Нижегородский государственный университет
им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Лабораторная работа 2

Применение модели грид-потоков для внедрения в грид существующего приложения

Цель лабораторной работы

- Целью лабораторной работы является изучение процесса внедрения *существующего* приложения в грид, построенную на базе инструментария Alchemi
- В качестве иллюстрирующего приложения выбирается известная программа трассировки лучей POV-Ray с предустановленной неофициальной сборкой MegaPOV. Подробнее данные программы обсуждаются ниже

Упражнения

- *Упражнение 1*

  Знакомство с программой трассировки лучей MegaPOV

- *Упражнение 2*

  Декомпозиция входного набора данных на независимые части

- *Упражнение 3*

  Разработка распределенного приложения для инструментария Alchemi

Примерное время выполнения лабораторной работы: ***240 минут***

Некоторые замечания…

- *Что означает внедрить в грид существующее приложение?*

  Под внедрением в грид некоторого приложения здесь подразумевается организация распределенной обработки входных данных на вычислительных ресурсах грид с помощью *данного приложения*. После завершения обработки данных пользователь должен получить итоговый результат как единое целое. Таким образом, для пользователя распределенные вычислительные ресурсы выступают в роли единого *виртуального вычислительного ресурса*

- *Для чего это нужно*

  Внедрять в грид существующие приложения, как правило, значительно проще чем разрабатывать новые. Одной из серьезных проблем существующих грид является недостаточная поддержка со стороны прикладных программ. Обеспечение пользователей и разработчиков простым механизмом интеграции приложений в грид – это составляющая успеха грид-технологии

Некоторые замечания…

- *Что такое трассировка лучей?*

  Это метод машинной графики, позволяющий создавать фотореалистические изображения любых трехмерных сцен. Трассировка лучей моделирует прохождение лучей света через изображаемую сцену. Фотореализм достигается путем математического моделирования оптических свойств света и его взаимодействия с объектами

- *Применение метода трассировки лучей*

  Трассировка лучей используется в компьютерной графике давно и успешно. Программы, выполняющие трассировку, нередко являются составной частью сложных систем моделирования, таких как, например, 3D Studio Max. Однако надо заметить, что, обеспечивая очень высокое качество изображений, трассировка лучей как метод визуализации требует довольно много времени и применяется на финальных стадиях подготовки изображений. *Зачастую одно изображение может трассироваться на протяжении нескольких часов и даже суток*, но результаты оправдывают временные затраты

Некоторые замечания

- Примеры изображений, синтезированных методом трассировки лучей (источник – http://www.povray.org):

↳ *Работу программ трассировки лучей можно значительно ускорить, если обрабатывать компьютерные сцены в грид*

Упражнение 1 – Знакомство с программой…

- *Пакет POV-Ray*

  Среди широкого семейства программ, в которых, так или иначе, используется трассировка лучей, особое место занимает POV-Ray. Данный пакет за долгий срок своего существования заслужил общее признание и уважение и используется в самых различных научных исследованиях. Однако сегодня пакет POV-Ray содержит лишь *графическую* версию программы трассировки лучей. Данная версия программы удобна в работе, однако большинство операций доступны в виде взаимодействий с графическим интерфейсом

- *Пакет MegaPOV*

  Для нашей задачи необходима консольная программа, которая позволяет специфицировать входные параметры и файл, в который будет сохранен результат. Неофициальная сборка MegaPOV содержит консольную версию программы трассировки, полностью совместимую с файлами сцен POV-Ray

↳ В данной лабораторной работе будет использоваться пакет MegaPOV

Упражнение 1 – Знакомство с программой…

- Рассмотрим работу программы трассировки MegaPOV на примере произвольной сцены:
  - установите приложение, распаковав дистрибутив, загруженный с сайта проекта (и прилагаемый к данной лабораторной работе), в произвольный каталог на жестком диске;
  - пропишите путь к подкаталогу **bin** установочного каталога в переменную окружения **Path**;
  - откройте командную строку с помощью команды **Start | Programs | Standard | Command Prompt**. Далее перейдите в подкаталог **bin** установочного каталога и выполните следующие действия:

```
megapov +I..\scenes\megapov\bear.pov +L..\include +W320 +H240 +Obear.png +FN16
```

