафедры:
(ученая степень, звание)

Макарычева В. А.
Преподаватель-стажёр

Практика защищена
с оценкой _____
Дата защиты _____

Полоцк 2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данный отчёт является результатом прохождения преддипломной практики в УО «Полоцкий государственный университет». Основная цель прохождения преддипломной практики – сбор информации о предметной области дипломного проекта, выбор технологий и инструментов для продуктивной и качественной разработки программного обеспечения, расширению и систематизации знаний в области программирования и проектирования программного обеспечения, полученных в процессе изучения специальных дисциплин. При проектировании и разработке программного обеспечения необходимо задействовать личный опыт проектирования и разработки программного обеспечения, а также заимствовать опыт руководителя и его коллег для решения задач, на решение которых не хватает опыта. Также во время прохождения практики необходимо ознакомиться с организационной структурой предприятия, функциональными обязанностями подразделений базы практики, степенью автоматизации предприятия, а также с услугами, оказываемыми предприятием.

В процессе прохождения преддипломной практики необходимо решить следующие задачи:

- проанализировать предметную область;
- произвести сравнительный анализ аналогов и прототипов разрабатываемого программного продукта;
- выбрать средства и методы решения задачи;
- разработать алгоритм решения задачи;
- выполнить проектирование программного обеспечения.

Объектом исследования являются шейдерные программы.

Предметом исследования является процесс разработки шейдерных программ.

1 ХАРАКТЕРИСТИКИ ОРГАНИЗАЦИИ

1.1 История развития, общие сведения о предприятии

Полоцкий государственный университет был основан в 1968 году и был известен как «Новополоцкий политехнический институт», а своё название приобрёл только в 1993 году. С момента основания велась подготовка специалистов для развития двух химических заводов-гигантов: Нафтан и Полимир. В настоящие время университет готовит специалистов в двадцать одном направлении образования. Будущих специалистов учат осваивать новое в условиях неопределенности, решать нестандартные задачи, работать в команде, быть готовым к экстремальным нагрузкам XXI века, ценить время, помнить свои корни, любить Беларусь.

Полоцкий государственный университет является интернациональным учреждением образования. Это значит, что в стенах этого университета могут обучаться студенты из любой точки мира. На текущий момент в стенах университета обучаются более 400 иностранных студентов из 24 стран. Университет активно принимает участие в международных образовательных программах:

- подготовка магистров по программе «Civil engineering» на английском языке с выдачей двух дипломов совместно с Политехническим институтом г. Лейри;
- бизнес-администрирование «Региональное планирование и развитие» совместно с Резекненской технологической академией, г. Резекне;
- управление проектами» совместно с Университетом Домброва-Гурнича.

1.2 Организационная структура

За управление развитием Полоцкого государственного университета отвечает ректорат университета, во главе которого стоит ректор университета. Ректор обеспечивает общее управление университетом в пределах своих полномочий в соответствии с Уставом университета и законодательством Республики Беларусь. Первый проректор осуществляет оперативное управление университетом, координирует работу проректоров и отвечает за состояние и развитие кадрового потенциала.

Организационная структура университета состоит из нескольких самостоятельных структур. Бухгалтерия отвечает за управление финансами университета. Отдел документационного обеспечения отвечает за хранение документов в архиве. Отдел кадров отвечает за набор персонала и управление им. Отдел международных связей ведет

комплексную работу по укреплению репутации Университета на международных мероприятиях, содействует в установлении и поддержании контактов с международными образовательными институтами и организациями, координирует участие университета в международных научно-образовательных программах и проектах международного сотрудничества, содействует приглашению иностранных специалистов для обмена опытом, а также оказывает языковую поддержку во время встреч и сопровождения иностранных делегаций. Отдел ремонта средств вычислительной техники и радиоэлектронных устройств занимается ремонтом и техническим обслуживанием компьютерного, периферийного и электронного оборудования университета по заявкам сотрудников. Профком защищает права студентов и преподавателей. Редакционно-издательский отдел осуществляет организацию и осуществление редакционно-издательского и полиграфического процессов с целью издания учебной, учебно-методической литературы, отвечающей требованиям государственного образовательного стандарта, а также выпуск научной, справочной, и других видов литературы, рекламной, бланочной и прочей продукции в интересах обеспечения учебного, воспитательного процессов, научно-исследовательских работ. Центр информационных технологий решает задачи внутриуниверситетской информатизации (выполнения научно-исследовательских и других работ в области обеспечения информационными технологиями и ресурсами учебного процесса, научной и административно-хозяйственной деятельности университета), межвузовской деятельности в области информационных технологий и ресурсов, международной деятельности в области информатизации. Учебно-методический отдел отвечает за разработку внутриуниверситетской нормативной базы, обеспечение учебно-методической документацией и контроль за ее ведением, разработку документации для открытия новых специальностей, специализаций, организацию мониторинга состояния учебного процесса, интенсификацию учебного процесса путем распространения новых технологий и методов обучения, оптимизации графиков учебного процесса и расписания занятий.

2 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧ

2.1 Предметная область

Растеризация – процесс, в результате которого получается растровое изображение. В компьютерной графике растеризация может быть выполнена двумя способами:

— Software – цвет каждой точки результирующего изображения вычисляется на центральном процессоре;

— Hardware acceleration (аппаратное ускорение) – цвет каждой точки вычисляется на отдельном устройстве – графическом ускорители.

