Частное учреждение образование

Колледж бизнеса и права

ОТЧЕТ

ПО ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

## ОП Т.594010

Руководитель практики (Д.В. Муха)

от предприятия

М.П.

Руководитель практики

от колледжа (Н.В. Ржеутская)

Учащийся (А.С. Ковалёв)

2019

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.594010.401

Разраб.

Ковалёв А.С.

Провер.

РжеутскаяН.В.

Реценз.

Н. Контр.

Утверд.

Отчет

по технологической практике

Лит.

Листов

КБП

1. Структура производства, характеристики основных видов деятельности
2. Должностные обязанности оператора ЭВМ, техника-программиста, инженера-программиста
3. План мероприятий по разработке и внедрению программы
4. Техно-рабочий проект
   1. Постановка задачи
   2. Выбор языка программирования
   3. Определение структуры данных
   4. Алгоритм решения задачи
   5. Руководство программиста
   6. Инструкция по эксплуатации
5. Опытная эксплуатация.
   1. Ошибки, выявленные в процессе опытной эксплуатации

Выводы

Список используемых источников

Приложение А Текст программных модулей

1 Структура производства, характеристики основных видов деятельности

Место прохождения практики - открытое акционерное общество (ОАО) «АЛЕВКУРП».

Открытое акционерное общество «АЛЕВКУРП» специализируется на выполнении НИОКР по разработке и модернизации, а также по созданию новых радиолокационных систем, систем управления оружием с использованием новейших технологий цифровой обработки сигналов и информации в реальном масштабе времени, а также техники СВЧ и промежуточной частоты. Также предприятие занимается ремонтом и модернизацией зенитного вооружения. В настоящее время география поставок предприятия охватывает Африку, Юго-Восточную Азию, Латинскую Америку.

Компания специализируется на научных исследованиях в области радиолокационных систем, оптических электронных систем, систем автоматического управления.

В настоящее время предприятие располагает высококвалифицированными научными и инженерно-конструкторскими кадрами, обладающими опытом создания и модернизации радиолокационной техники с использованием новейших технологий цифровой обработки радиолокационных сигналов в реальном масштабе времени.

На текущий момент структура компании выглядит следующим образом.

Структура предприятия представлена на рисунке 1.



Рисунок 1 - Структура предприятия

Примеры продукции ОАО «АЛЕВКУРП» представлены в таблице 1.

Таблица 1 - Неполный список проектов ОАО «АЛЕВКУРП» и краткое их описание.

|  |  |
| --- | --- |
| Название  проекта | Краткое описание |
| [ЗРК средней дальности С-125-2БМ «ПЕЧОРА-2БМ»](https://alevkurp.by/produktsiya-i-uslugi/glubokaya-modernizatsiya-zrk/zrk-sredney-dalnosti-s-125-2bm-pechora-2bm.html) | Зенитный ракетный комплекс (ЗРК) C 125 2БМ «Печора 2БМ» с зенитными управляемыми ракетами 5В27Д является мобильным зенитным ракетным комплексом средней дальности, предназначенным для уничтожения пилотируемых и беспилотных средств воздушного нападения на малых и средних дальностях |
| [Модернизация ЗРК КУБ (КВАДРАТ) до уровня (КВАДРАТ-МА)](https://alevkurp.by/produktsiya-i-uslugi/glubokaya-modernizatsiya-zrk/modernizatsiya-zrk-kub-kvadrat-do-urovnya-kvadrat-ma.html) | Данная модернизация позволяет получить практически новый, цифровой комплекс с использованием малой части старых узлов и блоков. Новые цифровые системы формирования, приема и обработки радиолокационных сигналов позволяют существенно упростить процедуру настройки и регулировки аппаратуры. |
| Устройства СВЧ | Блоки высокочастотных и сверхвысокочастотные малошумящие приемники, малогабаритные генераторы, элементы волноводного тракта, высокочастотные передатчики, антенны и антенны элементов, аттенюаторов, делителей, сумматоров, фильтров и другое |

2 Должностные обязанности оператора ЭВМ, техника-программиста, инженера-программиста

Для выполнения возложенных на него функций техник-программист обязан:

1. Выполнять работу по обеспечению механизированной и автоматизированной обработке информации на обслуживаемых вычислительных средствах, разработке технологии решения экономических и других задач производственного и научно-исследовательского характера.

2. Разрабатывать программы решения простых задач, проводить их отладку и экспериментальную проверку отдельных этапов работ.

3. Выполнять работу по подготовке технических носителей информации, обеспечивающих автоматический ввод данных в вычислительные средства, по накоплению и систематизации показателей нормативного и справочного фонда, внесению необходимых изменений и своевременному корректированию рабочих программ.

4. Участвовать в выполнении различных операций технологического процесса обработки информации (прием и контроль входной информации, подготовка исходных данных, обработка информации, выпуск исходящей документации и передача ее заказчику).

5. Участвовать в наладке, испытаниях и сдаче в эксплуатацию опытных образцов вычислительных средств, входящих в новые и модернизированные образцы выпускаемой предприятием продукции, а также в работах по совершенствованию, модернизации, унификации вычислительных средств, их элементов.

6. Вести учет объемов выполненных работ.

7. Выполнять отдельные служебные поручения своего непосредственного руководителя.

8. Выполнять требования документов системы менеджмента качества ОАО «АЛЕВКУРП», в части его ответственности.

3 План мероприятий по разработке и внедрению программы

Разработка программного приложения происходит в несколько этапов. Обычно этих этапов пять. На каждом этапе выполняются определенные работы. В таблице 2 описаны все этапы разработки по порядку.

Таблица 2. Этапы разработки

|  |  |
| --- | --- |
| Описание этапа разработки | Срок выполнения |
| Первый этап по разработке приложения −постановка и утверждение технического задания. Целью разработки технического задания является обоснование необходимости разработки данного программного средства (ПС). На этом этапе выполняются работы по обследованию информационных потоков в подразделении, в котором осуществляется автоматизация управления, постановка задачи, сбор исходных материалов и определение структуры входных и выходных данных. | с 06.02.2019 по 07.02.2019 |
| Второй этап разработки − это разработка и утверждение эскизного проекта. На этом этапе выполняются работы по предварительной разработке структуры входных и выходных данных, уточнение методов решения задачи, разработка общего описания методов решения задачи, а также согласование и утверждение эскизного проекта. | с 08.02.2019 по 21.02.2019 |
| Третьим этапом является разработка и утверждение технического проекта. На этом этапе происходит уточнение структуры входных и выходных данных, разработка структуры ПС в контексте среды разработки, определение формы представления входных и выходных данных, разработка плана по разработке и внедрению программы. | с 22.02.2019 по 14.03.2019 |
| Четвертым этапом является разработка рабочего проекта. В рабочий проект входит программные модули и программная документация, то есть рабочий проект непосредственно сама разрабатываемая программа со всей вспомогательной и программной документацией. На этой стадии ведутся работы по программированию и отладке модулей, разработке программных документов в соответствии с требованиями единой системы программной документации (ЕСПД) и техническим заданием, а также корректировка программы и программной документации по результатам испытаний. | с 15.03.2019 по 28.03.2019 |
| Пятым этапом является внедрение программы на предприятие, то есть подготовка к передаче программы на предприятие. Здесь ведутся работы по подготовке и передаче программы и программной документации для сопровождения, оформление и утверждение акта о передаче программы на сопровождение, работа в службе сопровождения (по эксплуатации разработанной программы). Внесение изменений в соответствии с выявленными в процессе опытной эксплуатации ошибками и недоработками. | с 29.03.2019 по 01.04.2019 |

4 Техно-рабочий проекта

4.1 Исследование предметной области

Сегодня вопрос вооружения Республики Беларусь направлен в большей степени на создание высококачественного оборонного комплекса. Опыт истории показывает, что не иметь возможности достойной защиты означает геноцид населения и взятие в плен. Так же благодаря военным разработкам создаются инновационные технологии, которые в скором времени попадают на потребительский рынок: компьютеры [7], нашлемная система индикации [8], радионавигация [9] и другое. Очевидно, что развивать технологии можно и другим путем, но при векторе развития через вооружение можно получить и хорошее оружие, а в следствии и защиту.

