МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ТВЕРСКОЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

**(ФГБОУ ВПО «ТвГУ»)**

Математический факультет

Кафедра Компьютерной безопасности и

математических методов управления

**ОТЧЕТ**

 по производственнойпрактике

студента 4 курса группы М-45

Ксенофонтовой Анастасии Александровны

Специальность 10.05.01 Компьютерная безопасность

Специализация «Математические методы защиты информации»

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_.\_\_.20\_\_

*Подпись, расшифровка, ученая степень, звание*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.\_\_.20\_\_

*Подпись, расшифровка подписи*

Руководитель практики от предприятия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.\_\_.20\_\_

*Подпись, расшифровка подписи, ученая степень, звание*

Тверь 2017

Оглавление

[Оглавление 2](#_Toc487193020)

[Введение 3](#_Toc487193021)

[1. О предприятии 3](#_Toc487193022)

[2. Задание практики 5](#_Toc487193023)

[2.1. Порядок отработки карты инженерной оценки местности 7](#_Toc487193024)

[2.2. Оценка дорожной сети 8](#_Toc487193025)

[2.3. Методы решения задачи «Инженерная оценка местности» 9](#_Toc487193026)

[3. Результаты 11](#_Toc487193027)

[4. Приложение 12](#_Toc487193028)

[Список литературы 30](#_Toc487193029)

# Введение

В период с 12.06.2017 по 9.07.2017 я проходила практику в Открытом Акционерном Обществе Научно производственном объединении «Русские базовые информационные технологии» (далее ОАО НПО «РусБИТех»).

Моим руководителем был начальник отдела Прикладного моделирования Новоселов Павел Вячеславович. Во время практики я ознакомилась с внутренним распорядком предприятия, режимом работы, уставом предприятия.

# О предприятии

ОАО «НПО РусБИТех» осуществляет лицензированную разработку, производство и внедрение комплексных тренажерных систем нового поколения, информационных и автоматизированных систем, систем поддержки принятия решений, отечественных программных средств общего назначения, разработку и создание средств защиты информации и телекоммуникационных средств.

Направления деятельности

* Создание, производство и внедрение комплексных тренажерных систем нового поколения, функционирующих в едином виртуальном пространстве (поле боя), способных информационно взаимодействовать не только между собой, но и с реальными образцами вооружения, дооборудованными средствами имитации и сопряжения.
* Создание, производство и внедрение центров боевой подготовки, позволяющих проводить весь комплекс подготовки военнослужащих от одиночной (индивидуальной) до слаживания экипажей подразделений и совместной подготовки органов военного управления (ОВУ) различного уровня.
* Создание, производство и внедрение ситуационно-аналитических центров, обеспечивающих решение широкого спектра задач планирования и управления развитием сложных систем с учетом межведомственного характера решаемых задач, особенностей существующих и перспективных средств вооруженной борьбы и динамики изменения обстановки.
* Создание, производство и внедрение систем поддержки принятия решений в мобильном и стационарном исполнении для автоматизированных систем управления различного назначения.
* Создание, производство и внедрение средств инфотелекоммуникационных систем (мультисервисные маршрутизаторы, коммутаторы, межсетевые экраны, средства сетевого управления, VoIP шлюзы с функцией АТС).
* Создание, производство и внедрение общего программного обеспечения (операционные системы общего и специального назначения с интегрированными в них офисными приложениями, защищенной системой управления базами данных и другими программными средствами в защищенном исполнении), аппаратно-программных средств защиты информации от несанкционированного доступа, в том числе аппаратно-программных модулей доверенной загрузки и модулей криптографической защиты информации, выполненных в виде плат расширения.
* Создание, производство и внедрение многофункциональных адаптируемых программно-аппаратных комплексов, включающих средства вычислительной техники с установленными общим и специальным программным обеспечением и средствами защиты информации, средства построения структурированной локальной вычислительной сети и телекоммуникационные средства.
* Подготовка специалистов в области передовых компьютерных технологий.
* Системная интеграция и внедрение высокотехнологичных комплексных решений.

# Задание практики

В ходе прохождения практики мною были получены следующие задания:

1. Для информационно-расчетной (ИРЗ) задачи «Инженерная оценка местности» добавить задачу по расчету характеристик дорожной сети.
2. Для информационно-расчетной задачи «Инженерная оценка местности» добавить задачу по расчету характеристик гидротехнических сооружений (ГТС).

При решении ИРЗ «Инженерная оценка местности» необходимо реализовать вывод на отображение в выбранном районе расчетных характеристик сети дорог и дорожно-мостовых сооружений на них и их отображение в соответствующих полях диалогового окна задачи.

Выходная информация должна выводиться после вызова ИРЗ «Инженерная оценка местности», выбора района и вызова ПКМ для этого района контекстного меню, в котором к уже имеющимся командам добавить **«Характеристики дорожной сети»** и **«Характеристики ГТС».**

После активации команды **«Характеристики дорожной сети»** должен отобразиться диалог с закладками «Характеристики дорог» и « Характеристики мостов».

Вид закладки «Характеристики дорог»:

|  |
| --- |
| Характеристика дорог |
| Номер дороги | <значение> |
| Ширина, м | <значение> |
| Ширина проезжей части, м | <значение> |
| Материал покрытия | <значение> |

Вид закладки «Характеристики мостов»:

|  |
| --- |
| Характеристика мостов |
| Материал сооружения | <значение> |
| Длина, м | <значение> |
| Ширина проезжей части, м | <значение> |
| Грузоподъемность, т | <значение> |
| Высота проезда, высота над уровнем воды | <значение> |

После активации команды **«Характеристики ГТС»** должен отобразиться диалог «Характеристики водохранилищ».

Вид закладки «Характеристики водохранилищ»

|  |
| --- |
| Характеристики водохранилищ |
| Название | <значение> |
| Объем,км3 | <значение> |
| Площадь зеркала, км2 | <значение> |
| Плотина |
| Материал | <значение> |
| Состояние | <значение> |
| Длина, м | <значение> |
| Ширина по верху, м | <значение> |
| Материал водосливной части | <значение> |
| Длина водосливной части | <значение> |
| Отметка верхнего уровня | <значение> |
| Отметка нижнего уровня | <значение> |

## Порядок отработки карты инженерной оценки местности

Изучение местности по карте осуществляется примерно в том же порядке, что и в процессе рекогносцировки. Сначала производят общий осмотр карты в полосе предстоящих действий с целью выявить тип и характерные особенности местности. Затем более детально изучают основные тактические свойства местности: условия проходимости, защиты войск, маскировки, водообеспечения, фортификационного оборудования и др.

В ряде случаев общий характер местности изучают по карте более мелкого масштаба, а тактические свойства оцениваются по карте более крупного масштаба, на которой все местные предметы изображены с большой подробностью и точностью.

Топографические карты создаются заблаговременно и поэтому могут несколько устареть. Поэтому вторым документом о местности, постепенно вытесняющим обычные топографические карты являются электронные карты. Отражая современное состояние местности, они подходят как для обзорного ее изучения, так и для детального изучения на ней отдельных объектов. Кроме того, для изучения отдельных тактических свойств местности могут использоваться специальные цифровые карты, например условий водообеспеченности местности, условий производства фортификационных работ или схемы проходимости местности, почвогрунтов и др.