С распространение и удешевлением видеокарт аппаратное ускорение стало активно применяться в компьютерных играх для достижения более качественных и реалистичных результатов с максимально возможной скоростью. Для этого часть работы по построению изображения перекладывалась с процессора на видео ускоритель. В конце 2000 года компания Microsoft осуществила релиз нового API – DirectX 8.0. Основным нововведением нового API стало понятие «Шейдерная модель» – новый подход к разработке, при котором программисту давалась возможность самостоятельно определить поведение видео ускорителя на каждом этапе построения конечного изображения. Для получения желаемого результата программисту предлагается написать шейдер, который затем будет выполняться на виде ускорители и обрабатывать данные так, как это было задумано самим программистом.

Шейдер – специальная программа, экземпляры которой выполняются на графическом процессоре параллельно. На сегодняшний день существует несколько видов шейдеров, которые выполняются на разном этапе в графическом конвейере:

- Vertex shaders – шейдер, в котором происходит обработка вершин геометрии;
- Geometry shaders – шейдер, в котором происходит генерация новых примитивов на основе уже существующих вершин;
- Fragment shaders – шейдер, в котором высчитывается итоговый цвет каждого фрагмента в итоговом изображении;
- Tessellation shaders – шейдер, который используется для изменения сетки объекта в зависимости от определённых условий, которые определяются в этом типе шейдеров;
- Compute shaders – используются для вычисления информации любого желаемого типа, например, анимации или особой формы освещения.

Основными языками программирования шейдеров являются GLSL и HLSL. GLSL используется чаще всего совместно с программами, которые для построения изображения используют OpenGL API. HLSL чаще всего используется в программах, использующих DirectX API. Оба языка могут использоваться для написания шейдеров с последующей их компиляцией для использования совместно с Vulkan API.

OpenGL и Vulkan являются кроссплатформенными API, благодаря чему они могут использоваться на любой платформе без необходимости полностью переписывать программный код. Для OpenGL существует несколько разных спецификаций: OpenGL, OpenGL ES, WebGL и другие. Отличия между ними минимальны и основные подходы к разработки очень сильно похожи, однако есть и различия, определяемые спецификой целевой платформы. DirectX является разработкой компании Microsoft, поэтому данный API разрабатывается только для операционной системы Windows.

Визуальное программирование – способ создания программ путём манипулирования графическими объектами вместо написания её текста. Преимуществом является визуальная составляющая данного подхода, так как глядя на графические объекты чаще всего проще понять логику программы.

2.2 Постановка задач и характеристика объекта проектирования

Задача разрабатываемой программы – предоставить инструмент разработки шейдерных программ, использующий механизм связанных узлов для представления логики, без необходимости писать код программы вручную. Кроме того, программа должна быть легко масштабируема как программа, так и путём подключения дополнительных модулей.

Необходимо определить структуру разрабатываемого программного обеспечения с учётом модульности и выявить необходимые функциональные возможности, которые оно должно выполнять. В структуре разрабатываемой системы будут присутствовать несколько базовых, не зависимых друг от друга компонентов, каждый из которых выполняет отдельную роль в общей системе:

1. Редактор. Редактор является главным модулем в программе, так как через его интерфейс будет происходить процесс построения логики шейдерных программ. Этот модуль будет иметь, преимущественно, монолитную структуру без возможности внешних модификаций.

2. Визуализатор. Визуализатор выполняет функции, связанные с визуализацией пользовательских объектов, за рисование которых отвечает шейдер, реализованный тем же

пользователем. Модуль может иметь дочерние модули, с помощью которых можно будет масштабировать систему.

3. Компилятор. Компилятор является связующим звеном между редактором и визуализатором. Его задача – преобразование логики шейдера из графического представления в шейдерный код, который затем будет помещён в визуализатор и применён на объект сцены.

4. Программа. Модуль программы будет связующим звеном между компонентами. Программа будет отправлять данные в компоненты, а ответы от них отправлять в другие компоненты. Таким образом весь поток данных будет проходить через единый модуль программы.

Следует выделить следующие задачи проектирования:

- разработка модуля редактора;
- разработка модуля визуализации;
- разработка модуля компиляции;
- разработка алгоритма декомпозиции логики на структурные блоки для последующего преобразования в текст шейдера;
- разработка алгоритмов рендеринга изображения;
- разработка интуитивно понятного интерфейса пользователя.

Разрабатываемый проект должен решить следующие основные задачи:

- предоставить удобный и понятный способ создания шейдерных программ;
- Предоставить возможность получать результат, который можно применить в разных проектах, использующих разные API.

2.3 Требования, предъявляемые к программному продукту

Разрабатываемое приложение должно выполнять рад требований:

- разработка логики шейдерных программ должна быть максимально интерактивной;
- графический интерфейс должен быть интуитивно понятный и адаптивный;
- программа работать стабильно и максимально быстро;
- программа должна быть расширяема путём изменения исходного кода, а так же путём дополнения её внешними компонентами.

2.4 Анализ аналогов и прототипов

Для более продуктивной и результативной разработки программного продукта необходимо найти и проанализировать аналоги и прототипы проектируемого продукта. Это делается с целью выявления плюсов и минусов существующих решений, что в дальнейшем будет влиять на всю архитектуру и конечные возможности готового программного продукта. В результате поиска инструментов для графического программирования шейдеров были найдены следующие решения: ShaderFrog, GSN Composer, прочие. Прочие программные продукты не рассматриваются, так как в них отсутствует визуальное программирование.

ShaderFrog – WEB приложение для разработки шейдеров на языке программирования шейдеров GLSL. Основным преимуществом является кроссплатформенность, которая достигается за счёт использования браузера в качестве платформы. Присутствует как текстовый, так и графический редакторы, однако минусом является то, что в визуальном редакторе можно использовать только готовые компоненты, написанные в текстовом редакторе. Это создаёт ряд сложностей для активного творчества неподготовленных пользователей. В качестве дополнительной возможности присутствует возможность экспорта наработок для использования в других системах: Three.js, IOS, Unity, однако последние доступны только обладателям платной подписки.