Разрабатываемый модуль должен обеспечивать следующие функции:

* создание редактирование сцены;
* просмотр сцены;
* построение плана полета объекта и его просмотр;
* сохранение информации о полете объекта в файл типа «xml».

4.2 Структура входных и выходных данных

Входными данными в программе являются:

* ключевые точки полета тела в пространстве;
* тип маневра ключевых точек;
* атрибуты тела;
* информация о полете.

Так же входными данными могут являться файлы типа .obj и выходные файлы программы.

К выходным файлам программы относятся файлы типа .xml или .json, содержащие в себе информацию о:

* полете тела;
* структуре сцены.

4.3 Проектирование модели

Документы, ориентированные на данные, характеризуются четкой упорядоченной структурой, их глубокой проработкой (наименьший независимый элемент данных может располагаться на уровне атрибута), кроме того, смешанного информационного наполнения нет, либо оно присутствует в минимальном объеме. Порядок, в котором появляются однородные элементы, часто не имеет значения. Хорошим примером в данном случае служат XML-документы.

Исходя из требований, выведенных на этапе проектирования, требуется применять в качестве хранения данных xml-документы или json-документы. Такое решение обусловлено сложностью переноса обработки данных в среде Windows в среду Linux и простой организацией данных.

Исходя из анализа предметной области, выделяем главные сущности:

– точка маневра.

– местонахождение тела;

Структура данных разрабатываемого программного средства включает две таблицы.

Структура данных таблиц, и их краткое описание приводится в таблицах 3-4.

Таблица «maneuver points» хранит информацию об очках маневра.

Таблица «flight points» хранит информацию о точках полета.

Таблица 3 – Структура таблицы «maneuver points»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,  байт | Описание |
| x | float | 4 | Положение точки в пространстве по «x» |
| y | float | 4 | Положение точки в пространстве по «y» |
| z | float | 4 | Положение точки в пространстве по «z» |

Таблица 4 – Структура таблицы «flight points»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,  байт | Описание |
| x | float | 4 | Положение точки в пространстве по «x» |
| y | float | 4 | Положение точки в пространстве по «y» |
| z | float | 4 | Положение точки в пространстве по «z» |
| time | decimal | 8 | Время с момента полета в наносекундах |

Выходной информацией данного приложения будут данные в формате .json или .xml.

Физическая структура данных представлена схемой данных на рисунке 3.1.

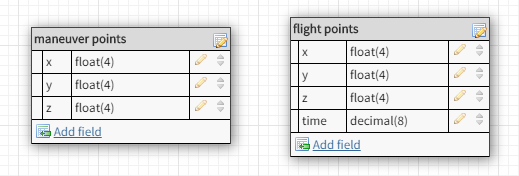


Рисунок 2 – физическая структура базы данных

4.4 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного модуля будет представлять оконное приложение Windows.

В модуле предполагается создание тестовой формы для демонстрации функций модуля. Данная форма будет состоять из текстовых полей ввода, выпадающего списка функций, области отрисовки, списка точек в пространстве.

На форме будут располагаться различные элементы управления, предназначенные для определенных задач. За компонентом Button будут закреплены разные действия для выполнения определенных функций, таких как: выполнение функций и, очистка и данных с формы.

4.5 Реализация функций

Модуль «Сцена» предназначен для создания плана полета тела, редактирования этого плана и просмотра общего плана.

Модуль «Запуск» предназначен для эмуляции полета тела по заданному маршруту с заданными критериями.

Модуль «Выделение объектов» предназначен для упрощения работы с выбором объектов на сцене.

При клике компьютерной мыши на сцене будет происходить выделение объектов. Код функции представлен ниже:

public SceneElement DoHitTest(int mouse\_x, int mouse\_y, OpenGLControl viewport)

{

var colorsElements = this.CreateSelectColors();

var clickedColor = ColorPickerHelper.PerformSelection(viewport, colorsElements, this.Camera, mouse\_x, mouse\_y);

return colorsElements.ContainsKey(clickedColor) ? colorsElements[clickedColor] : null;

}

4.6 Функциональное тестирование

В процессе написания программного средства необходимо производить тестирование на правильность работы приложения. Одной из основных задач тестирования является устранение ошибок, происходящих при вводе данных.

Функциональное тестирование – это тестирование функций приложения на соответствие требованиям. Оценка производится в соответствии с ожидаемыми и полученными результатами (на основании функциональной спецификации), при условии, что функции отрабатывали на различных значениях.

Тестирование программы будет производиться последовательно, переходя из одной части программы в другую. Во время теста будут проверяться все действия с программой, навигация пунктам меню, которые может произвести пользователь. После чего, все собранные и найденные ошибки будут исправлены.

Главное меню программы представлено на рисунке 3.

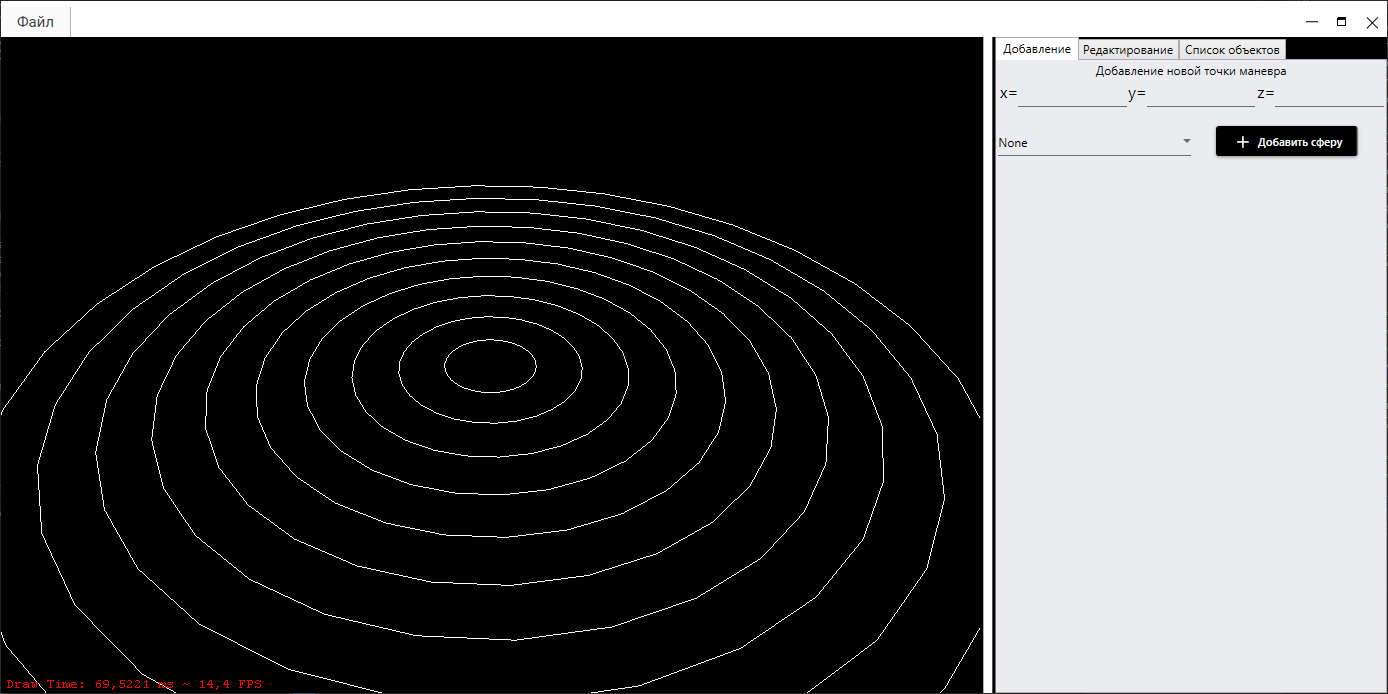


Рисунок 3 – Главное меню

Таблица 5 – Test-case для выбора объекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 001 | Позитивный | Выбор объекта на сцене | 1. Ввести данные новой точки маневра:  * Координата «x»; * Координата «y»; * Координата «z»; * Тип маневра.  1. Нажать кнопку добавить сферу;   3. Нажать на появившуюся сферу. | Ожидаемый результат  В меню «Редактирование» появились параметры выделенной точки |
| Фактический результат  Результат соответствует ожидаемому. Все условия соблюдены. Результат добавления данных представлен на рисунках 4, 5 и 6. |