  - в результате выполнения данной команды в подкаталоге **bin** появится сгенерированное изображение

Упражнение 1 – Знакомство с программой…

- Общий формат команды запуска программы MegaPOV:

```
megapov [+/-]Опция [+/-]Опция [+/-]Опция ...
```

Для активации некоторой новой опции необходимо воспользоваться символом “+”, для отключения опции – символом “–”

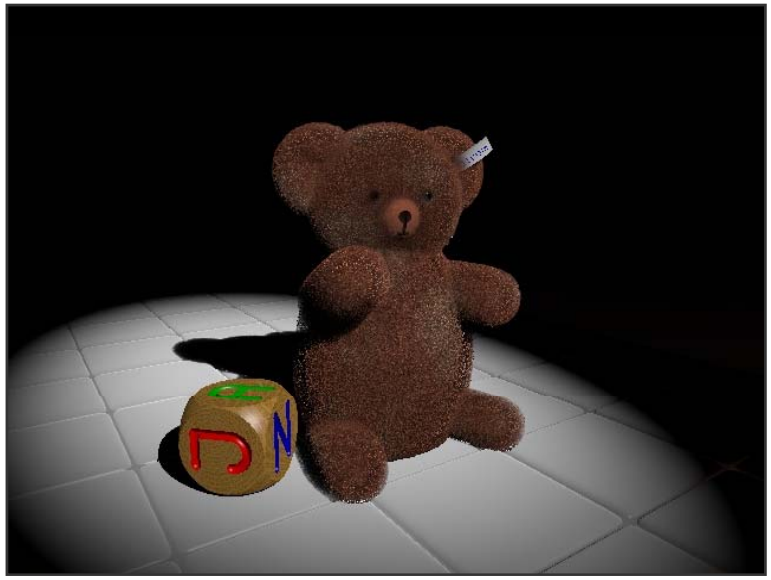
- Перечислим основные параметры программы MegaPOV:
  - **I<имя файла>** – путь к файлу сцены для трассировки;
  - **L<имя папки>** – путь к каталогу вспомогательных файлов;
  - **Hn** – высота изображения *n* пикселей;
  - **Wn** – ширина изображения *n* пикселей;
  - **SRn** – начальная строка *n* обрабатываемой части изображения;
  - **ERn** – конечная строка *n* обрабатываемой части изображения;
  - **SCn** – начальный столбец *n* обрабатываемой части изображения;
  - **ECn** – конечный столбец *n* обрабатываемой части изображения;