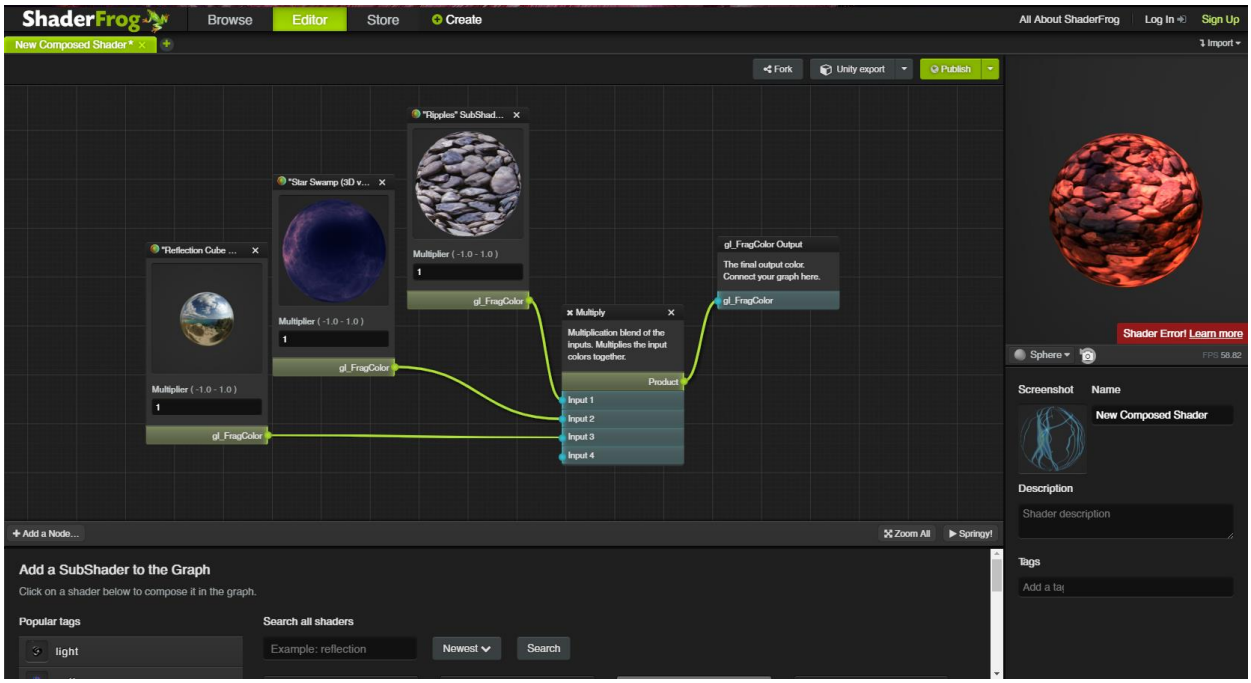

Рисунок 2.4.1 – интерфейс WEB приложения ShaderFrog

GSN Composer – так же является WEB приложением, однако, в отличии от ShaderFrog, основной способ разработки шейдеров – визуальное программирование с

использованием узлов. Разработчику доступно большое количество узлов, выполняющих разные задачи, которые могут быть использованы для достижения поставленной цели. Присутствует возможность экспорта проекта в виде WEB содержимого для встраивания результата на свои HTML страницы.

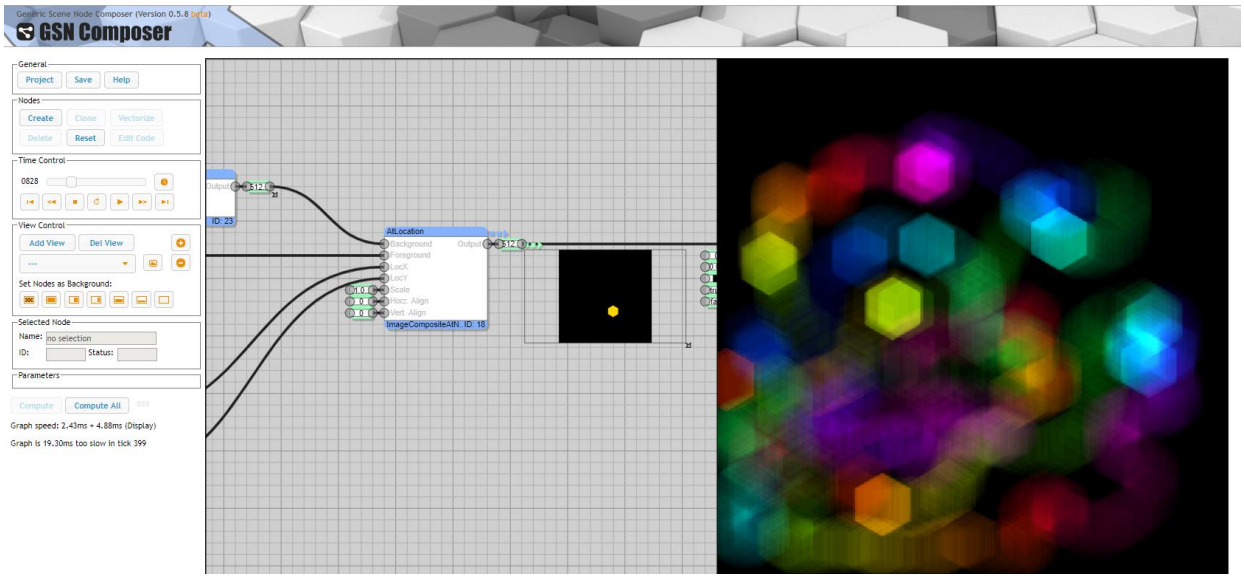

Рисунок 2.4.2 – интерфейс WEB приложения GSN Composer

Минусом этих решений является то, что они, в основном, ориентированы на генерацию кода для программ, использующих API OpenGL. Так же у них нет полностью бесплатной возможности экспортировать проект в другие среды, что было бы очень удобно в некоторых случаях. Одним из таких случаев может стать разработка шейдера, который будет использоваться как OpenGL API в приложениях для всех систем, так и, например, в приложениях, использующих API DirectX, написанных для операционной системы Windows.