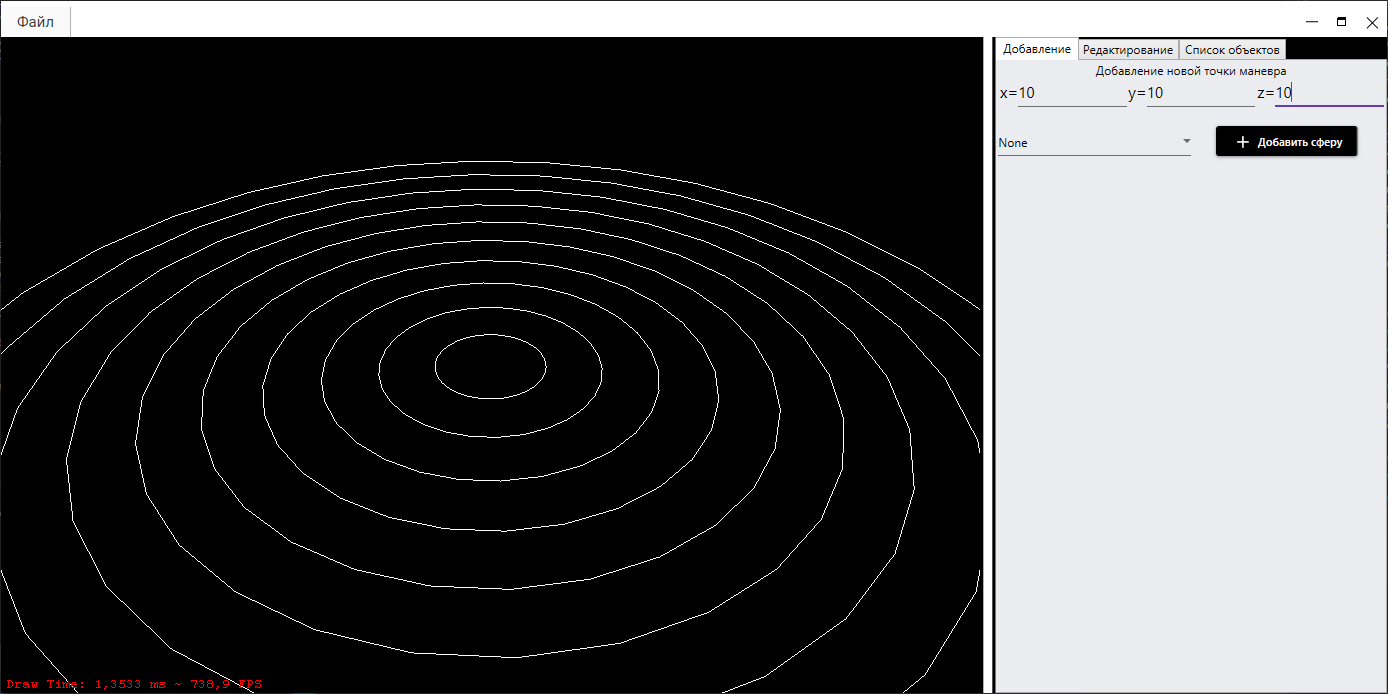


Рисунок 4 – Тест выбора объекта

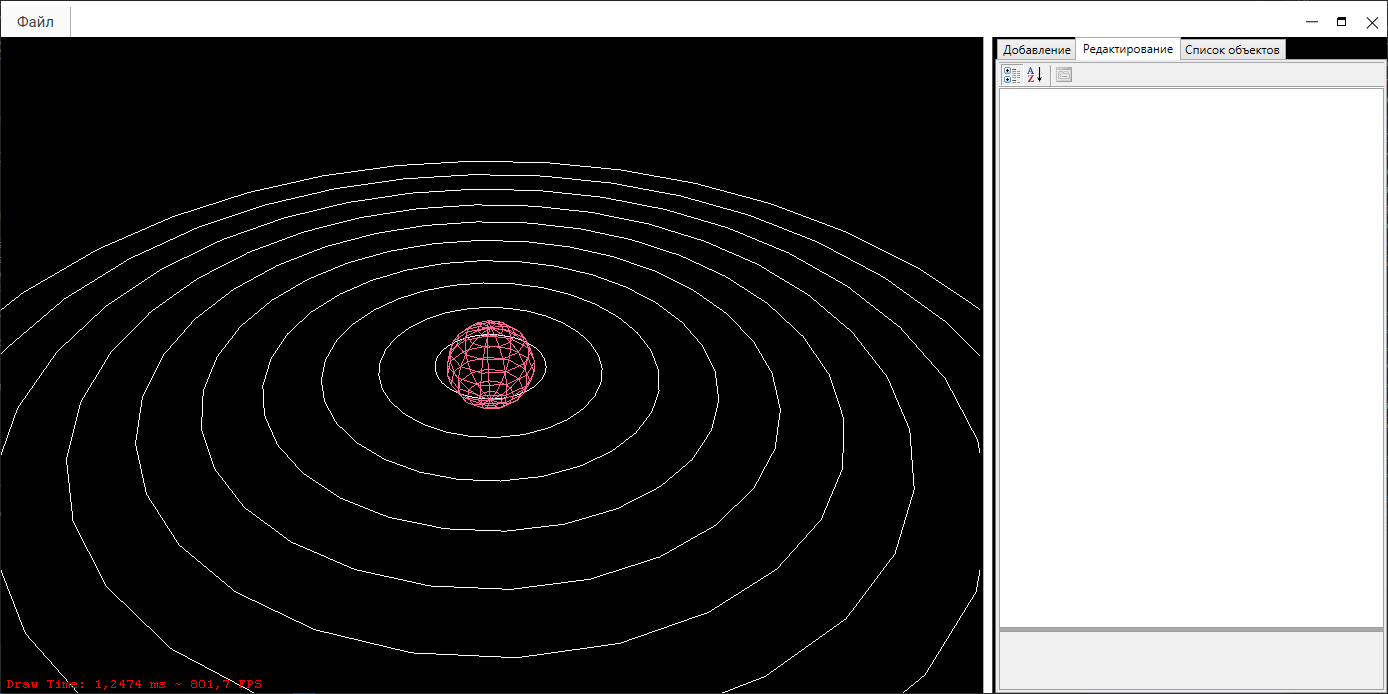


Рисунок 5 – Тест выбора объекта

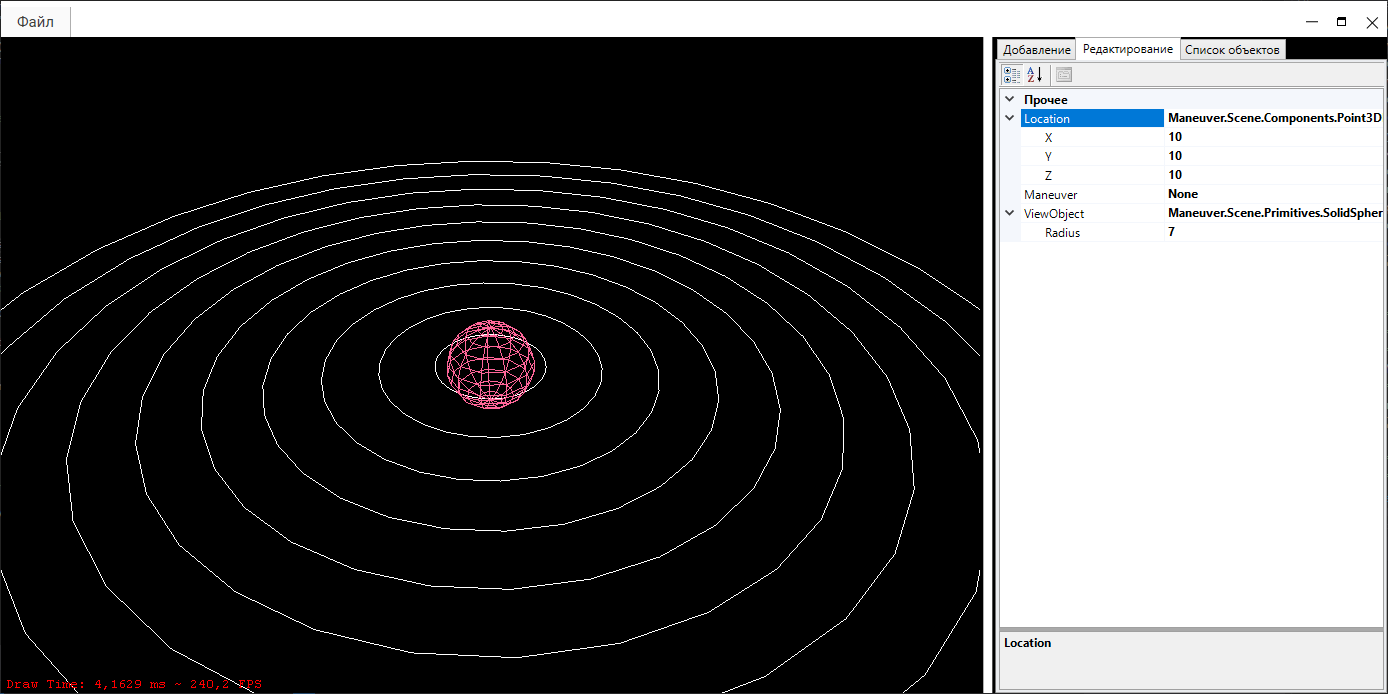


Рисунок 6 – Результат выбора объекта

Таблица 6 – Test-case для изменения радиуса сферы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тип теста | Описание | Шаги воспроизведения | Результат |
| 002 | Позитивный | Изменение данных о клиенте | 1. Нажать на сферу; 2. Перейти в меню «Редактирование»; 3. Выбрать параметр «Radius» в свойстве «ViewObject»; 4. Изменить значение. | Ожидаемый результат  Радиус выбранной сферы изменен |
| Фактический результат  Результат соответствует ожидаемому. Все условия соблюдены. Результат изменения данных представлен на рисунке 7 и 8. |