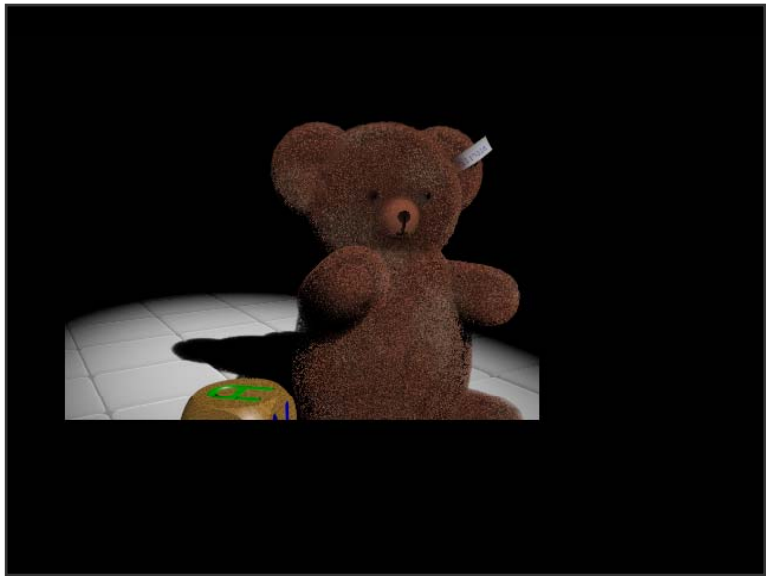
Упражнение 1 – Знакомство с программой

- Для изучения работы расширенных параметров программы MegaPOV выполните следующую команду:

```
megapov +I..\scenes\megapov\bear.pov +L..\include +W640 +H480 +SR1 +ER200 +SC50 +EC250 +A
+R4  +Obear.png +FN16
```

Данная команда производит рендеринг прямоугольной части изображения с активированными настройками сглаживания





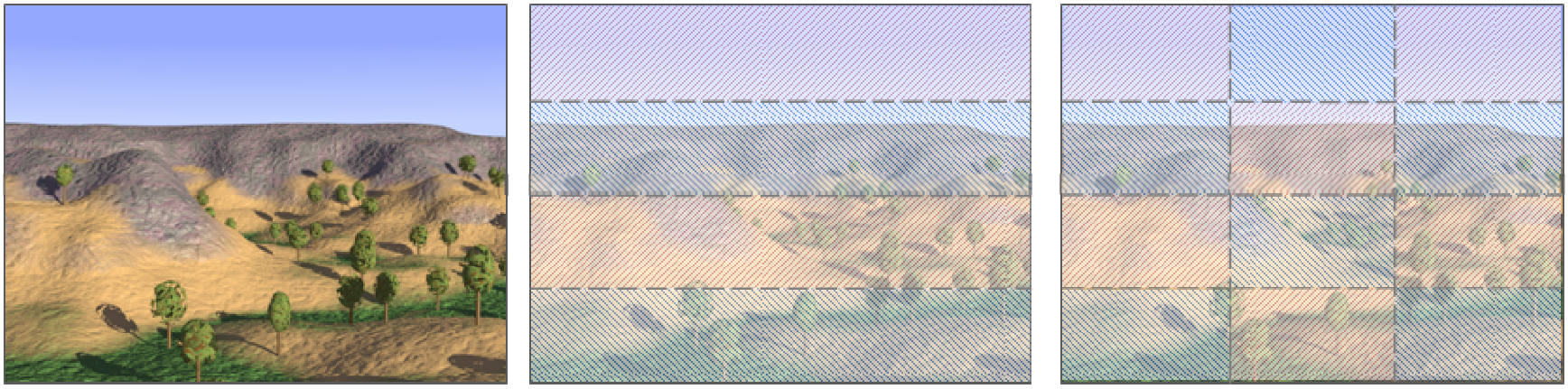
Упражнение 2 – Декомпозиция вычислений…

- Процесс внедрения существующего приложения в грид с помощью модели грид-потоков можно представить в виде последовательности трех основных шагов:
  - анализ входного набора данных и его декомпозиция на *независимые части*;
  - реализация класса *грид-потока* для запуска существующего приложения на удаленном Исполнителе с определенной порцией данных;
  - организация параллельной работы грид-потоков в вычислительной сети с помощью класса *грид-приложения*

Рассмотрим первый этап – декомпозицию входного набора данных

Упражнение 2 – Декомпозиция вычислений…

- При внедрении *существующего* приложения в грид распределенные вычисления сводятся к выполнению однотипной обработки элементов большого набора входных данных на различных Исполнителях
- Имеет место *параллелизм по данным*, и выделение подзадач для грид-потоков сводится к разделению имеющихся входных данных