2.5 Выбор и обоснование средств и методов решения задач

Для разработки целевого программного обеспечения выбран язык программирования C++. Данный язык программирования является одновременно гибким и мощным, что является плюсом при проектировании системы, в которой производительность имеет значение. Недостатком является то, что для C++ нет стандартной библиотеки для разработки графических программ. Для решения этой проблемы подойдёт Qt Framework. Qt – кроссплатформенный framework, инструментарий которого не ограничивается оконными виджетами. В Qt есть всё, что может понадобиться во время разработки: коллекции, контейнеры, классы для работы с JSON, классы для работы с сетью, а та же реализованный паттерн “Observer”, использование которого допускается со специальным препроцессором MOC.

Для автоматического разворачивания проекта на разных операционных системах и для использования с разными компиляторами необходимо использовать автоматическую кроссплатформенную систему сборки программного обеспечения. CMake – проверенная временем система сборки, которая стала негласным стандартом при разработке кроссплатформенных проектов на языках C/C++. Благодаря CMake выбор IDE для разработки целевого программного обеспечения не имеет значения, так как в конечном итоге проект надо будет приготовить в CMake и скомпилировать компилятором. Современные IDE, такие как MSVS, Qt Creator, CLION, Visual Studio Code и другие умеют работать с CMake, что ещё больше упрощает процесс разработки программного обеспечения на языках программирования C/C++.

Для проверки результата своей работы пользователь будет смотреть в модуль вывода, где по умолчанию будет виден стандартный объект, на который применяется шейдер. Чтобы предоставить пользователю возможность загружать объекты для тестирования своих программ, необходимо знать множество форматов, в которых эти объекты могут храниться. Среди огромного количества форматов сложно выбрать наиболее удобный и гибкий формат, однако в этом нет необходимости. Assimp – библиотека, разработанная для получения геометрии объектов и сцен из множества поддерживаемых форматов. Assimp может разобрать более 20 форматов файлов с геометрией, а так же автоматически выполнить оптимизации, добавить данные о вершинах и исправить ошибки в модели.

Главной задачей, которую необходимо будет решить – сбор информации о логике программы для её дальнейшего преобразования в исходный код шейдера. Для этих целей оптимально будет использовать JSON-объекты. Для работы с JSON-объектами в Qt реализованы специальные классы, использование которых поможет построить схему логики программы в единый JSON-объект, который затем можно будет передать и обработать в компиляторе.

2.6 Разработка технического задания

Техническое задание на разрабатываемое программное обеспечение представлено в приложении А.

3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1 Функциональная структура разрабатываемой системы

Разрабатываемая программа должна выполнять три набора разных действий, которые можно объединить в подсистемы:

- подсистема редактирования;
- подсистема компилирования;
- подсистема визуализации.

Рассмотрим функции, которые должна выполнять каждая из подсистем.

Подсистема редактирования выполняет работу, связанную с подготовкой пользовательской логики к компиляции, а также предоставляет инструменты для составления пользовательской логики шейдерной программы. В зависимости от текущего действия пользователя выбирается подкомпонент, выполняющий функционал:

- создание переменной;
- перемещение узла переменной на сцену;
- управление узлами на сцене;
- создание соединений между узлами;
- удаление узлов;
- добавление узлов на сцену;
- управление сценой;
- поиск пользовательских узлов;
- загрузка пользовательских узлов;
- загрузку пользовательского проекта;
- сохранение пользовательского проекта.

Подсистема компилирования выполняет функцию преобразования шейдерной логики в шейдерный программный код, выполняя следующие функции:

- чтение списка доступных компиляторов;
- загрузка необходимого компилятора;
- создание логического дерева;
- преобразование в исходный код.

Подсистема визуализации отвечает за отображение результатов работы пользователя и программы, для этого выполняя следующие функции:

- загрузка целевого объекта;
- установка целевого объекта;

- установка камеры для управления сценой;
- установка сцены с нужным целевым контекстом;
- отображение результата рендеринга изображения;
- управление параметрами рендеринга;
- управление передачей данных в шейдер.