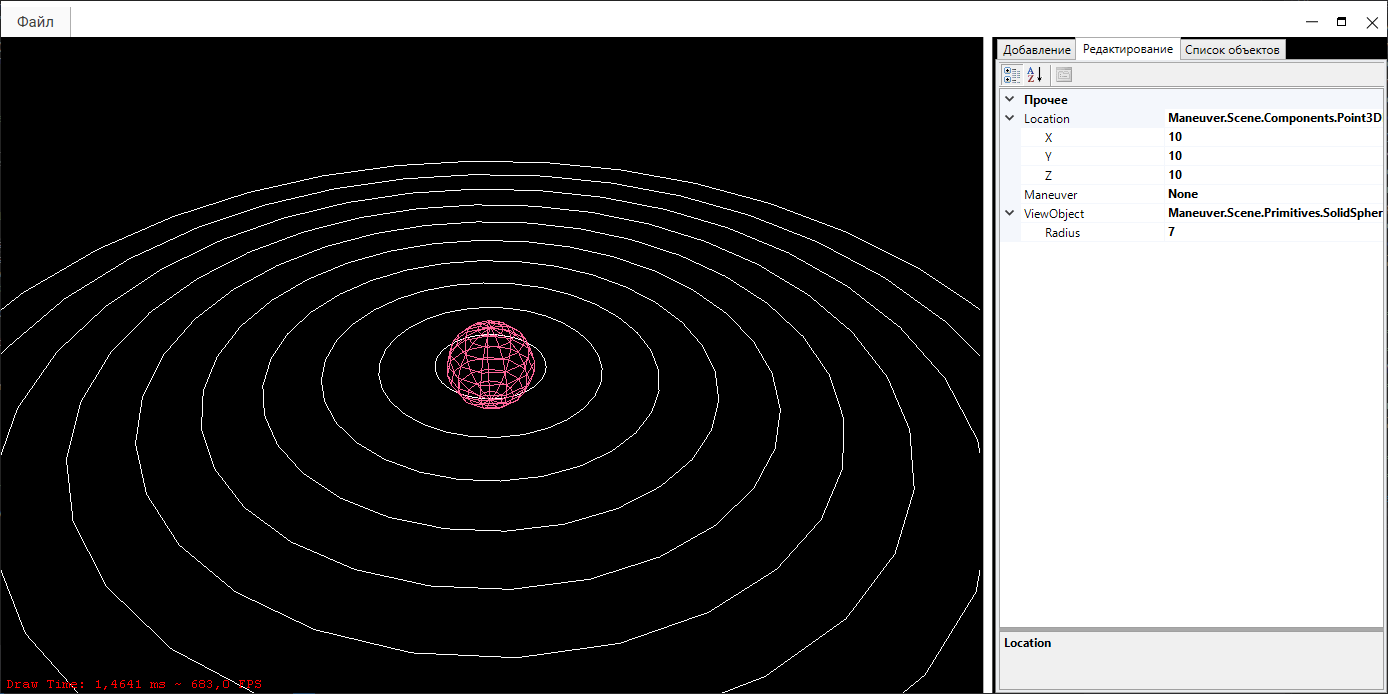


Рисунок 7 - Тест изменения данных в базе данных

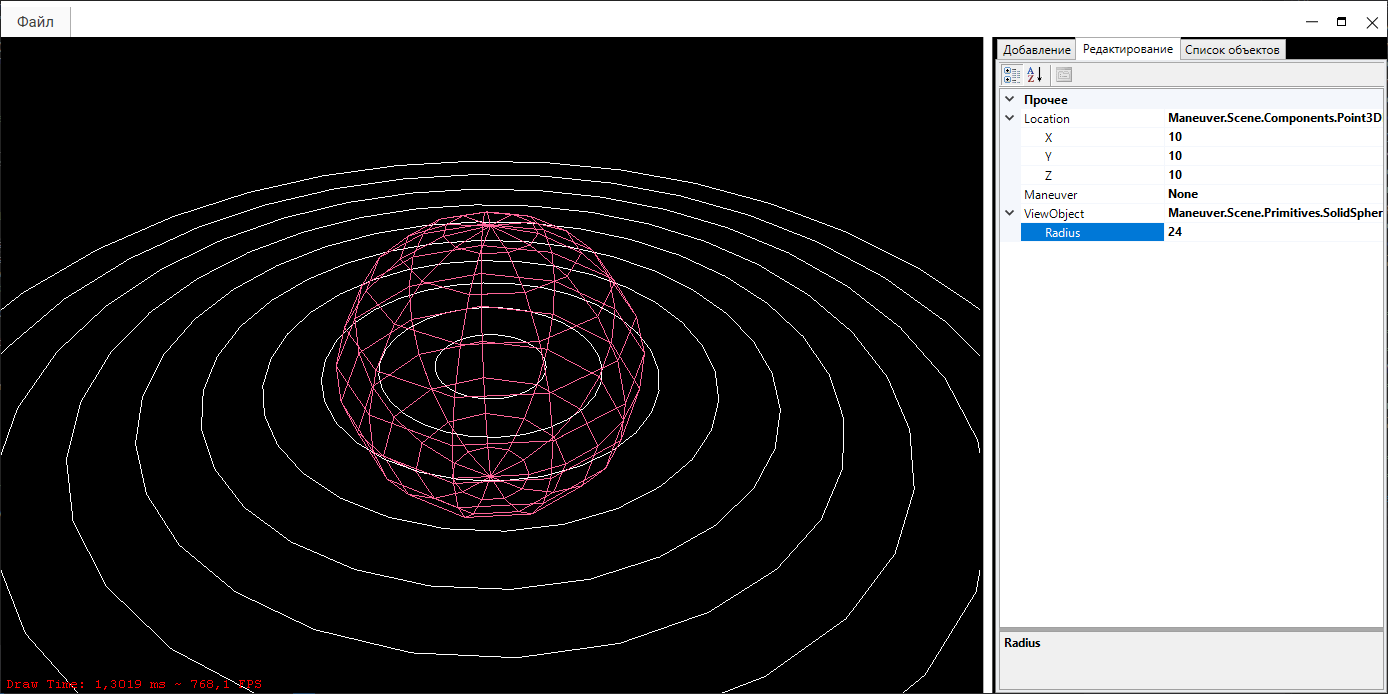


Рисунок 8 - Результат изменения данных в базе данных

5 Опытная эксплуатация

5.1 Ошибки, выявленные в процессе опытной эксплуатации

Ошибки и проблемы, которые возникли на стадии разработки средства, приведены в таблице 7.

Таблица 7 - Ошибки и их описание

|  |  |
| --- | --- |
| Классификация ошибок | Содержание |
| Ошибки загрузки файлов .obj | В программе возникли ошибки загрузки и преобразования .obj файлов. |
| Неверно работающая функциональность. | Функция «CalculateScaledNormalizedCoordinate» работает не так, как предусмотрено спецификацией. |
| Пропущенная информация. | Ошибка интерфейса: отсутствие признаков активности выполнения программы. |
| Ошибки отображения. | Вращение камеры работает правильно, но зачастую с ошибками в управлении при поворотах. |

Выводы

За время прохождения технологической практики были исследованы информационные потоки в ОАО «АЛЕВКУРП», было составлено техническое задание, разработан технический проект, проведено испытание программы.

При большом количестве объектов в сцене сложно выбрать нужный объект в списке, и модуль «Выбор объекта» позволяет быстро выполнять эту задачу.

Поставленные задачи выполнены.

Для обеспечения надежности функционирования программы была проведена опытная эксплуатация.

Программное средство можно модифицировать, как того пожелает заказчик.

В процессе разработки программы использовался в большом объеме материал по программированию и алгоритмизации, что способствовало закреплению наработанных навыков и умений в этих областях знаний.

Список используемых источников

1. Багласова Т.Г. Методические указания по выполнению дипломного проекта для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г.Багласова. – Минск: КБП, 2017
2. Багласова Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г.Багласова, К.О.Якимович. – Минск: КБП, 2015
3. Гради Буч. Объектно - ориентированный анализ и проектирование / Гради Буч. – 2-е изд. – М.: Бином, 1998
4. ГОСТ 2.105-95 ЕСКД. Общие требования к текстовым документам. – М.: Из-во стандартов, 1995
5. Web-технологии [Электронный ресурс]. – Web-технологии: HTML, DHTML, JavaScript, PHP, MySQL, XML+XLST, Ajax. – сор. 2008-2013 – Режим доступа: <http://htmlweb.ru/>
6. Первый современный компьютер [Электронный ресурс] – Первый рабочий компьютер времен второй мировой – сор. 2019 – Режим доступа: <https://ru.wikipedia.org/wiki/Colossus_(компьютер)>
7. Нашлемная индикация [Электронный ресурс] – Идея нашлемной индикации – сор. 2019 – Режим доступа: <https://ru.wikipedia.org/wiki/Индикатор_на_лобовом_стекле>
8. Радар [Электронный ресурс] – Технология радара – сор. 2019 – Режим доступа: <https://en.wikipedia.org/wiki/Radar>

Приложение А

(обязательное)

Текст программных модулей

using System;

using System.Collections.ObjectModel;

using System.Windows.Input;

using System.Windows.Threading;

using System.Windows;

using System.Threading;

using SharpGL;

using Maneuver.Helpers;

using Maneuver.Scene;

using Maneuver.Scene.Components;