Спецификация прецедента «Инженерная оценка местности»

| Элемент спецификации | Описание элемента спецификации |
| --- | --- |
| *Имя прецедента* | Инженерная оценка местности |
| *Краткое описание* | Прецедент реализует функцию автоматизированной оценку физико-географических условий местности по данным заложенным в ЦКМ |
| *Актер* | Оператор  |
| *Предусловия* | Должен быть загружен в ППС ММВД вариант оперативной обстановки и вызван интерфейс задачи |
| *Основной поток* | Оператор выбирает на цифровой карте участок местности (операционного направления). Система производит выборку характеристик подстилающей поверхности выбранного участка (операционного направления), производит качественную и количественную оценку и выводит ее на отображение в информационном окне задачи |
| *Альтернативные потоки* | Нет |
| *Постусловия* | Нет |

## Оценка дорожной сети

Оценка дорожной сети в целом является элементом оценки местности, осуществляемой в процессе оценки обстановки и способствует принятию объективного и рационального решения на выполнение поставленных задач.

Кроме оценки дорожной сети, оценке (особенно при решении задач инженерного оборудования территории) подвергаются: местная производственная база гражданских дорожных организаций, возможности этих организаций по строительству и ремонту автомобильных дорог, наличие местных дорожно-строительных материалов и т.п.

Следует подчеркнуть, необходимость анализа всей дорожной сетив пределах заданной (или обосновываемой) полосы (района), так как немаловажным вопросом является степень ее использования, во многом определяющая обеспечиваемую надежность передвижений, наличие резервных путей, протяженность и частоту возможных обходов и т.п.

Дополнительными задачами, решаемыми при военно-инженерной оценке дорожной сети в полосах (районах), является определение степени скрытности передвижений войск, обеспечиваемой сетью дорог (комплексом маршрутов), а также оценка потенциальных объемов заграждений и разрушений на путях.

Оценка дорожной сети района (полосы) может осуществляться в трех аспектах.

*Оценка всей дорожной сети полосы (района)* – плотность автомобильных дорог, предельное количество и состав маршрутов, средняя скорость движения, пропускная способность дорожной сети.

*Оценка отдельных дорог* – состав маршрута, средняя допустимая скорость движения, пропускная способность дороги, срок службы дорожной одежды.

*Оценка проходимости местности вне дорог* – скорость движения, максимально преодолеваемый уклон местности, количество машин, которые могут быть пропущены по одному следу.

## Методы решения задачи «Инженерная оценка местности»

Порядок работы задачи «Инженерная оценка местности» представлен в виде диаграммы активности на рис. 1.

По условиям проходимости местность подразделяется на: легкопроходимую, проходимую, труднопроходимую, непроходимую (таблица 1).

Таблица 1 – Коэффициенты проходимости вне дорог

|  |  |  |
| --- | --- | --- |
| Местность | В обычныхусловиях | В районахразрушений |
| Легкопроходимая для всех видов транспорта (равнинная, открытая, слабопересеченная) | 1,0 | 0,7 |
| Проходимая (холмистая, залесенная на 20-30 %, среднепересеченная) | 0,8 | 0,5 |
| Труднопроходимая (заболоченная, залесенная до 60 %, низкогорная) | 0,5 | 0,3 |
| Непроходимая (скалистые горы, непроходимые болота, плавни и т.п.) | 0,1 | 0,1 |

Основными факторами, определяющими проходимость местности вне дорог, являются рельеф, растительность, гидрографическая сеть и почвогрунты, особенно в период распутицы.

По степени пересеченности местность характеризуется как сильнопересеченная, среднепересеченная, слабопересеченная и непересеченная**.**



Рис. 1 – Диаграмма активности задачи «Инженерная оценка местности»

Сильнопересеченная изобилует большим количеством (свыше 30 % всей площади) труднопроходимых естественных препятствий (балки, овраги, горы, реки, озера, болота), ограничивая маневр и скорость движения подразделений. Такая местность позволяет применять тяжелую боевую технику только на отдельных направлениях и требует выполнения значительных объемов инженерных задач.

По среднепересеченной местности (менее 30 % препятствий) массированное применение техники возможно, но затруднено на отдельных направлениях.

По слабопересеченной (до 10 % естественных препятствий) и непересеченной местности хотя и снижена скорость перемещения, однако возможно применение маневра в любом направлении.

В результате определения общего характера местности делают вывод о доступности района и отдельных его направлений.

# Результаты

На некотором военном формировании (ВФ) вызывается ИРЗ «Инженерная оценка местности», выделяется некоторая область на карте и формируется район для оценки. На выделенном районе вызывается контекстное меню, в списке которого находим пункты «характеристика дорожной сети» и «характеристика гидротехнических сооружений» (рис. 2, приложение).

При вызове задачи «характеристика дорожной сети» получаем выходную форму, изображенную на рис. 3 и рис. 4 (приложение). На рис. 3 и рис. 4 при щелчке левой кнопкой мыши на нужный участок в таблице выходной формы на карте выделяется соответствующий характеристикам объект (красной линией подсветки). Все объекты дорожной сети на карте выделяются коричневым цветом заливки.

Аналогично, на рис. 5 и рис. 6 изображена выходная форма задачи «характеристика ГТС». На рис. 5 выделен объект – водохранилище (красным цветом контура), а на рис. 6 – одна из плотин этого водохранилища (также красным цветом). В этом случае объекты гидрографии на карте выделены синим цветом заливки.

На рис. 7 изображена выходная форма для проходимости местности и общая характеристика местности.

# Приложение

Исходный код программы:

 /// Класс характеристик дорожной сети

 class CDlgPropertiesRoadNet : public CSBaseDialog

 {

 DECLARE\_NO\_COPY\_CLASS(CDlgPropertiesRoadNet)

 public:

 CDlgPropertiesRoadNet(CSBaseWindow\* pParent\_, const SRegionTMM& RegionTMM\_);

 virtual ~CDlgPropertiesRoadNet() {};

 /// Отображает закрытость местности

 void Show(CGisMapWindow \*pWnd\_);

 protected:

 long OnCreate();

 long OnDestroy();

 void OnSize(UINT, int, int);

 void OnChangeRow();

 private:

 //\*\*\*

 SRegionTMM m\_RegionTMM;

 //\*\*\*

 CDrawOnGisWnd m\_DrawOnGisWnd;

 CVector<TContours> m\_vContours;

 CSGrid m\_grid;

 int m\_nFixedRow;

 bm::vector<TgisData> m\_vGisData;

 void fillGridHead();

 void fillGridData();

 };

 //=======================================================================================

 CDlgPropertiesRoadNet::CDlgPropertiesRoadNet(CSBaseWindow \*pParentWnd\_, const SRegionTMM& RegionTMM\_)

 : CSBaseDialog(DLG\_TERRAIN\_PARAMS, pParentWnd\_, g\_hInstance), m\_RegionTMM(RegionTMM\_)

 {

 m\_nFixedRow = 1;

 m\_DrawOnGisWnd.SetParams(this, GetIdActiveWnd(), (CDrawOnGisWnd::TDrawHandler)&CDlgPropertiesRoadNet::Show);

 IFace\_Attach(IFD\_GIS, this);

 EnableSerialize(true);