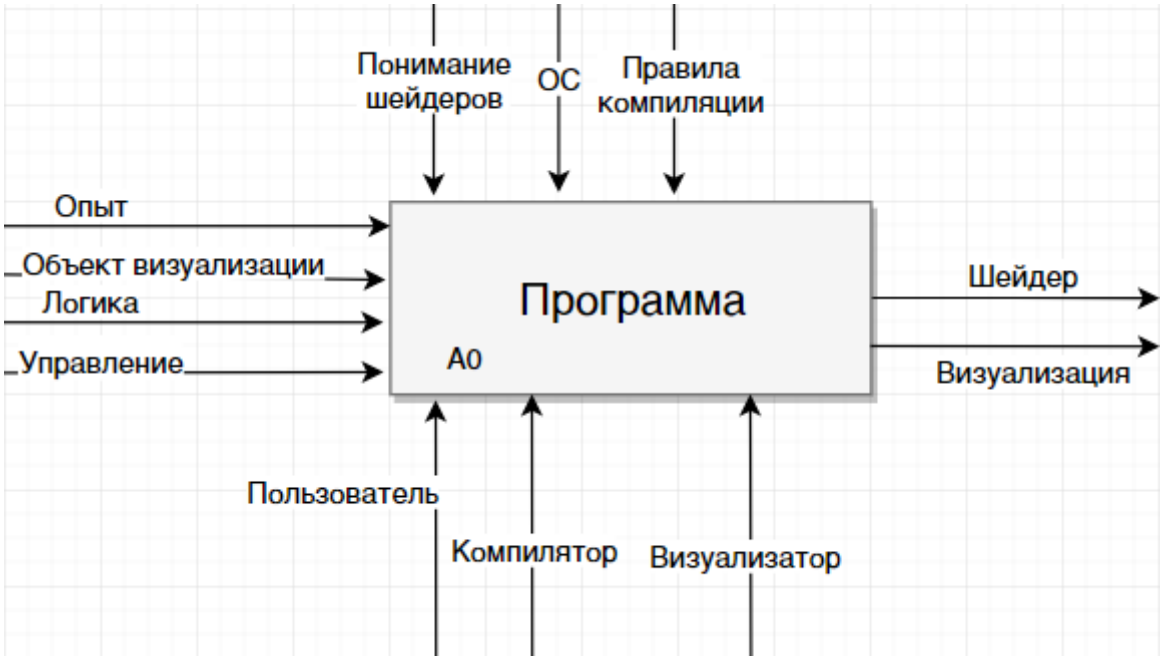

Рисунок 3.1.1 – функциональный блок «программа»

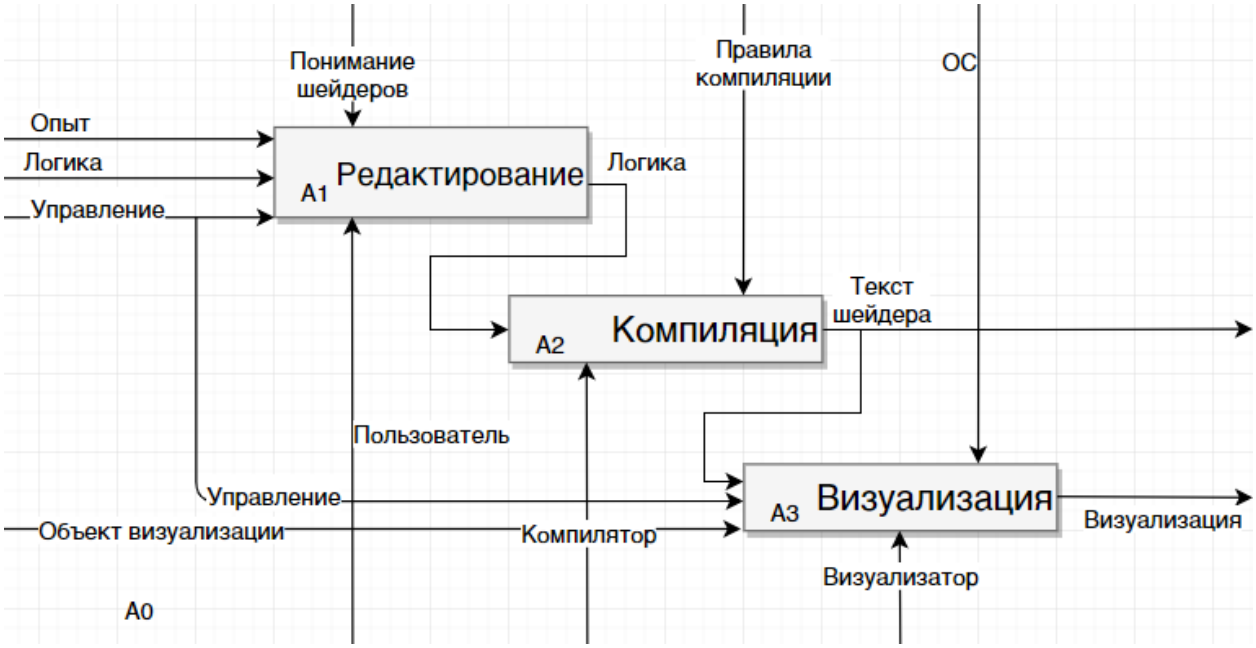

Рисунок 3.1.2 – функциональная декомпозиция программы

3.2 Описание вариантов использования

Диаграмма вариантов использования (use case diagram) описывает функциональное назначение системы в самом общем виде с точки зрения всех ее пользователей и заинтересованных лиц. Диаграмма вариантов использования представляет собой диаграмму, на которой изображаются варианты использования проектируемой системы, как правило, заключенные в границу субъекта и внешние актеры, а также определенные взаимоотношения между актерами и вариантами использования. Данная диаграмма предназначена для достижения следующих целей:

- определения общих границ функциональности проектируемой системы в контексте моделируемой предметной области;
- предъявления требований к функциональному поведению проектируемой системы в форме вариантов использования;
- разработке исходной концептуальной модели системы для ее последующей детализации в форме логических и физических моделей;
- подготовке исходной документации для взаимодействия разработчиков системы с ее заказчиками.

Таким образом, основным назначением диаграммы вариантов использования является спецификация функциональных требований к проектируемой системе. Так как требования выступают в качестве исходных данных для разработки системы, то визуализация их в форме диаграммы повышает наглядность представления и позволяет целенаправленно управлять процессом разработки других моделей на языке UML.

Основными элементами (предметами языка UML), отображаемыми на диаграмме вариантов использования, являются: вариант использования, актер, системная граница, примечание.