using Maneuver.Scene.Components.Elements;

using Maneuver.Scene.Primitives;

using Maneuver.Scene.Primitives.ExtendedPrimitives;

using Maneuver.Scene.ManeuverManagement;

using System.Windows.Controls;

namespace Maneuver

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private bool isMenuDragMove = false;

ManeuverScene Scene { get; set; } = null;

ObservableCollection<ManeuverPoint> ManeuverPoints { get; set; } = null;

public MainWindow()

{

InitializeComponent();

Scene = new ManeuverScene(OpenGLView.OpenGL);

ManeuverPoints = new ObservableCollection<ManeuverPoint>();

Scene.SceneElements.Add(new Kompas(new Circle2DArray(100, 10), new Point3DFloat(0, 0, 0)));

ProgramStateHelper.OpenGLControl = OpenGLView;

SceneObjectsDataGrid.ItemsSource = ManeuverPoints;

}

#region window ui activity and animation

void CloseCommandExecute(object sender, RoutedEventArgs e)

{

this.Close();

}

void MinimizeCommandExecute(object sender, RoutedEventArgs e)

{

WindowState = WindowState.Minimized;

}

void ChangeStateCommandExecute(object sender, RoutedEventArgs e)

{

if (this.WindowState == WindowState.Maximized)

{

this.WindowState = WindowState.Normal;

}

else

{

this.WindowState = WindowState.Maximized;

}

}

private void Menu\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

if (WindowState == WindowState.Maximized)

WindowState = WindowState.Normal;

else WindowState = WindowState.Maximized;

}

private void Menu\_MouseMove(object sender, MouseEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed && isMenuDragMove)

{

if (WindowState == WindowState.Maximized)

WindowState = WindowState.Normal;

this.DragMove();

}

}

private void Menu\_MouseDown(object sender, MouseButtonEventArgs e) => isMenuDragMove = true;

private void Menu\_MouseUp(object sender, MouseButtonEventArgs e) => isMenuDragMove = false;

private void MainAppWindow\_StateChanged(object sender, EventArgs e)

=> ChangeWindowStateButton.Content = this.WindowState != WindowState.Maximized ? this.Resources["MaximizeWindowSimbol"] : this.Resources["MinimizeWindowSimbol"];

private void GridSplitter\_DragDelta(object sender, System.Windows.Controls.Primitives.DragDeltaEventArgs e)

{

this.Dispatcher.BeginInvoke(DispatcherPriority.Send,

(ThreadStart)delegate ()

{

this.OpenGLView.Refresh();

}

);

}

private void MainAppWindow\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.System)

e.Handled = true;

}

#endregion

private void AddNewMoveSphereClick(object sender, RoutedEventArgs e)

{

string x\_text = CreatePointXTextBox.Text.Replace('.', ','); // 12.5 -> 12,5

string y\_text = CreatePointYTextBox.Text.Replace('.', ','); // 12.5 -> 12,5

string z\_text = CreatePointZTextBox.Text.Replace('.', ','); // 12.5 -> 12,5

ManeuverType maneuver = (ManeuverType) PointManeuverTypeComboBox.SelectedItem;

// чуть ниже идут проверки на значения в полях

if(!x\_text.IsFloat())

{

MessageBox.Show("Значение в поле \"X\" не является числом", "Ошибка");

return;

}

if (!y\_text.IsFloat())

{

MessageBox.Show("Значение в поле \"Y\" не является числом", "Ошибка");

return;

}

if (!z\_text.IsFloat())