 }

 //--------------------------------------------------------------------------

 long CDlgPropertiesRoadNet::OnCreate()

 {

 SetText(\_S("Характеристики дорожной сети") + " [" + m\_RegionTMM.m\_sName + "]");

 m\_grid.SubclassDlgItem(IDC\_GRID, this);

 m\_grid.SetEditable(false);

 ON\_GRID\_SELCHANGE(IDC\_GRID, &CDlgPropertiesRoadNet::OnChangeRow);

 fillGridHead();

 fillGridData();

 std::set<int> typesSpecified;

 typesSpecified.insert(UT\_ROAD);

 typesSpecified.insert(UT\_BRIDGE);

 GetContoursBySurfaceStateTypes(m\_RegionTMM, typesSpecified, m\_vContours);

 RepaintServiceModellingEnvironment(m\_DrawOnGisWnd.GetGisWndId());

 ShakeWindow();

 return 1L;

 }

 //--------------------------------------------------------------------------

 long CDlgPropertiesRoadNet::OnDestroy()

 {

 m\_vContours.DelAllItems();

 RepaintServiceModellingEnvironment(m\_DrawOnGisWnd.GetGisWndId());

 CSP<ICtrlGis> CtrlGis;

 if (Get\_Interface(CtrlGis))

 if (CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd()))

 {

 SUUID wndId = pWnd->getUserWndId();

 CtrlGis->SelectObjectById(wndId, UNKNOWN\_EXTGOBJKEY, true);//Очистка отселектированных объектов в отселектированных типах

 CtrlGis->updateMapImage(wndId, gis::GetDefaultSettingsForRedrawDynamicLayers());

 }

 IFace\_Detach(IFD\_GIS, this);

 CSBaseDialog::OnDestroy();

 delete this;

 return 1;

 }

 //--------------------------------------------------------------------------

 void CDlgPropertiesRoadNet::OnSize(UINT, int, int)

 {

 if (IsInitialized())

 {

 RECT rc;

 GetClientRect(&rc);

 const int nBorder = 4;

 m\_grid.Move(nBorder, nBorder, rc.right - nBorder \* 2, rc.bottom - nBorder \* 2);

 m\_grid.AutoSize(true);

 }

 }

 //--------------------------------------------------------------------------

 void CDlgPropertiesRoadNet::fillGridHead()

 {

 m\_grid.SetColumnCount(2);

 m\_grid.SetFixedRowCount(m\_nFixedRow);

 m\_grid.SetItemAsText(0, 0, \_S("Основные характеристики"));

 m\_grid.SetItemAsText(0, 1, \_S("Значения"));

 }

 //--------------------------------------------------------------------------

 void CDlgPropertiesRoadNet::fillGridData()

 {

 SetWaitCursor();

 //Слой (ДОРОЖНАЯ СЕТЬ) LAYER8

 CVector<STR> vType;

 vType.AddItem(\_S("L0061220000")); // АВТОДОРОГА УСОВЕРШЕНСТВОВАННЫМ ПОКРЫТИЕМ (усовершенствованное шоссе)

 vType.AddItem(\_S("L0061230000")); // АВТОДОРОГА С ПОКРЫТИЕМ (шоссе)

 vType.AddItem(\_S("L0061310000")); // АВТОДОРОГА БЕЗ ПОКРЫТИЯ (улучшенная грунтовая дорога)

 CVector<TpairSS> vSem;

 vSem.AddItem(make\_pair(\_S("SEM53"), \_S("Номер дороги")));

 vSem.AddItem(make\_pair(\_S("SEM11"), \_S("Ширина, м")));

 vSem.AddItem(make\_pair(\_S("SEM46"), \_S("Ширина проезжей части, м")));

 vSem.AddItem(make\_pair(\_S("SEM55"), \_S("Материал покрытия")));

 m\_vGisData.resize(3);

 fillData(vType, vSem, m\_RegionTMM.m\_vAreaKrds, \_S(""), m\_vGisData[0]);

 int nIdx = m\_grid.InsertRow();

 m\_grid.SetItemAsExpanded(nIdx, 0, nIdx, 1, \_S("Характеристики дорог"));

 m\_grid.SetRowBkColour(nIdx, SLBROWN);

 fillGrid(m\_vGisData[0], m\_grid);

 bm::vector<STR> vRowName;

 vRowName.push\_back(\_S("Характеристики путепроводов"));

 vRowName.push\_back(\_S("Характеристики мостов"));

 bm::vector<CVector<STR>> vTypes;

 // Слой (ДОРОЖНЫЕ СООРУЖЕНИЯ) LAYER9

 vType.DelAllItems();

 vType.AddItem(\_S("S0062324000")); // ПУТЕПРОВОДЫ (прочие)

 vType.AddItem(\_S("L0062324000"));

 vType.AddItem(\_S("V0062324000"));

 vTypes.push\_back(vType);

 vType.DelAllItems();

 vType.AddItem(\_S("P0062315000")); // МОСТЫ (прочие)

 vType.AddItem(\_S("L0062315000"));

 vType.AddItem(\_S("V0062315000"));

 vType.AddItem(\_S("S0062315000"));

 vTypes.push\_back(vType);

 vSem.DelAllItems();

 vSem.AddItem(make\_pair(\_S("SEM10"), \_S("Материал сооружения")));

 vSem.AddItem(make\_pair(\_S("SEM2"), \_S("Длина, м")));

 vSem.AddItem(make\_pair(\_S("SEM46"), \_S("Ширина проезжей части, м")));

 vSem.AddItem(make\_pair(\_S("SEM12"), \_S("Грузоподъемность, т")));

 vSem.AddItem(make\_pair(\_S("SEM87"), \_S("Высота проезда, высота над уровнем воды, м")));

 for (size\_t i = 0; i < vRowName.size(); i++)

 {

 fillData(vTypes[i], vSem, m\_RegionTMM.m\_vAreaKrds, \_S(""), m\_vGisData[i + 1]);

 size\_t nSize = 0;

 for (auto itr : m\_vGisData[i + 1])

 {

 if (itr.second.empty())

 nSize++;

 }

 if (nSize == m\_vGisData[i + 1].size())

 continue;

 nIdx = m\_grid.InsertRow();

 m\_grid.SetItemAsExpanded(nIdx, 0, nIdx, 1, vRowName[i]);

 m\_grid.SetRowBkColour(nIdx, SLBROWN);

 fillGrid(m\_vGisData[i + 1], m\_grid);

 }

 RestoreCursor();

 }

 //--------------------------------------------------------------------------

 void CDlgPropertiesRoadNet::Show(CGisMapWindow\* pWnd\_)

 {

 if (m\_vContours.IsEmpty())

 return;

 CGraphics &gr = pWnd\_->graph;

 CSPen penRgn(PS\_SOLID, 1, COLOR\_BROWN);

 CSBrush brushRgn(BS\_HATCHED, COLOR\_BROWN, HS\_SOLID);

 CGdiObj oldPen = gr.SelectObject(&penRgn);

 CGdiObj oldBrush = gr.SelectObject(&brushRgn);

 CSRgn rgnFull;

 const int nCountContours = m\_vContours.GetCount();

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

 {

 const TContours& vCounture = m\_vContours[i];

 const int nCountCounture = vCounture.GetCount();

 for (int j = 0; j < nCountCounture; ++j)