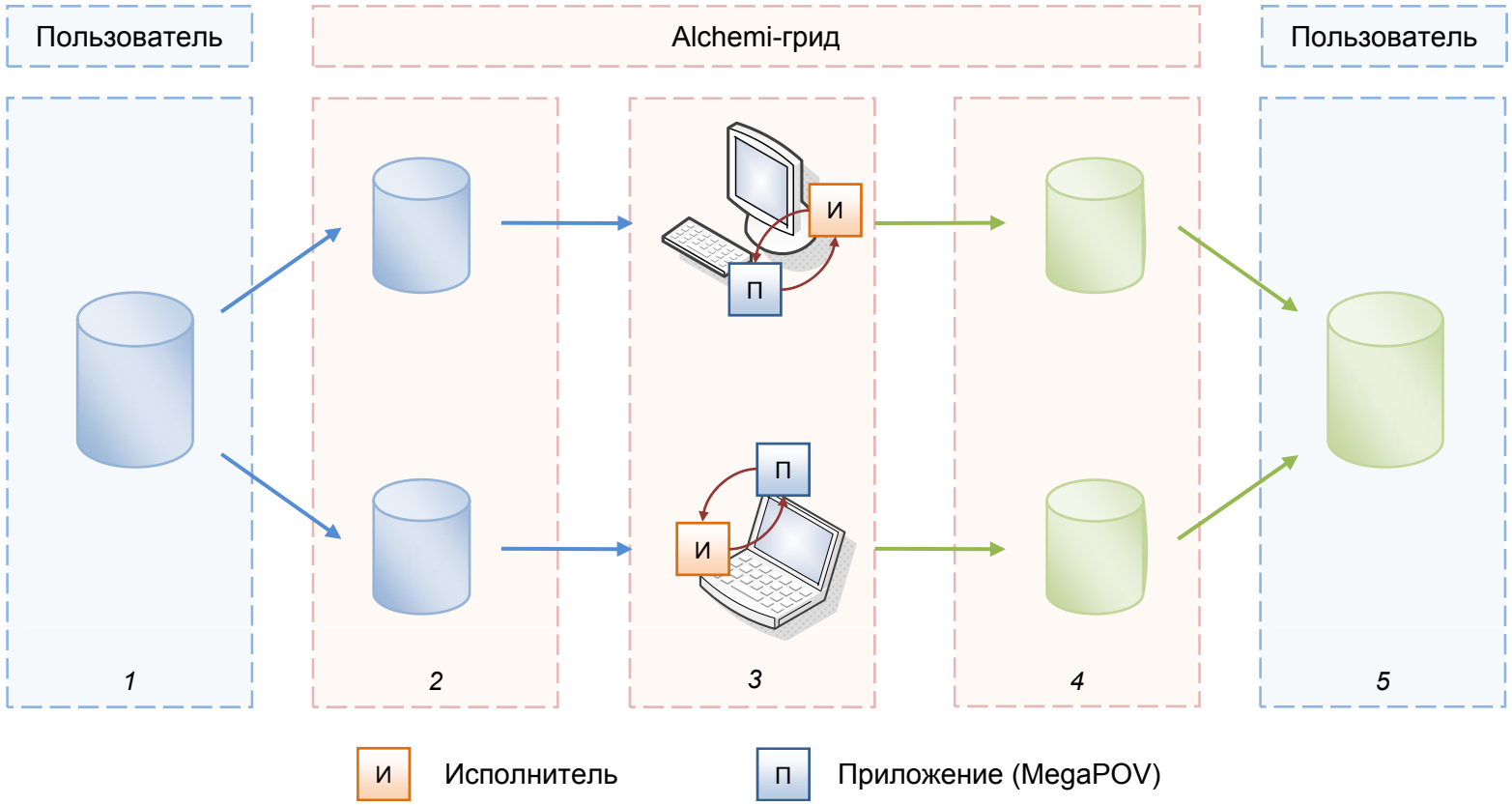
- Исходное множество пикселей изображения может быть разделено на отдельные группы строк – *ленточная схема* разделения данных или на прямоугольные наборы пикселей – *блочная схема*