{

MessageBox.Show("Значение в поле \"Z\" не является числом", "Ошибка");

return;

}

float x = x\_text.ToFloat();

float y = y\_text.ToFloat();

float z = z\_text.ToFloat();

var maneuverPoint = new ManeuverPoint(new SolidSphere(7, 12, 12), new Point3DFloat(x, y, z));

Scene.SceneElements.Add(maneuverPoint); // add to scene

ManeuverPoints.Add(maneuverPoint); // add to UI

OpenGLView.Refresh();

}

private void OpenGLView\_OpenGLDraw(object sender, SharpGL.RenderEventArgs args)

{

// draw scene

Scene.Render();

// save info

}

private void OpenGLView\_MouseDown(object sender, System.Windows.Forms.MouseEventArgs e)

{

switch (e.Button)

{

case System.Windows.Forms.MouseButtons.Left:

case System.Windows.Forms.MouseButtons.Right:

case System.Windows.Forms.MouseButtons.Middle:

ProgramStateHelper.IsMouseKeyPressed = true;

ProgramStateHelper.LastPressedKey = e.Button;

ProgramStateHelper.LastMouseCursorPressedPosition = new Point(e.X, e.Y);

break;

}

}

private void OpenGLView\_MouseUp(object sender, System.Windows.Forms.MouseEventArgs e)

{

// не нужно реагировать на случай, когда какой-то дурак решил перетаскивание с другого UI-элемента сделать

if (ProgramStateHelper.LastPressedKey != e.Button) return;

switch (e.Button)

{

case System.Windows.Forms.MouseButtons.Left:

// если хотя бы одна из ранее зафиксированных координат мыши не совпадает с текущим, то и не надо производить выделение

if (ProgramStateHelper.LastMouseCursorPressedPosition.X != e.X || ProgramStateHelper.LastMouseCursorPressedPosition.Y != e.Y)

return;

// производим выделение объектов (hit test)

var selected = Scene.DoHitTest(e.X, e.Y, this.OpenGLView);

switch(selected)

{

case ManeuverPoint point:

CurrentSelectedObjectPropertyGrid.SelectedObject = point;

Scene.Camera.Target.Set(point.Location.X, point.Location.Y, point.Location.Z);

OpenGLView.Refresh();

break;

default:

CurrentSelectedObjectPropertyGrid.SelectedObject = null;

break;

}

ProgramStateHelper.IsMouseKeyPressed = false;

break;

case System.Windows.Forms.MouseButtons.Middle:

ProgramStateHelper.LastPressedKey = System.Windows.Forms.MouseButtons.None;

ProgramStateHelper.IsMouseKeyPressed = false;

break;

}

}

private void OpenGLView\_MouseMove(object sender, System.Windows.Forms.MouseEventArgs e)

{

// если клавиша мышы не задействована, то и нечего нам тут делать

if (!ProgramStateHelper.IsMouseKeyPressed) return;

// объяснение

// если нажаты alt, ctrl и колесико мышки, то выполняется вращение самого target

// также, если нажаты только если нажаты alt и колесико мышки, то выполняется вокруг текущего target

if (Keyboard.IsKeyDown(Key.LeftAlt) && Keyboard.IsKeyDown(Key.LeftCtrl) && ProgramStateHelper.LastPressedKey == System.Windows.Forms.MouseButtons.Middle)

{

// повторение: этот блок выполняет вращение самого target

}

else if (Keyboard.IsKeyDown(Key.LeftAlt) && ProgramStateHelper.LastPressedKey == System.Windows.Forms.MouseButtons.Middle)

{

// повторение: этот блок выполняет вращение вокруг target

// получаем предыдущие координаты мыши

var last\_mouse\_x = ProgramStateHelper.LastMouseCursorPressedPosition.X;

var last\_mouse\_y = ProgramStateHelper.LastMouseCursorPressedPosition.Y;

// получаем текущее положение мыши

var current\_mouse\_x = e.X;

var current\_mouse\_y = e.Y;

/\* вычисления + изменение камеры \*/

// инициализация относитьельного вектора (camera.position -> cameta.target) и вычисление длинны относительного вектора

Point3DFloat vertex = new Point3DFloat()

{

X = Scene.Camera.Position.X - Scene.Camera.Target.X,

Y = Scene.Camera.Position.Y - Scene.Camera.Target.Y,

Z = Scene.Camera.Position.Z - Scene.Camera.Target.Z

};

vertex.X += Scene.Camera.Target.X;

vertex.Y += Scene.Camera.Target.Y;

vertex.Z += Scene.Camera.Target.Z;

var radius = Math.Sqrt(vertex.X \* vertex.X + vertex.Y \* vertex.Y + vertex.Z \* vertex.Z);

// получение текущих theta и phi (подробнее: https://en.wikipedia.org/wiki/Spherical\_coordinate\_system)

var theta = Math.Acos(Scene.Camera.Position.Z / radius); // угол наклона по оси z (упрощенное объяснение)

var phi = Math.Atan2(Scene.Camera.Position.Y, Scene.Camera.Position.X); // угол наклона по оси xy (упрощенное объяснение)

// изменение theta и phi

theta -= (float)(current\_mouse\_y - last\_mouse\_y) \* ProgramStateHelper.MouseKoef;

phi -= (float)(current\_mouse\_x - last\_mouse\_x) \* ProgramStateHelper.MouseKoef;

// вычисление новых позиций x, y, z

var newX = SphericalSystemHelper.Calculate3DXPointPosition(radius, theta, phi);

var newY = SphericalSystemHelper.Calculate3DYPointPosition(radius, theta, phi);

var newZ = SphericalSystemHelper.Calculate3DZPointPosition(radius, theta);

Scene.Camera.Position.Set((float)newX, (float)newY, (float)newZ);

// апдейт текущего положения мыши

ProgramStateHelper.LastMouseCursorPressedPosition = new Point(e.X, e.Y);

OpenGLView.Refresh();

}

else if (ProgramStateHelper.LastPressedKey == System.Windows.Forms.MouseButtons.Middle)

{

// этот блок выполняет перемещение камеры и цели камеры (target-а) перпендикулярно

// не реализовано по причине хз как

}

}

private void OpenGLView\_MouseWheel(object sender, System.Windows.Forms.MouseEventArgs e)

{

var delta\_value = e.Delta;

if (delta\_value > 0)

{

float newX = SphericalSystemHelper.CalculateScaledCoordinate(Scene.Camera.Position.X, Scene.Camera.Target.X, 0.2f);

float newY = SphericalSystemHelper.CalculateScaledCoordinate(Scene.Camera.Position.Y, Scene.Camera.Target.Y, 0.2f);

float newZ = SphericalSystemHelper.CalculateScaledCoordinate(Scene.Camera.Position.Z, Scene.Camera.Target.Z, 0.2f);

// если мы в плонтную станем к обекту на который смотрим то не надо становится

if (Math.Truncate(Scene.Camera.Target.X) == Math.Truncate(newX) &&

Math.Truncate(Scene.Camera.Target.Y) == Math.Truncate(newY) &&

Math.Truncate(Scene.Camera.Target.Z) == Math.Truncate(newZ)

) return;

Scene.Camera.Position.Set(newX, newY, newZ);

OpenGLView.Refresh();

}

else

{

float newX = SphericalSystemHelper.CalculateScaledCoordinateNormalize(Scene.Camera.Position.X, Scene.Camera.Target.X, 1.1f);

float newY = SphericalSystemHelper.CalculateScaledCoordinateNormalize(Scene.Camera.Position.Y, Scene.Camera.Target.Y, 1.1f);

float newZ = SphericalSystemHelper.CalculateScaledCoordinateNormalize(Scene.Camera.Position.Z, Scene.Camera.Target.Z, 1.1f);

Scene.Camera.Position.Set(newX, newY, newZ);

OpenGLView.Refresh();