 {

 CSRgn rgn;

 const CVector<CPointD>& vKrd = vCounture[j];

 const int nCountKrds = vKrd.GetCount();

 bm::vector<CPointS> vPnt(nCountKrds);

 for (int k = 0; k < nCountKrds; ++k)

 ::KmPixelEx(pWnd\_, CKoord(vKrd[k].x, vKrd[k].y), vPnt[k]);

 rgn.CreatePolygonRgn(&vPnt.front(), nCountKrds, ALTERNATE);

 vPnt.push\_back(vPnt.front());

 pWnd\_->graph.Polyline(&vPnt.front(), nCountKrds + 1);

 if (i % 2)

 rgnFull = rgnFull - rgn;

 else

 rgnFull = rgnFull + rgn;

 }

 }

 gr.PaintRgn(&rgnFull);

 gr.SelectObject(&oldPen);

 gr.SelectObject(&oldBrush);

 }

 //--------------------------------------------------------------------------

 void CDlgPropertiesRoadNet::OnChangeRow()

 {

 selectMapObj(m\_grid);

 }

}

/// Класс характеристик гидротехнических сооружений

 class CDlgHydroTechnicalConstruction : public CSBaseDialog

 {

 DECLARE\_NO\_COPY\_CLASS(CDlgHydroTechnicalConstruction)

 public:

 CDlgHydroTechnicalConstruction(CSBaseWindow\* pParent\_, const SRegionTMM& RegionTMM\_);

 virtual ~CDlgHydroTechnicalConstruction() {};

 /// Отображает закрытость местности

 void Show(CGisMapWindow \*pWnd\_);

 protected:

 long OnCreate();

 long OnDestroy();

 void OnSize(UINT, int, int);

 void OnChangeRow();

 private:

 //\*\*\*

 SRegionTMM m\_RegionTMM;

 //\*\*\*

 CDrawOnGisWnd m\_DrawOnGisWnd;

 CVector<TContours> m\_vContours;

 CSGrid m\_grid;

 int m\_nFixedRow;

 TgisDataGeoPnts m\_gisDataGeoPnts;

 TObjBossSubj m\_gisData;

 void fillGridHead();

 void fillGridData();

 };

//=======================================================================================

 CDlgHydroTechnicalConstruction::CDlgHydroTechnicalConstruction(CSBaseWindow \*pParentWnd\_, const SRegionTMM& RegionTMM\_)

 : CSBaseDialog(DLG\_TERRAIN\_PARAMS, pParentWnd\_, g\_hInstance), m\_RegionTMM(RegionTMM\_)

 {

 m\_nFixedRow = 1;

 m\_DrawOnGisWnd.SetParams(this, GetIdActiveWnd(), (CDrawOnGisWnd::TDrawHandler)&CDlgHydroTechnicalConstruction::Show);

 IFace\_Attach(IFD\_GIS, this);

 EnableSerialize(true);

 }

 //--------------------------------------------------------------------------

 long CDlgHydroTechnicalConstruction::OnCreate()

 {

 SetText(\_S("Характеристики гидротехнических сооружений") + " [" + m\_RegionTMM.m\_sName + "]");

 m\_grid.SubclassDlgItem(IDC\_GRID, this);

 m\_grid.SetEditable(false);

 ON\_GRID\_SELCHANGE(IDC\_GRID, &CDlgHydroTechnicalConstruction::OnChangeRow);

 fillGridHead();

 fillGridData();

 std::set<int> typesSpecified;

 typesSpecified.insert(UT\_SEA);

 typesSpecified.insert(UT\_DAM);

 GetContoursBySurfaceStateTypes(m\_RegionTMM, typesSpecified, m\_vContours);

 RepaintServiceModellingEnvironment(m\_DrawOnGisWnd.GetGisWndId());

 ShakeWindow();

 return 1L;

 }

 //--------------------------------------------------------------------------

 long CDlgHydroTechnicalConstruction::OnDestroy()

 {

 m\_vContours.DelAllItems();

 RepaintServiceModellingEnvironment(m\_DrawOnGisWnd.GetGisWndId());

 CSP<ICtrlGis> CtrlGis;

 if (Get\_Interface(CtrlGis))

 if (CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd()))

 {

 SUUID wndId = pWnd->getUserWndId();

 CtrlGis->SelectObjectById(wndId, UNKNOWN\_EXTGOBJKEY, true);//Очистка отселектированных объектов в отселектированных типах

 CtrlGis->updateMapImage(wndId, gis::GetDefaultSettingsForRedrawDynamicLayers());

 }

 IFace\_Detach(IFD\_GIS, this);

 CSBaseDialog::OnDestroy();

 delete this;

 return 1;

 }

 //--------------------------------------------------------------------------

 void CDlgHydroTechnicalConstruction::OnSize(UINT, int, int)

 {

 if (IsInitialized())

 {

 RECT rc;

 GetClientRect(&rc);

 const int nBorder = 4;

 m\_grid.Move(nBorder, nBorder, rc.right - nBorder \* 2, rc.bottom - nBorder \* 2);

 m\_grid.AutoSize(true);

 }

 }

 //--------------------------------------------------------------------------

 void CDlgHydroTechnicalConstruction::fillGridHead()

 {

 m\_grid.SetColumnCount(2);

 m\_grid.SetFixedRowCount(m\_nFixedRow);

 m\_grid.SetItemAsText(0, 0, \_S("Основные характеристики"));

 m\_grid.SetItemAsText(0, 1, \_S("Значения"));

 }

 //--------------------------------------------------------------------------

 void CDlgHydroTechnicalConstruction::fillGridData()

 {

 SetWaitCursor();

 //Слой (ГИДРОГРАФИЯ) LAYER5

 bm::vector<CVector<STR>> vTypes(2);

 vTypes[0].AddItem(\_S("P0031131000")); // ВОДОХРАНИЛИЩА

 vTypes[0].AddItem(\_S("S0031131000")); // ВОДОХРАНИЛИЩЕ

 // Слой (ГИДРОТЕХНИЧЕСКИЕ СООРУЖЕНИЯ) LAYER7

 vTypes[1].AddItem(\_S("L0032110000")); // ПЛОТИНА

 vTypes[1].AddItem(\_S("V0032110000")); // ПЛОТИНЫ НЕПРОЕЗЖИЕ

 vTypes[1].AddItem(\_S("P0032110000")); // ПЛОТИНЫ

 bm::vector<CVector<TpairSS>> vSems(2);

 vSems[0].AddItem(make\_pair(\_S("SEM9"), \_S("Название")));

 vSems[0].AddItem(make\_pair(\_S("SEM77"), \_S("Объем, куб. км")));

 vSems[0].AddItem(make\_pair(\_S("SEM78"), \_S("Площадь зеркала, кв. км")));

 vSems[1].AddItem(make\_pair(\_S("SEM10"), \_S("Материал")));

 vSems[1].AddItem(make\_pair(\_S("SEM3"), \_S("Состояние")));

 vSems[1].AddItem(make\_pair(\_S("SEM2"), \_S("Длина, м")));

 vSems[1].AddItem(make\_pair(\_S("SEM11"), \_S("Ширина по верху, м")));

 vSems[1].AddItem(make\_pair(\_S("SEM75"), \_S("Материал водосливной части")));

 vSems[1].AddItem(make\_pair(\_S("SEM76"), \_S("Длина водосливной части")));

 vSems[1].AddItem(make\_pair(\_S("SEM25"), \_S("Отметка верхнего уровня")));

 vSems[1].AddItem(make\_pair(\_S("SEM26"), \_S("Отметка нижнего уровня")));

 bm::vector<STR> vRowNames;

 vRowNames.push\_back(\_S("Характеристики водохранилища"));

 vRowNames.push\_back(\_S("Плотины"));