Упражнение 2 – Декомпозиция вычислений…

- Описанная схема не является универсальной – она не применима к *любому* приложению. Рассмотрим некоторые трудности, которые *могут* возникнуть в процессе ее реализации:
  - набор входных данных должен допускать декомпозицию на *независимые* части (в противном случае их некорректно разделять);
  - должна существовать возможность запуска приложения для обработки определенной *порции входных данных*;
  - трудности при конструировании *итогового* результата из *частичных* результатов, вычисленных отдельными Исполнителями
- Четко формализовать все условия, которым должна удовлетворять программа для внедрения в грид (имеется в виду распределенная обработка данных, а не запуск приложения на удаленном ресурсе), довольно трудно. Однако вряд ли имеет смысл подробно останавливаться на этом вопросе, поскольку для каждого конкретного случая возможность внедрения приложения в грид вполне очевидна

Упражнение 2 – Декомпозиция вычислений

- Рассмотрим процесс распределенной обработки данных в Alchemi-грид:



Упражнение 3 – Разработка грид-приложения

- *Задание 1*

  Открытие проекта **DistributedMegaPOV**

- *Задание 2*

  Разработка класса грид-потока

- *Задание 3*

  Запуск грид-потоков на локальной машине

- *Задание 4*

  Тестирование приложения на локальной машине

- *Задание 5*

  Запуск грид-потоков в вычислительной грид

- *Задание 6*

  Тестирование приложения в вычислительной грид

Универсальная схема

Задание 1 – Открытие проекта

- Для открытия проекта **DistributedMegaPOV** выполните следующие шаги:
  - запустите среду Microsoft Visual Studio 2005, если она еще не запущена,
  - в меню **File** выполните команду **Open | Project/Solution**,
  - в диалоговом окне **Open Project** выберите папку **DistributedMegaPOV**,
  - выберите файл **DistributedMegaPOV.sln** и выполните команду **Open**
- После выполнения описанных шагов в окне **Solution Explorer** будет отображена структура проекта **DistributedMegaPOV**:
  - динамическая библиотека **RenderThread** – содержит реализацию класса грид-потока для запуска приложения MegaPOV;
  - приложение **DistributedMegaPOV** – основное приложение, в котором формируются грид-потоки и отправляются для исполнения в вычислительную грид, построенную на базе инструментария Alchemi

Задание 2 – Разработка класса грид-потока…

- Чтобы открыть заготовку кода для класса грид-потока разверните проект **RenderThread** и дважды щелкните мышкой на файле **RenderThread.cs** *Открыть*
- Замечания об объявлении класса грид-потока:
  - перед объявлением класса необходимо подключить библиотеку **Alchemi** (подключается в *любом* проекте) и дополнительные системные библиотеки;
  - класс грид-потока наследуется от *абстрактного* класса **`GThread`**, в котором определяется базовая функциональность для всех грид-потоков;
  - перед объявлением класса необходимо поместить атрибут **`Serializable`** для того, чтобы объект был снабжен методами сериализации и десериализации

Задание 2 – Разработка класса грид-потока

- Основные этапы реализации класса грид-потока:
  - определить все переменные, которые могут потребоваться *для вычислений* и последующего *восстановления результата* на машине пользователя из имеющихся фрагментов, вычисленных отдельными грид-потоками *Открыть*
  - создать *конструктор* для класса грид-потока. Действия конструктора, как правило, сводятся к инициализации полей класса и выделению необходимых ресурсов *Открыть*
  - реализовать *основной* метод грид-потока, в котором будет производиться обработка заданной порции данных – метод **`void Start()`**. В каждой конкретной реализации данный метод необходимо переопределять, заменяя ключевое слово **`abstract`** словом **`override`** *Перейти*
  - для того чтобы получить доступ к необходимым полям класса и при этом не нарушить инкапсуляцию данных, в конце класса определить *свойства* для доступа к данным полям *Открыть*