Вариант использования представляет собой общую спецификацию совокупности выполняемых системой действий с целью предоставления некоторого наблюдаемого результата, имеющего значение для одного или нескольких актеров. Проще говоря, вариант использования представляет собой законченный фрагмент поведения системы с точки зрения тех или иных заинтересованных лиц без указания технических или физических особенностей его реализации. Описание этого фрагмента поведения называется сценарием. Сценарии могут задаваться в нескольких формах: в виде обычного неструктурированного текста, в виде упорядоченного списка действий или в виде текста на некотором формализованном языке.

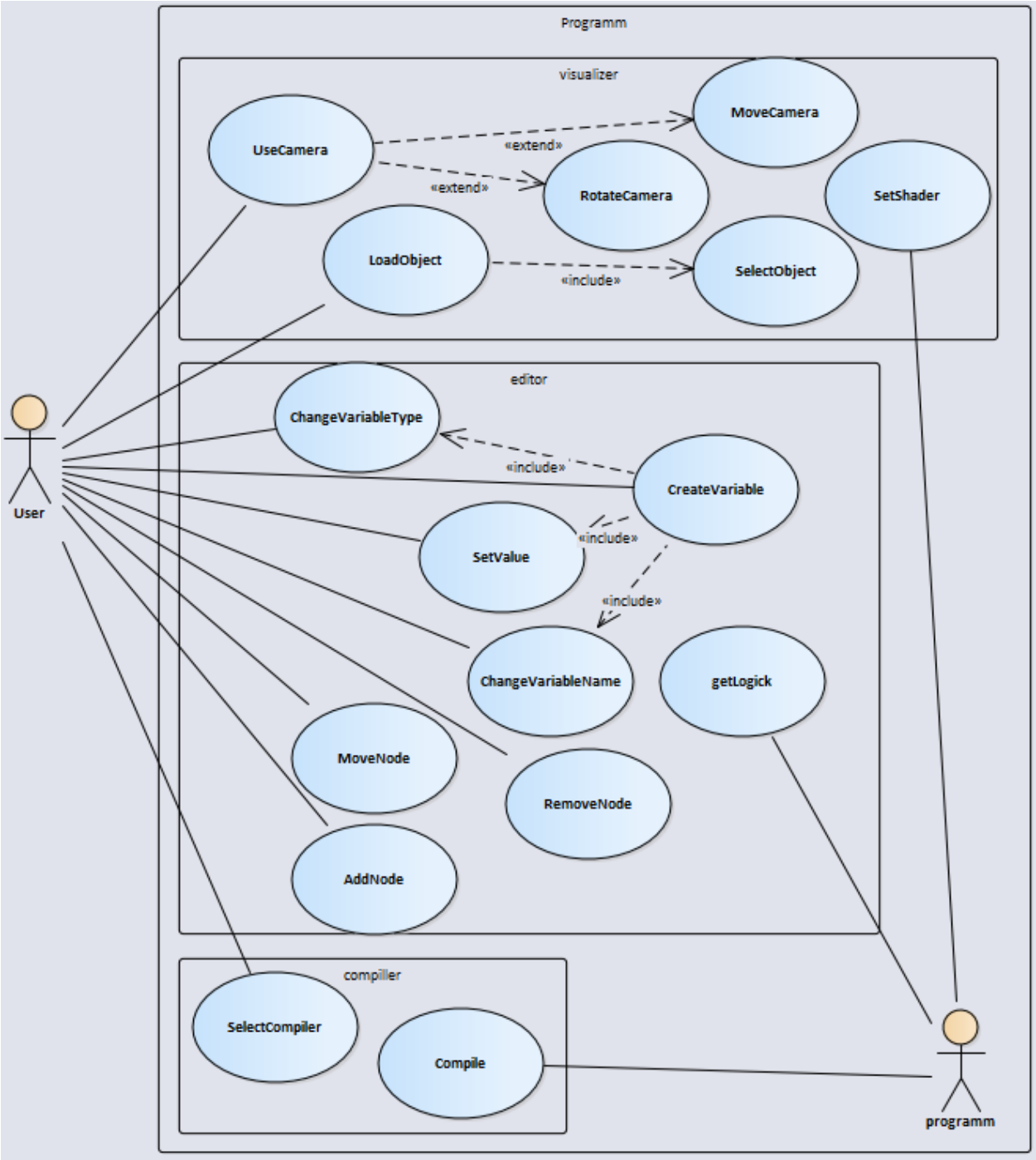

Рисунок 3.2.1 – диаграмма вариантов использования

3.3 Проектирование пользовательского интерфейса

Пользовательский интерфейс является своеобразным коммуникационным каналом, по которому осуществляется взаимодействие пользователя и устройства. Поэтому одной из важных задач при разработке приложения является проектирование пользовательского интерфейса.

На рисунке 3.3.1 представлен прототип узлового построения. На рисунке видно, как соединяются между собой блоки через входы и выходы одинаковых цветов. На рисунке

3.3.2 представлен прототип окна визуализации сцены с пользовательским объектом. Присутствуют кнопки для запуска сцены, для паузы и для выбора файла визуализируемой модели.