 TColors vRowColors;

 vRowColors.push\_back(SBLUE);

 vRowColors.push\_back(SGRAYSILVER);

 TColors vTextColors;

 vTextColors.push\_back(SWHITE);

 fillDataWithGeoPnts(vTypes[0], vSems[0], m\_RegionTMM.m\_vAreaKrds, \_S(""), m\_gisDataGeoPnts);

 fillDataByGeoPnts(vTypes[1], vSems[1], m\_RegionTMM.m\_vAreaKrds, m\_gisDataGeoPnts, m\_gisData);

 fillGridWithGeo(m\_gisDataGeoPnts, m\_gisData, m\_grid, vRowNames, vRowColors, vTextColors);

 RestoreCursor();

 }

 //--------------------------------------------------------------------------

 void CDlgHydroTechnicalConstruction::Show(CGisMapWindow\* pWnd\_)

 {

 if (m\_vContours.IsEmpty())

 return;

 CGraphics &gr = pWnd\_->graph;

 CSPen penRgn(PS\_SOLID, 1, SBLUE);

 CSBrush brushRgn(BS\_HATCHED, SBLUE, HS\_SOLID);

 CGdiObj oldPen = gr.SelectObject(&penRgn);

 CGdiObj oldBrush = gr.SelectObject(&brushRgn);

 CSRgn rgnFull;

 const int nCountContours = m\_vContours.GetCount();

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

 {

 const TContours& vCounture = m\_vContours[i];

 const int nCountCounture = vCounture.GetCount();

 for (int j = 0; j < nCountCounture; ++j)

 {

 CSRgn rgn;

 const CVector<CPointD>& vKrd = vCounture[j];

 const int nCountKrds = vKrd.GetCount();

 bm::vector<CPointS> vPnt(nCountKrds);

 for (int k = 0; k < nCountKrds; ++k)

 ::KmPixelEx(pWnd\_, CKoord(vKrd[k].x, vKrd[k].y), vPnt[k]);

 rgn.CreatePolygonRgn(&vPnt.front(), nCountKrds, ALTERNATE);

 vPnt.push\_back(vPnt.front());

 pWnd\_->graph.Polyline(&vPnt.front(), nCountKrds + 1);

 if (i % 2)

 rgnFull = rgnFull - rgn;

 else

 rgnFull = rgnFull + rgn;

 }

 }

 gr.PaintRgn(&rgnFull);

 gr.SelectObject(&oldPen);

 gr.SelectObject(&oldBrush);

 }

 //--------------------------------------------------------------------------

 void CDlgHydroTechnicalConstruction::OnChangeRow()

 {

 selectMapObj(m\_grid);

 }

}

Реализация функций основного диалога:

 typedef pair<STR, STR> TpairSS;

 typedef map<STR, STR> TmapType2Data; // Имена семантик к их данным

 typedef pair<TExtGObjID, TmapType2Data> TObj2Data; // Ключ объекта карты к списку семантик

 typedef multimap<double, TObj2Data> TSort2Obj; // Сортированный набор объектов

 typedef map<STR, TSort2Obj> TgisData; // Cлой ГИС к данным

 typedef pair<TObj2Data, TGeoPoints> TObj2GeoPnts; // Набор объектов с их геодезическими координатами

 typedef multimap<double, TObj2GeoPnts> TSort2ObjGeoPnts; // Сортированный набор объектов

 typedef map<STR, TSort2ObjGeoPnts> TgisDataGeoPnts; // Слой ГИС с данными

 typedef map<TExtGObjID, set<TObj2Data>> TObjBossSubj; // Набор номеров объектов и с прилегающиит к ним объектами

 typedef bm::vector<COLORREF> TColors;

 /// @brief Получает интерфейс запроса

 CSP<gis::IQuery> GetQuery(const TVectorKrd &vAreaKrds\_);

 CSP<gis::IQuery> GetQuery(const TGeoPoints& vGeoPoints, const TVectorKrd& vAreaKrds\_);

 /// @brief Получает данные по заданным объектам в слоях ГИС

 /// @param [in] vTypes\_ - список объектов

 /// @param [in] mapSem\_ - семантика объектов

 /// @param [in] vKrds\_ - координаты района

 /// @param [in] semToSort\_ - семантика, по которой осуществляется сортировка объектов

 /// @param [out] Data\_ - данные по заданным объектам в слоях ГИС

 void fillData(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vKrds\_,

 const STR& semToSort\_, TgisData& Data\_);

 /// @brief Заполняет данные по объектам ГИС в таблице

 /// @param [in] Data\_ - данные по заданным объектам в слоях ГИС

 /// @param [in] grid\_ - таблица с результами

 void fillGrid(const TgisData& Data\_, CSGrid& grid\_);

 /// @brief Селектирует заданные объекты ГИС на карте

 /// @param [in] grid\_ - таблица с результами

 void selectMapObj(const CSGrid& grid\_);

 /// @brief Получает данные по заданным объектам в слоях ГИС

 /// @param [in] vTypes\_ - список объектов

 /// @param [in] vSem\_ - семантика объектов

 /// @param [in] vKrds\_ - координаты района

 /// @param [in] semToSort\_ - семантика, по которой осуществляется сортировка объектов

 /// @param [out] Data\_ - данные по заданным объектам в слоях ГИС, в том числе геодезические координаты местоположения объекта

 void fillDataWithGeoPnts(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vKrds\_,

 const STR& semToSort\_, TgisDataGeoPnts& Data\_);

 /// @brief Получает данные по заданным объектам в слоях ГИС

 /// @param [in] vTypes\_ - список объектов

 /// @param [in] vSem\_ - семантика объектов

 /// @param [in] vKrds\_ - координаты района

 /// @param [out] geoData\_ - данные по заданным объектам в слоях ГИС, в том числе геодезические координаты местоположения объекта

 /// @param [out] Data\_ - данные по заданным объектам

 void fillDataByGeoPnts(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vAreaKrds\_,

 TgisDataGeoPnts& geoData\_, TObjBossSubj& Data\_);

 /// @brief Заполняет данные по объектам ГИС в таблице

 /// @param [in] Data\_ - данные по заданным объектам в слоях ГИС, в том числе геодезические координаты местоположения объекта

 /// @param [in] DataB\_ - данные по прилегающим объектам

 /// @param [in] grid\_ - таблица с результами

 /// @param [in] vRowNames\_ - массив с заголовками для объектов

 /// @param [in] vRowColors\_ - массив с цветами выделения заголовков

 /// @param [in] vTextColors\_ - массив с цветами для текста заголовка

 void fillGridWithGeo(const TgisDataGeoPnts& Data\_, TObjBossSubj& DataB\_, CSGrid& grid\_,

 bm::vector<STR> vRowNames\_, TColors vRowColors\_ = TColors(), TColors vTextColors\_ = TColors());