}

}

private void CurrentSelectedObjectPropertyGrid\_PropertyValueChanged(object s, System.Windows.Forms.PropertyValueChangedEventArgs e)

=> OpenGLView.Refresh();

private void PointManeuverTypeComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)

{

if (!(sender as ComboBox).IsVisible)

return;

var point = (sender as ComboBox).DataContext as ManeuverPoint;

point.Maneuver = (ManeuverType) e.AddedItems[0];

}

}

}

using System;

using System.Collections.Generic;

using SharpGL;

using Maneuver.Scene.Components;

using Maneuver.Helpers;

namespace Maneuver.Scene

{

public class ManeuverScene

{

private OpenGL OpenGLContext = default;

/// <summary>

/// Camera of <see cref="ManeuverScene"/>

/// </summary>

public Camera Camera { get; set; } = new Camera(new Point3DFloat(100, 100, 100), new Point3DFloat(0, 0, 0));

/// <summary>

/// <see cref="ManeuverScene"/> elements

/// </summary>

public List<SceneElement> SceneElements { get; set; } = new List<SceneElement>();

/// <summary>

/// Init <see cref="ManeuverScene"/> with <see cref="OpenGL"/> context

/// </summary>

/// <param name="openGLContext">It is <see cref="OpenGL"/> context</param>

/// <exception cref="ArgumentNullException"></exception>

public ManeuverScene(OpenGL openGLContext)

=> this.OpenGLContext = openGLContext != null ? openGLContext : throw new ArgumentNullException("OpenGL context is null");

/// <summary>

/// Perform the render of <see cref="ManeuverScene"/>

/// </summary>

public void Render()

{

var gl = OpenGLContext;

gl.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT);

gl.MatrixMode(SharpGL.Enumerations.MatrixMode.Projection);

gl.LoadIdentity();

int[] viewport\_size = new int[4];

gl.GetInteger(OpenGL.GL\_VIEWPORT, viewport\_size);

float x = (float)viewport\_size[2] / viewport\_size[3];

gl.Perspective(45, x, 0.1, double.MaxValue);

Camera.LookAt(gl);

foreach (SceneElement element in SceneElements)

element.Render(gl);

gl.Flush();

}

/// <summary>

/// Perform color-selection

/// </summary>

/// <param name="mouse\_x">mouse X position</param>

/// <param name="mouse\_y">mouse Y position</param>

/// <param name="viewport">Original <see cref="OpenGLControl"/></param>

/// <returns>Selected <see cref="SceneElement"/></returns>

public SceneElement DoHitTest(int mouse\_x, int mouse\_y, OpenGLControl viewport)

{

var colorsElements = this.CreateSelectColors();

var clickedColor = ColorPickerHelper.PerformSelection(viewport, colorsElements, this.Camera, mouse\_x, mouse\_y);

return colorsElements.ContainsKey(clickedColor) ? colorsElements[clickedColor] : null;

}

/// <summary>

/// Create ditionary of scene elements and colors

/// </summary>

/// <returns></returns>

private Dictionary<System.Windows.Media.Color, SceneElement> CreateSelectColors()

{

var pairs = new Dictionary<System.Windows.Media.Color, SceneElement>();

byte r = 0, g = 0, b = 1;

const byte alpha = 1;

foreach (var element in this.SceneElements)

{

if (++b == 0)

if (++g == 0)

if (++r == 0)

// происходит только в случае 16.5 млн объектов-сфер в сцене

throw new Exception("you have reached the limit of colors");

pairs.Add(new System.Windows.Media.Color() { R = r, G = g, B = b, A = alpha }, element);

}

return pairs;

}

}

}

using System;

using SharpGL;

using Maneuver.Scene.Primitives;

using System.ComponentModel;

namespace Maneuver.Scene.Components

{

/// <summary>

/// Represents the element of <seealso cref="ManeuverScene"/>

/// </summary>

public abstract class SceneElement : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropetyChanged(PropertyChangedEventArgs e) => PropertyChanged?.Invoke(this, e);

private Point3DFloat location = default;

private IRenderablePrimitive viewObject = default;

/// <summary>

/// Set <see cref="SceneElement.ViewObject"/> in <paramref name="viewObject"/> and <see cref="SceneElement.Location"/> in (x, y, z) = (0, 0, 0)

/// </summary>

/// <param name="viewObject">Renderable object of <see cref="SceneElement"/></param>

/// /// <exception cref="ArgumentNullException"></exception>

public SceneElement(IRenderablePrimitive viewObject)

{

if (viewObject is null)

throw new ArgumentNullException();

this.ViewObject = viewObject;

this.Location = new Point3DFloat(0, 0, 0);

}

/// <summary>

/// Set <see cref="SceneElement.ViewObject"/> in <paramref name="viewObject"/> and <see cref="SceneElement.Location"/> in <paramref name="location"/>

/// </summary>

/// <param name="viewObject"><see cref="SceneElement.ViewObject"/> of <see cref="SceneElement"/></param>

/// <param name="location"><see cref="SceneElement.Location"/> of <see cref="SceneElement"/></param>

/// <exception cref="ArgumentNullException"></exception>

public SceneElement(IRenderablePrimitive viewObject, Point3DFloat location)

{

if (viewObject is null || location is null)

throw new ArgumentNullException();

this.ViewObject = viewObject;

this.Location = location;

}

/// <summary>

/// The location of element

/// </summary>

[TypeConverter(typeof(ExpandableObjectConverter))]

public Point3DFloat Location

{

get => location;

set

{

if (location == value)

return;

location = value;

OnPropetyChanged(new PropertyChangedEventArgs("Location"));

}

}

/// <summary>

/// The rendered primitive of the scene element

/// </summary>

[TypeConverter(typeof(ExpandableObjectConverter))]

public IRenderablePrimitive ViewObject

{

get => viewObject;

set

{

if (viewObject == value)

return;

viewObject = value;

OnPropetyChanged(new PropertyChangedEventArgs("ViewObject"));

}

}

/// <summary>

/// Perform the render of element. Call <see cref="IRenderablePrimitive.Draw"/> of <see cref="SceneElement.ViewObject"/> object

/// </summary>

/// <param name="context">It is <see cref="OpenGL"/> context</param>

public void Render(OpenGL context)

=> ViewObject.Draw(context, Location);

/// <summary>

/// Perform color-pick render

/// </summary>

/// <param name="context">It is <see cref="OpenGL"/> context</param>

/// <param name="color">Color-pick color</param>

public void ColorPickRender(OpenGL context, System.Windows.Media.Color color)

{

byte[] rgba = { color.R, color.G, color.B, color.A };

this.ColorPickRender(context, rgba, Location);

}

/// <summary>

/// Perform color-pick render

/// </summary>

/// <param name="context">It is <see cref="OpenGL"/> context</param>

/// <param name="rgba">It is array {r, g, b, a} array</param>

public void ColorPickRender(OpenGL context, byte[] rgba, Point3DFloat location)

{

if(ViewObject is ISelectedPrimitive selectModeView)

selectModeView.ColorPickDraw(context, rgba, Location);

}

}

}

using System;

using System.ComponentModel;

using SharpGL;

using Maneuver.Scene.Components;

namespace Maneuver.Scene.Primitives

{

public class SolidSphere : IRenderablePrimitive, ISelectedPrimitive, INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropetyChanged(PropertyChangedEventArgs e) => PropertyChanged?.Invoke(this, e);

private float[] vertices;

private float[] normals;

private float[] texcoords;

private uint[] indices;

private float[] colors;

private float \_radius;

private int \_rings, \_sectors;

public float Radius

{

get => \_radius;

set

{

if (\_radius == value)

return;

\_radius = value;

this.Generate(\_radius, \_rings, \_sectors);

OnPropetyChanged(new PropertyChangedEventArgs("Radius"));

}

}

/// <summary>

/// Init sphere with preset radius, ringss and sectors

/// </summary>

/// <param name="radius"></param>

/// <param name="rings"></param>

/// <param name="sectors"></param>

/// <param name="location"></param>

public SolidSphere(float radius, int rings, int sectors)