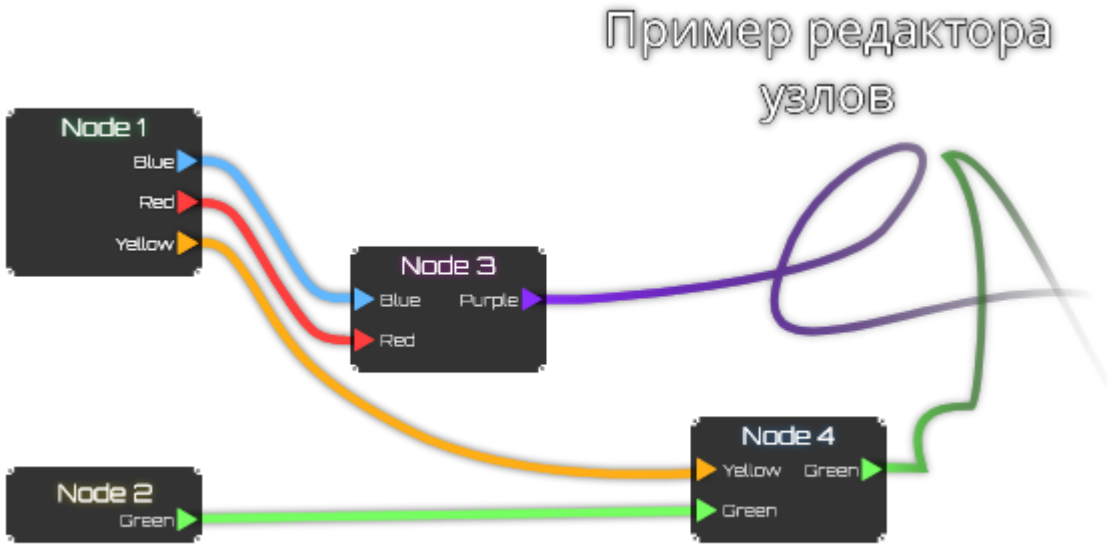

Рисунок 3.3.1 – проектируемое графическое представление узлов

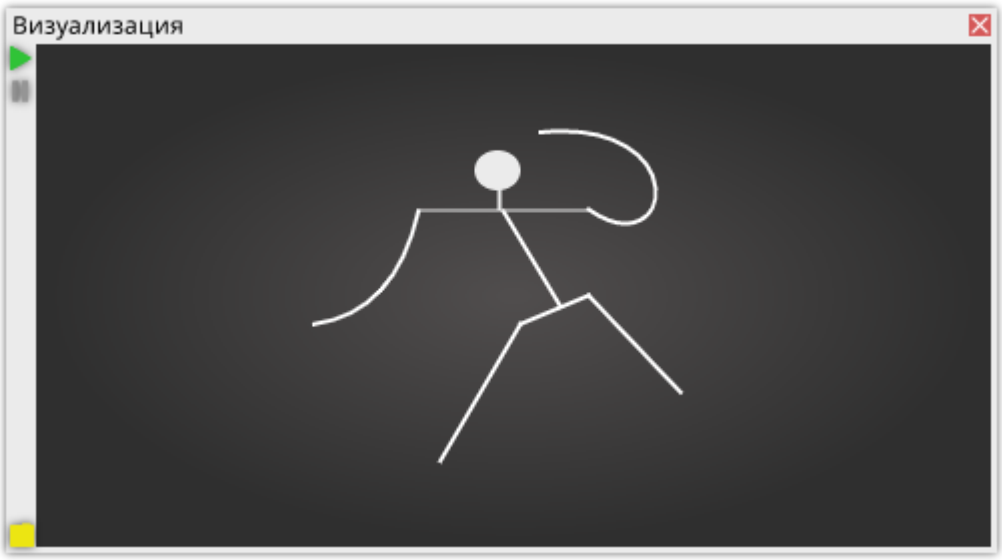

Рисунок 3.3.2 – Страница поиска препаратов

ЗАКЛЮЧЕНИЕ

В результате прохождения преддипломной практики была сформулирована тема дипломного проектирования – графическое приложение для разработки шейдерных программ с использованием визуального программирования.

В рамках прохождения преддипломной практики были разработаны диаграмма функционального представления, диаграмма вариантов использования. Так же был разработаны элементы интерфейса программы.

Для разработки программного продукта был выбран язык программирования C++ совместно с Qt Framework для разработки графической части программного продукта, а также для использования дополнительных возможностей Qt.

В результате прохождения преддипломной практики были достигнуты следующие цели:

- сбор и изучение информации необходимой для написания дипломного проекта;
- приобретение профессиональных навыков по профилю специальности;
- закрепление, расширение и систематизация знаний, полученных при изучении специальных дисциплин.

В результате прохождения индивидуальной аналитической части практики были достигнуты следующие результаты:

- проанализирована предметная область;
- произведен сравнительный анализ аналогов и прототипов разрабатываемого программного продукта;
- выбраны средства и методы решения задачи;
- выполнено проектирование программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Википедия [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ . Дата обращения: 30.03.2019 – 22.04.2019.

2 Диаграмма вариантов использования [Электронный ресурс]/ Файловый архив для студентов – Режим доступа: http://www.studfiles.ru/preview/3529698/. – Дата доступа: 09.04.2017

3 Office-menu.ru [Электронный ресурс] – Режим доступа: http://office-menu.ru/uroki-sql/41-tipy-svyazej-v-relyatsionnykh-bazakh-dannykh. Дата обращения 20.04.2019 – 22.04.2019.

ПРИЛОЖЕНИЕ А
(обязательное)
Техническое задание

Введение

Наименование продукта: «Графическое приложение для разработки шейдерных программ с использованием визуального программирования».

Программный продукт будет использоваться для разработки шейдерных программ для разных платформ и задач. Результат работы разрабатываемой программы направлен на использование в других программах, которые используют шейдеры для рисования объектов.

А.1 Основание для разработки

Основанием для разработки является задание по дипломному проектированию на тему «Графическое приложение для разработки шейдерных программ с использованием визуального программирования», выданное студенту группы 15-ИТ-1 Стеняеву А. Д., руководителем назначен преподаватель-стажёр кафедры технологий программирования Макарычева В. А. Разработка проводится в соответствие с приказом № XXX от dd.mm.2019 г. о закреплении тем дипломных проектов студентов факультета информационных технологий дневной формы обучения по специальности 1-40-01-01 «Программное обеспечение информационных технологий» по кафедре технологий программирования УО «Полоцкий государственный университет».