 //----------------------------------------------------------------------------

 CSP<gis::IQuery> GetQuery(const TVectorKrd &vAreaKrds\_)

 {

 CGisPolygon pSelRgn;

 for (int i = 0; i < vAreaKrds\_.GetCount(); ++i)

 pSelRgn.addPoint(SGeoPnt(vAreaKrds\_[i].GetB(), vAreaKrds\_[i].GetL()));

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd)

 return NULL;

 CSP<gis::IQuery> query = dyn\_cast<gis::IQuery>(CFactoryClass::SafeCreate(GET\_OBJETS\_UNDER\_GEOM\_QUERY));

 gis::CGetObjectsUnderGeomQuery\* pQueryG = dynamic\_cast<gis::CGetObjectsUnderGeomQuery\*>(GetImpl(query));

 if (pQueryG)

 {

 pSelRgn.setProjection(pWnd->getProjection());

 pSelRgn.buildBoundRect();

 CVector< CVector<SGeoPnt> > points;

 pSelRgn.getMetrics(points);

 if (!points.IsEmpty())

 {

 for (int j = 0; j < points[0].GetCount(); ++j)

 pQueryG->pnts.push\_back(points[0][j]);

 pQueryG->eGeom = (gis::CGetObjectsUnderGeomQuery::EGeometryQuery)3;

 }

 }

 return query;

 }

 //----------------------------------------------------------------------------

 CSP<gis::IQuery> GetQuery(const TGeoPoints& vGeoPoints\_, const TVectorKrd &vAreaKrds\_)

 {

 CGisPolygon pReg1;

 for (int i = 0; i < vAreaKrds\_.GetCount(); ++i)

 pReg1.addPoint(SGeoPnt(vAreaKrds\_[i].GetB(), vAreaKrds\_[i].GetL()));

 pReg1.buildBoundRect();

 CGisPolygon pReg2;

 for (size\_t i = 0; i < vGeoPoints\_.size(); i++)

 pReg2.addPoint(vGeoPoints\_[i]);

 pReg2.buildBoundRect();

 CGisPolygon pSelRgn;

 for (int i = 0; i < vAreaKrds\_.GetCount(); ++i)

 {

 SGeoPnt geoPnt = SGeoPnt(vAreaKrds\_[i].GetB(), vAreaKrds\_[i].GetL());

 if (pReg2.IsHit(geoPnt))

 pSelRgn.addPoint(geoPnt);

 }

 for (size\_t i = 0; i < vGeoPoints\_.size(); i++)

 if (pReg1.IsHit(vGeoPoints\_[i]))

 pSelRgn.addPoint(vGeoPoints\_[i]);

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd)

 return NULL;

 CSP<gis::IQuery> query = dyn\_cast<gis::IQuery>(CFactoryClass::SafeCreate(GET\_OBJETS\_UNDER\_GEOM\_QUERY));

 gis::CGetObjectsUnderGeomQuery\* pQueryG = dynamic\_cast<gis::CGetObjectsUnderGeomQuery\*>(GetImpl(query));

 if (pQueryG)

 {

 pSelRgn.setProjection(pWnd->getProjection());

 pSelRgn.buildBoundRect();

 CVector< CVector<SGeoPnt> > points;

 pSelRgn.getMetrics(points);

 if (!points.IsEmpty())

 {

 for (int j = 0; j < points[0].GetCount(); ++j)

 pQueryG->pnts.push\_back(points[0][j]);

 pQueryG->eGeom = (gis::CGetObjectsUnderGeomQuery::EGeometryQuery)3;

 }

 }

 return query;

 }

 //----------------------------------------------------------------------------

 static int addData2Map(const IGisSemList\* Semantics\_, const STR& sSem\_, const STR& sName\_,

 TmapType2Data& mapData\_, const TMapSemInfo \*pSemInfo\_, const STR& SortSem\_)

 {

 int nRet = 0;

 if (const CGisSemValue\* pType = Semantics\_->GetSemantic(sSem\_))

 {

 if (SortSem\_ == sSem\_)

 nRet = (int)pType->ToInt32();

 STR sData = \_S("-");

 if (pSemInfo\_)

 {

 TMapSemInfo::const\_iterator it\_sem = pSemInfo\_->find(sSem\_);

 if (it\_sem != pSemInfo\_->end())

 {

 if (!it\_sem->second->haract.IsEmpty())

 {

 for (int i = 0; i < it\_sem->second->haract.GetCount(); ++i)

 if (const SSemanticHar\* pHar = it\_sem->second->haract.GetItemByIdx(i))

 if (pHar->kod\_znach == pType->ToInt32())

 sData = pHar->sod\_znach;

 }

 else

 sData = pType->ToStr();

 }

 if (sSem\_ == \_S("SEM15") && mapData\_.empty())

 return nRet;

 else

 mapData\_.insert(std::make\_pair(sName\_, sData));

 }

 }

 return nRet;

 }

 //----------------------------------------------------------------------------

 void fillData(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vKrds\_, const STR& semToSort\_, TgisData& Data\_)

 {

 CSP<ICtrlGis> CtrlGis;

 Get\_Interface(CtrlGis);

 if (!CtrlGis)

 return;

 if (CSP<gis::IQuery> spQuery = GetQuery(vKrds\_))

 {

 gis::CGetObjectsQuery\* pQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spQuery));

 if (!pQuery)

 return;

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd)

 return;

 SUUID idWnd = pWnd->getUserWndId();

 pQuery->srcId = idWnd;

 CMapWndHandle wndHandle;

 pWnd->getMapWndHandle(wndHandle);

 for (size\_t i = 0; i < wndHandle.getLayersCount(); ++i) //layer

 {

 const CLayerHandle& layerHandle = wndHandle.getLayer(i);

 for (size\_t j = 0; j < layerHandle.getDataTypesCount(); ++j) //type

 {

 const CDataTypeAttributes& typeAttributes = layerHandle.getDataType(j);

 SDTID idType = typeAttributes.getTypeId();

 if (IsStdServiceDataType(idType))

 continue;

 CGisStdTemplate Classifier;

 CtrlGis->GetClassifierById(typeAttributes.getTemplateId(), Classifier);

 TMapSemInfo \*pSemInfo = Classifier.getSemanticTpl();

 if (!pSemInfo)

 return;

 pQuery->m\_dataTypes.clear();

 pQuery->m\_dataTypes.insert(idType); /// перечень идентификаторов слоёв данных, где осуществляется поиск

 CtrlGis->getObjectsByQuery(spQuery);

 TSort2Obj mapObj;

 set<TGOBJ> setObj;

 for (size\_t k = 0; pQuery && k < pQuery->mapObjs.size(); ++k)

 {

 CMapObjectHandleExt \*pHandle = &pQuery->mapObjs[k];

 for (int l = 0; l < vTypes\_.GetCount(); ++l)

 {

 if (pHandle->getKeyKod() != vTypes\_[l])

 continue;

 TGOBJ idOwn = pHandle->getOwnerId();

 CSP<gis::IQuery> spOwner = new gis::CGetObjectsByID(idOwn, idType, idWnd);

 if (idOwn)