{

\_sectors = sectors;

\_rings = rings;

\_radius = radius;

Generate(radius, rings, sectors);

}

/// <summary>

/// Perform color-only draw

/// </summary>

/// <param name="context">It is <see cref="OpenGL"/> context</param>

/// <param name="rgba">It is {r, g, b, a} array</param>

/// <param name="location">Location for render</param>

public void ColorPickDraw(OpenGL context, byte[] rgba, Point3DFloat location)

{

var gl = context;

byte[] pick\_colors = new byte[colors.Length];

Maneuver.Helpers.PerformHelper.FillArrayByArray(ref pick\_colors, rgba);

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_FILL);

gl.PushMatrix();

gl.Translate(location.X, location.Y, location.Z);

gl.Rotate(90, 0, 0);

gl.EnableClientState(OpenGL.GL\_VERTEX\_ARRAY);

gl.EnableClientState(OpenGL.GL\_NORMAL\_ARRAY);

gl.EnableClientState(OpenGL.GL\_TEXTURE\_COORD\_ARRAY);

gl.EnableClientState(OpenGL.GL\_COLOR\_ARRAY);

unsafe

{

fixed (byte\* p = pick\_colors)

{

IntPtr ptr = (IntPtr)p;

gl.ColorPointer(4, OpenGL.GL\_UNSIGNED\_BYTE, 0, ptr);

}

}

gl.VertexPointer(3, 0, vertices);

gl.NormalPointer(OpenGL.GL\_FLOAT, 0, normals);

gl.TexCoordPointer(2, OpenGL.GL\_FLOAT, 0, vertices);

// gl.DrawElements(OpenGL.GL\_QUADS, indices.Length, indices); - в идеале, но почему-то баги

gl.DrawElements(OpenGL.GL\_QUADS, indices.Length - 50, indices);

gl.DisableClientState(OpenGL.GL\_VERTEX\_ARRAY);

gl.DisableClientState(OpenGL.GL\_NORMAL\_ARRAY);

gl.DisableClientState(OpenGL.GL\_TEXTURE\_COORD\_ARRAY);

gl.PopMatrix();

}

/// <summary>

/// Renders an object in the specified <see cref="OpenGL"/> context

/// </summary>

/// <param name="context">It is <see cref="OpenGL"/> context</param>

/// <param name="location">Location for rende</param>

public void Draw(OpenGL context, Point3DFloat location)

{

var gl = context;

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_LINE);

gl.PushMatrix();

gl.Translate(location.X, location.Y, location.Z);

gl.Rotate(90, 0, 0);

gl.EnableClientState(OpenGL.GL\_VERTEX\_ARRAY);

gl.EnableClientState(OpenGL.GL\_NORMAL\_ARRAY);

gl.EnableClientState(OpenGL.GL\_TEXTURE\_COORD\_ARRAY);

gl.EnableClientState(OpenGL.GL\_COLOR\_ARRAY);

unsafe

{

fixed (float\* p = colors)

{

IntPtr ptr = (IntPtr)p;

gl.ColorPointer(4, OpenGL.GL\_FLOAT, 0, ptr);

}

}

gl.VertexPointer(3, 0, vertices);

gl.NormalPointer(OpenGL.GL\_FLOAT, 0, normals);

gl.TexCoordPointer(2, OpenGL.GL\_FLOAT, 0, vertices);

// gl.DrawElements(OpenGL.GL\_QUADS, indices.Length, indices); - в идеале, но почему-то баги

gl.DrawElements(OpenGL.GL\_QUADS, indices.Length - 50, indices);

gl.DisableClientState(OpenGL.GL\_VERTEX\_ARRAY);

gl.DisableClientState(OpenGL.GL\_NORMAL\_ARRAY);

gl.DisableClientState(OpenGL.GL\_TEXTURE\_COORD\_ARRAY);

gl.PopMatrix();

}

/// <summary>

/// Perform generate

/// </summary>

/// <param name="radius"></param>

/// <param name="rings"></param>

/// <param name="sectors"></param>

private void Generate(float radius, int rings, int sectors)

{

float m\_pi\_2 = (float)Math.PI / 2f;

float R = 1f / (rings - 1);

float S = 1f / (sectors - 1);

int r = 0, s = 0;

vertices = new float[rings \* sectors \* 3];

normals = new float[rings \* sectors \* 3];

texcoords = new float[rings \* sectors \* 2];

int v\_index = 0, n\_index = 0, t\_index = 0;

for (r = 0; r < rings; r++)

{

for (s = 0; s < sectors; s++)

{

float y = (float)Math.Sin(-m\_pi\_2 + Math.PI \* r \* R);

float x = (float)(Math.Cos(2f \* Math.PI \* s \* S) \* Math.Sin(Math.PI \* r \* R));

float z = (float)(Math.Sin(2f \* Math.PI \* s \* S) \* Math.Sin(Math.PI \* r \* R));

texcoords[t\_index++] = s \* S;

texcoords[t\_index++] = r \* R;

vertices[v\_index++] = x \* radius;

vertices[v\_index++] = y \* radius;

vertices[v\_index++] = z \* radius;

normals[n\_index++] = -x;

normals[n\_index++] = -y;

normals[n\_index++] = -z;

}

}

indices = new uint[rings \* sectors \* 4];

int in\_index = 0;

for (r = 0; r < rings; r++)

{

for (s = 0; s < sectors; s++)

{

indices[in\_index++] = (uint)((r + 1) \* sectors + s);

indices[in\_index++] = (uint)((r + 1) \* sectors + (s + 1));

indices[in\_index++] = (uint)(r \* sectors + (s + 1));

indices[in\_index++] = (uint)(r \* sectors + s);

}

}

colors = new float[vertices.Length \* 4];

for (int i = 0; i < colors.Length; i += 4)

{

/\*colors[i] = 0f;

colors[i + 1] = 1f;

colors[i + 2] = 1f;

colors[i + 3] = 1f;\*/

colors[i] = 1f;

colors[i + 1] = 0.4f;

colors[i + 2] = 0.6f;

colors[i + 3] = 1f;

}

}

}

}

using System;

using SharpGL;

using Maneuver.Scene.Components;

namespace Maneuver.Scene.Primitives

{

/// <summary>

/// Represents the circle of <see cref="ManeuverScene"/>/>

/// </summary>

public class Circle : IRenderablePrimitive

{

/// <summary>

/// Vertex array

/// </summary>

private Point3DFloat[] vertexes = null;

public Circle() { }

/// <summary>

/// Creates a circle with a given radius and number of sides

/// </summary>

/// <param name="radius">radius of circle</param>

/// <param name="slides">number of sides</param>

public Circle(float radius, int slides)

=> Generate(radius, slides);

/// <summary>

/// Generate a circle with a given radius and number of sides and set to this object

/// </summary>

/// <param name="radius">radius of circle</param>

/// <param name="slides">number of sides</param>

public void Generate(float radius, int slides)

{

vertexes = new Point3DFloat[slides];

for (int i = 0; i < slides; i++)

{

float theta = 2.0f \* (float)Math.PI \* i / (float)slides;

float x = radius \* (float)Math.Cos(theta);

float y = radius \* (float)Math.Sin(theta);

Point3DFloat vertex = new Point3DFloat()

{

X = x,

Y = y,

Z = 0

};

vertexes[i] = vertex;

}

}

public void Draw(OpenGL context, Point3DFloat location)

{

if (vertexes == null)

return;

var gl = context;

gl.Translate(location.X, location.Y, location.Z);

gl.Begin(OpenGL.GL\_LINE\_LOOP);

for (int i = 0; i < vertexes.Length; i++)

{

gl.Color(1f, 1f, 1f);

gl.Vertex(vertexes[i].X, vertexes[i].Y, vertexes[i].Z);

}

gl.End();

}

}

}