А.2 Назначение разработки

Разрабатываемое программное обеспечение позволит пользователям разрабатывать шейдерные программ для разных платформ и задач. Программа должна реализовать возможность выстраивать логику шейдерных программ в графическом режиме путём построения логики шейдера с помощью логических узлов. После реализации логической структуры шейдерной программы разрабатываемое программное обеспечение должно предоставлять возможность скомпилировать полученый результат в текст-программу шейдера на желаемом языке программирования шейдеров.

А.3 Требования к программному изделию

А.3.1 Требования к интерфейсу

Программа должна иметь дружественный и интуитивно понятный интерфейс.

Диалоговое окно приветствия должно включать в себя возможность создать новый проект или открыть существующий.

Интерфейс окна просмотра результата должно включать в себя механизмы для загрузки объектов разных форматов для их дальнейшей визуализации. Так же должна представляться сцена с загруженным объектом и применённым к нему шейдером. Управление состоянием сцены должно быть реализовано с помощью мыши.

Интерфейс графического редактора логики шейдерной программы должен реализовать модель узлов и предоставлять простой способ для создания, удаления, перемещения и объединения узлов конечной логики шейдерной программы. Должен существовать список переменных и доступных функций, применимых к этим переменным. Переменные создаются пользователем. Исключением являются переменные, обязательные к существованию.

Интерфейс настроек проекта должен предоставлять доступ к управлению параметрами графического конвейера сцены, вариантами отображения вспомогательной информации и т.д.

А.3.2 Требования к функциональным характеристикам

Данная программа должна предоставлять функционал, необходимый для разработки логики шейдерных программ. Пользователь программы должен получить следующие функциональные возможности:

- создание, управление, удаление узлов в редакторе шейдерной логики;
- создание, управление, удаление переменых в редакторе шейдерной логики;
- возможность сохранять результат в желаемом доступном формате проекта и результирующего шейдера;
- возможность создавать и продолжать развивать существующие пользовательские проекты.

А.3.3 Требования к надежности

Программное средство должно обеспечивать контроль корректности входных данных. В случае обнаружения ошибок во входных данных пользователю должны выводиться соответствующие сообщения с указанием возможных путей исправления.

К надежности программы предъявляются следующие требования:

- соблюдение целостности данных;
- корректность вносимых данных.

А.3.4 Условия эксплуатации

Конечные пользователи программы – любой человек. Программный продукт может быть использован любым пользователем, в том числе и полностью не подготовленным. Особых условий для хранения и использования не требуется.

Максимальной эффективности может добиться только подготовленный пользователь, который понимает, что такое шейдерная программа, какая у неё структура и как её логически реализовать.

А.3.5 Требования к составу и параметрам технических средств

В состав технических средств должен входить компьютер.

Минимальными и достаточными требованиями по конфигурации оборудования для функционирования системы являются:

Для нормального функционирования программного средства минимальный состав и параметры технических средств должны соответствовать нижеследующему:

- Процессор с тактовой частотой от 2000 mhz и выше;
- Оперативная память от 4 ГБ;
- Жесткий диск с объемом памяти не менее 128 ГБ свободного дискового пространства;
- Видеокарта с объемом оперативной памяти 1024 МБ и поддержкой стандарта OpenGL 3.3;
- Монитор с минимальным матрицей 1024x768@60Hz 24bit
- Манипулятор типа мышь PS/2 или USB;
- Клавиатура стандартная PS/2 или USB (101/102 клавиши);
- Операционная система Windows 8.1 и выше.

А.3.6 Требования к информационной и программной совместимости

А.3.6.1 Требования к исходным кодам и языкам программирования

Для компиляции проекта необходимо иметь компилятор, поддерживающий C++17 или выше. CMake версии 3.10 или выше.

А.3.6.2 Требования к программным средствам, используемым программой

- операционная система: Windows 8.1 и выше, Linux, Mac OS X;
- драйвер видеокарты, совместимый с OpenGL 3.3 и выше.

А.3.6.3 Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

А.3.7 Требования к маркировке и упаковке

Требования к маркировке и упаковке отсутствуют.

А.3.8 Требования к транспортировке и хранению

Поставка данного программного продукта не требуется. Хранение будет осуществляться на сервере, а доступ – посредством сети интернет.

А.4 Требования к программной документации

Программная документация должна состоять из следующих документов:

- описание программы (сведения о логической структуре и функционировании программы);
- пояснительная записка;
- руководство пользователя;
- руководство системного администратора;
- программа и методика испытаний (требования, подлежащие проверке при испытании программы);

Содержание и структура программной документации соответствует требованиям ЕСПД.

А.5 Технико-экономические показатели

Технико-экономические показатели отсутствуют.

А.6 Стадии и этапы разработки

Разработка программы включает в себя следующие стадии:

- анализ исходных данных и постановка задачи проектирования;
- разработка и утверждение технического задания;
- разработка структуры приложения;
- разработка отдельных модулей системы;
- интегрирование модулей в систему;
- тестирование системы;
- отладка системы;
- разработка программной документации.

А.7 Порядок контроля и приемки

Контроль выполнения осуществляется руководителем еженедельно. Так же для контроля выполнения требований технического задания необходимо провести испытания. Порядок и состав испытания определяются программой и методикой испытаний. Контроль и приемка программного обеспечения осуществляются в соответствии с программой и методикой испытаний, разработанной по ГОСТ 19.301-2000 «Программа и методика испытаний. Требования к содержанию и оформлению».

Основным методом испытания программы будет визуальный контроль выполнения программой требующихся функций.