 {

 if (CtrlGis->getObjectsByQuery(spOwner))

 if (gis::CGetObjectsQuery\* pOwnQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spOwner)))

 if (!pOwnQuery->mapObjs.empty())

 pHandle = &pOwnQuery->mapObjs[0];

 }

 if (!pHandle)

 continue;

 if (IGisSemList\* pSemantics = pHandle->GetSemantics())

 {

 TmapType2Data Data;

 double dSort = 0;

 for (int m = 0; m < vSem\_.GetCount(); ++m)

 {

 double dTmp = (double)addData2Map(pSemantics, vSem\_[m].first, vSem\_[m].second, Data, pSemInfo, semToSort\_);

 if (dTmp > 0) dSort = dTmp;

 }

 if (Data.empty())

 continue;

 TGOBJ idThis = pHandle->getObjectId();

 auto it = setObj.insert(idThis);

 if (it.second)

 {

 TObj2Data pairSort(make\_pair(typeAttributes.getTypeId(), idThis), Data);

 mapObj.insert(make\_pair(dSort > 0 ? 1 / dSort : 0, pairSort));

 }

 }

 }

 }

 STR sLayer = STR().Format("Cлой ГИС - \"%s\"", typeAttributes.getName().c\_str());

 Data\_.insert(make\_pair(sLayer, mapObj));

 }

 }

 }

 }

 //----------------------------------------------------------------------------

 void fillGrid(const TgisData& Data\_, CSGrid& grid\_)

 {

 for (auto &it\_layer : Data\_)

 {

 int nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, it\_layer.first);

 grid\_.SetRowBkColour(nRow, SLBLUE);

 int nPart = 1;

 for (auto &it\_part : it\_layer.second)

 {

 nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, STR().Format("Участок %d", nPart));

 grid\_.SetRowBkColour(nRow, SLGREEN);

 ++nPart;

 for (auto &it\_sem : it\_part.second.second)

 {

 grid\_.SetItemData(nRow, 0, (STival)&it\_part.second.first);

 nRow = grid\_.InsertRow(it\_sem.first);

 grid\_.SetItemAsText(nRow, 1, it\_sem.second);

 grid\_.SetItemData(nRow, 0, (STival)&it\_part.second.first);

 }

 }

 }

 }

 //----------------------------------------------------------------------------

 void selectMapObj(const CSGrid& grid\_)

 {

 CSP<ICtrlGis> CtrlGis;

 Get\_Interface(CtrlGis);

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd || !CtrlGis)

 return;

 CSGridCellID cell = grid\_.GetCurrentCell();

 if (TExtGObjID \*pId = (TExtGObjID\*)grid\_.GetItemData(cell.row, 0))

 {

 SUUID wndId = pWnd->getUserWndId();

 CtrlGis->SelectObjectById(wndId, \*pId, true);

 CtrlGis->updateMapImage(wndId, gis::GetDefaultSettingsForRedrawDynamicLayers());

 }

 }

 //----------------------------------------------------------------------------

 void fillDataWithGeoPnts(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vKrds\_,

 const STR& semToSort\_, TgisDataGeoPnts& Data\_)

 {

 CSP<ICtrlGis> CtrlGis;

 Get\_Interface(CtrlGis);

 if (!CtrlGis)

 return;

 if (CSP<gis::IQuery> spQuery = GetQuery(vKrds\_))

 {

 gis::CGetObjectsQuery\* pQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spQuery));

 if (!pQuery)

 return;

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd)

 return;

 SUUID idWnd = pWnd->getUserWndId();

 pQuery->srcId = idWnd;

 CMapWndHandle wndHandle;

 pWnd->getMapWndHandle(wndHandle);

 for (size\_t i = 0; i < wndHandle.getLayersCount(); ++i) //layer

 {

 const CLayerHandle& layerHandle = wndHandle.getLayer(i);

 for (size\_t j = 0; j < layerHandle.getDataTypesCount(); ++j) //type

 {

 const CDataTypeAttributes& typeAttributes = layerHandle.getDataType(j);

 SDTID idType = typeAttributes.getTypeId();

 if (IsStdServiceDataType(idType))

 continue;

 CGisStdTemplate Classifier;

 CtrlGis->GetClassifierById(typeAttributes.getTemplateId(), Classifier);

 TMapSemInfo \*pSemInfo = Classifier.getSemanticTpl();

 if (!pSemInfo)

 return;

 pQuery->m\_dataTypes.clear();

 pQuery->m\_dataTypes.insert(idType); /// перечень идентификаторов слоёв данных, где осуществляется поиск

 CtrlGis->getObjectsByQuery(spQuery);

 TSort2ObjGeoPnts mapObj;

 set<TGOBJ> setObj;

 for (size\_t k = 0; pQuery && k < pQuery->mapObjs.size(); ++k)

 {

 CMapObjectHandleExt \*pHandle = &pQuery->mapObjs[k];

 for (int l = 0; l < vTypes\_.GetCount(); ++l)

 {

 if (pHandle->getKeyKod() != vTypes\_[l])

 continue;

 TGOBJ idOwn = pHandle->getOwnerId();

 CSP<gis::IQuery> spOwner = new gis::CGetObjectsByID(idOwn, idType, idWnd);

 if (idOwn)

 {

 if (CtrlGis->getObjectsByQuery(spOwner))

 if (gis::CGetObjectsQuery\* pOwnQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spOwner)))

 if (!pOwnQuery->mapObjs.empty())

 pHandle = &pOwnQuery->mapObjs[0];

 }

 if (!pHandle)

 continue;

 if (IGisSemList\* pSemantics = pHandle->GetSemantics())

 {

 TmapType2Data Data;

 double dSort = 0;

 for (int m = 0; m < vSem\_.GetCount(); ++m)

 {

 double dTmp = (double)addData2Map(pSemantics, vSem\_[m].first, vSem\_[m].second, Data, pSemInfo, semToSort\_);

 if (dTmp > 0) dSort = dTmp;

 }

 if (Data.empty())

 continue;

 TGOBJ idThis = pHandle->getObjectId();

 TGeoPoints vTGeoPnts;

 CVector<CVector<SGeoPnt>> vGeoPnts;

 pHandle->getMetrics(vGeoPnts);

 for (auto &itr : vGeoPnts)

 {

 for (auto &itr\_v : itr)

 vTGeoPnts.push\_back(itr\_v);

 }

 auto it = setObj.insert(idThis);

 if (it.second)

 {

 TObj2Data pairObj(make\_pair(typeAttributes.getTypeId(), idThis), Data);

 TObj2GeoPnts pairSort(make\_pair(pairObj, vTGeoPnts));

 mapObj.insert(make\_pair(dSort > 0 ? 1 / dSort : 0, pairSort));

 }

 }

 }

 }

 STR sLayer = STR().Format("Cлой ГИС - \"%s\"", typeAttributes.getName().c\_str());

 Data\_.insert(make\_pair(sLayer, mapObj));

 }

 }

 }

 }

 //----------------------------------------------------------------------------

 void fillDataByGeoPnts(const CVector<STR> &vTypes\_, const CVector<TpairSS>& vSem\_, const TVectorKrd &vAreaKrds\_,

 TgisDataGeoPnts& geoData\_, TObjBossSubj& Data\_)

 {

 CSP<ICtrlGis> CtrlGis;

 Get\_Interface(CtrlGis);

 if (!CtrlGis)

 return;

 for (auto &itr : geoData\_)

 {

 for (auto &itr\_m : itr.second)

 {

 if (CSP<gis::IQuery> spQuery = GetQuery(itr\_m.second.second, vAreaKrds\_))

 {

 gis::CGetObjectsQuery\* pQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spQuery));

 if (!pQuery)

 return;

 CGisMapWindow\* pWnd = dynamic\_cast<CGisMapWindow\*>(GetMapActiveWnd());

 if (!pWnd)

 return;

 SUUID idWnd = pWnd->getUserWndId();

 pQuery->srcId = idWnd;

 CMapWndHandle wndHandle;

 pWnd->getMapWndHandle(wndHandle);

 for (size\_t i = 0; i < wndHandle.getLayersCount(); ++i) //layer

 {

 const CLayerHandle& layerHandle = wndHandle.getLayer(i);

 for (size\_t j = 0; j < layerHandle.getDataTypesCount(); ++j) //type

 {

 const CDataTypeAttributes& typeAttributes = layerHandle.getDataType(j);

 SDTID idType = typeAttributes.getTypeId();

 if (IsStdServiceDataType(idType))

 continue;

 CGisStdTemplate Classifier;

 CtrlGis->GetClassifierById(typeAttributes.getTemplateId(), Classifier);

 TMapSemInfo \*pSemInfo = Classifier.getSemanticTpl();

 if (!pSemInfo)

 return;

 pQuery->m\_dataTypes.clear();

 pQuery->m\_dataTypes.insert(idType); /// перечень идентификаторов слоёв данных, где осуществляется поиск

 CtrlGis->getObjectsByQuery(spQuery);

 set<TGOBJ> setObj;

 for (size\_t k = 0; pQuery && k < pQuery->mapObjs.size(); ++k)

 {

 CMapObjectHandleExt \*pHandle = &pQuery->mapObjs[k];

 for (int l = 0; l < vTypes\_.GetCount(); ++l)

 {

 if (pHandle->getKeyKod() != vTypes\_[l])

 continue;

 TGOBJ idOwn = pHandle->getOwnerId();

 CSP<gis::IQuery> spOwner = new gis::CGetObjectsByID(idOwn, idType, idWnd);

 if (idOwn)

 {

 if (CtrlGis->getObjectsByQuery(spOwner))

 if (gis::CGetObjectsQuery\* pOwnQuery = dynamic\_cast<gis::CGetObjectsQuery\*>(GetImpl(spOwner)))

 if (!pOwnQuery->mapObjs.empty())

 pHandle = &pOwnQuery->mapObjs[0];

 }

 if (!pHandle)

 continue;

 TObj2Data mapObj;

 if (IGisSemList\* pSemantics = pHandle->GetSemantics())

 {

 TmapType2Data Data;

 for (int m = 0; m < vSem\_.GetCount(); ++m)

 addData2Map(pSemantics, vSem\_[m].first, vSem\_[m].second, Data, pSemInfo, \_S(""));

 if (Data.empty())

 continue;

 TGOBJ idThis = pHandle->getObjectId();

 auto it = setObj.insert(idThis);

 if (it.second)

 {

 TObj2Data pairObj(make\_pair(typeAttributes.getTypeId(), idThis), Data);

 mapObj = pairObj;

 }

 }

 Data\_[itr\_m.second.first.first].insert(mapObj);

 }

 }

 }

 }

 }

 }

 }

 }

 //----------------------------------------------------------------------------

 void fillGridWithGeo(const TgisDataGeoPnts& Data\_, TObjBossSubj& DataB\_, CSGrid& grid\_,

 bm::vector<STR> vRowNames\_, TColors vRowColors\_, TColors vTextColors\_)

 {

 for (auto &it\_layer : Data\_) // проход по слоям ГИС

 {

 int nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, it\_layer.first);

 grid\_.SetRowBkColour(nRow, SLBLUE);

 int nPart = 1;

 for (auto &it\_part : it\_layer.second) // проход по главным объектам

 {

 // заголовок для главного объекта

 int i = 0;

 nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, vRowNames\_[i]);

 if ((int)vRowColors\_.size() > i)

 grid\_.SetRowBkColour(nRow, vRowColors\_[i]);

 if ((int)vTextColors\_.size() > i)

 grid\_.SetItemTextColour(nRow, 0, vTextColors\_[i]);

 i++;

 nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, STR().Format("Участок %d", nPart));

 grid\_.SetRowBkColour(nRow, SLGREEN);

 ++nPart;

 // вывод характеристик главного объекта

 for (auto &it\_sem : it\_part.second.first.second)

 {

 grid\_.SetItemData(nRow, 0, (STival)&it\_part.second.first.first);

 nRow = grid\_.InsertRow(it\_sem.first);

 grid\_.SetItemAsText(nRow, 1, it\_sem.second);

 grid\_.SetItemData(nRow, 0, (STival)&it\_part.second.first.first);

 }

 // ищем по номеру главного объекта, есть ли у него прилегающие объекты

 TObjBossSubj::iterator it\_mObj = DataB\_.find(it\_part.second.first.first);

 if (it\_mObj != DataB\_.end())

 {

 // заголовок прилегающих объектов

 nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, vRowNames\_[i]);

 if ((int)vRowColors\_.size() > i)

 grid\_.SetRowBkColour(nRow, vRowColors\_[i]);

 if ((int)vTextColors\_.size() > i)

 grid\_.SetItemTextColour(nRow, 0, vTextColors\_[i]);

 for (auto &it\_sObj : it\_mObj->second)

 {

 nRow = grid\_.InsertRow();

 grid\_.SetItemAsExpanded(nRow, 0, nRow, 1, STR().Format("Участок %d", nPart));

 grid\_.SetRowBkColour(nRow, SLGREEN);

 ++nPart;

 for (auto &it\_value : it\_sObj.second)

 {

 // вывод прилегающих объектов

 grid\_.SetItemData(nRow, 0, (STival)&it\_sObj.first);

 nRow = grid\_.InsertRow(it\_value.first);

 grid\_.SetItemAsText(nRow, 1, it\_value.second);

 grid\_.SetItemData(nRow, 0, (STival)&it\_sObj.first);

 }

 }

 }

 }

 }

 }

}

# Список литературы

1. Поляков И.С. Методика оперативной оценки условий передвижения войск по местности и определение потребных сил для подготовки путей с использованием картограмм. – М., ВИА.1998. – 69 с.
2. Поляков И.С. Картограммы для оперативной оценки условий передвижения войск по местности и определения потребных сил для подготовки путей. Западный регион РФ ВИА. – М., 1995.
3. Исследование операций: Учебник / Под ред. Б.Н. Юркова. – М,: ВИКА, 1990. – 528 с.
4. Кузьминский Р.А., Каканов В.Г. Основы устройства водных заграждений. Лекция. -М.:ВИА,1993.
5. Кузьминский Р.А., Ткачев А.П. Разрушение сооружений гидроузла для устройства водного заграждения. Учебное пособие.-М.:ВИУ,2000.
6. Кузьминский Р.А. Прогнозирование разрушения сооружений гидроузла. Учебное пособие. –М.:ВИА,1984.