Задание 3 – Локальный запуск грид-потоков…

- Прежде чем организовывать исполнение грид-потоков в Alchemi-грид желательно произвести тестирование вычислительного кода на локальной машине. Выявленные на этом этапе ошибки будет проще устранить
- Для более удобной обработки грид-потоков на локальной машине лучше всего создать вспомогательный класс **`LocalAppication`**, семантически близкий классу **`GApplication`**, предназначенному для запуска грид-потоков в вычислительной грид  *Открыть*
- Данный класс является универсальным и может быть использован для тестирования грид-потоков в *любом* приложении
- Дополнительная выгода от использования данного класса состоит в том, что для запуска грид-потоков в вычислительной грид достаточно будет изменить всего несколько строк кода

Задание 3 – Локальный запуск грид-потоков…

- Запуск грид-потоков осуществляется в основном приложении. Для того чтобы открыть заготовку кода, предназначенную для дальнейшей доработки, выберите в окне **Solution Explorer** проект **DistributedMegaPOV**, затем правой кнопкой мышки щелкните на файле **MainForm.cs** и в появившемся контекстном меню выберите команду **View Code**  ***Открыть***
- Обратите внимание, что перед объявлением класса необходимо подключить библиотеки **RenderThread** и **Alchemi** (последняя подключается в *любом* проекте), а также некоторые дополнительные системные библиотеки

Задание 3 – Локальный запуск грид-потоков…

- Возможная схема реализации запуска грид-потоков на локальной машине (с использованием класса **`LocalApplication`**):
  - определить все глобальные переменные, которые могут потребоваться *во время вычислений* и для *сохранения результатов* *Открыть*
  - реализовать метод для обработки *успешно завершившихся* грид-потоков. В нашем приложении данный метод называется **`void UpdateProgress(GThread thread)`** *Открыть*
  - реализовать метод для обработки *неудачно завершившихся* грид-потоков. В нашем приложении *не предусмотрен*
  - реализовать обработчики событий **`ThreadFinish`** и **`ThreadFailed`**, возникающих при *успешном* и *неудачном* завершении работы грид-потока. Действия данных обработчиков сводятся к вызову описанных выше методов из *основного* потока приложения *Открыть*
  - реализовать метод для запуска грид-потоков на локальной машине с использованием класса **`LocalApplication`**. В нашем приложении данный метод называется **`void RenderImageLocal()`** *Открыть*

Задание 3 – Локальный запуск грид-потоков

- Замечания о локальном запуске грид-потоков:
  - использование нескольких грид-потоков в данном случае *не приводит к значительному* ускорению вычислений. Грид-потоки обрабатываются последовательно в отдельном потоке. Следующий грид-поток запускается лишь тогда, когда предыдущий завершает свою работу;
  - однако такой подход позволяет быстро и просто отладить приложение и с минимальными усилиями реализовать обработку грид-потоков в вычислительной сети

Задание 4 – Тестирование приложения

- Для того чтобы откомпилировать и запустить приложение выполните команду **Debug | Start Debugging** или нажмите клавишу **F5**
- Убедитесь в корректности настроек приложения, таких как пути к рабочему каталогу приложения MegaPOV и каталогу временных файлов. Для этого выполните команду **File | Path Settings**, в результате которой на экране появится диалоговое окно с настройками
- Выберите файл сцены для рендеринга с помощью команды **File | Open Scene**. Напомним, что множество готовых сцен хранятся в подкаталоге **scenes** установочного каталога программы POV-Ray

Для обработки сцены на локальной машине щелкните на кнопке ***Render Image on Local***

Задание 5 – Запуск грид-потоков в грид…

- Возможная схема реализации запуска грид-потоков в вычислительной грид (с использованием класса **`GApplication`**):
  - определить все глобальные переменные, которые могут потребоваться *во время вычислений* и для *сохранения результатов*  *Открыть*
  - реализовать метод для обработки *успешно завершившихся* грид-потоков. В нашем приложении данный метод называется **`void UpdateProgress(GThread thread)`**
  - реализовать метод для обработки *неудачно завершившихся* грид-потоков. В нашем приложении не предусмотрен
  - реализовать обработчики событий **`ThreadFinish`** и **`ThreadFailed`**, возникающих при *успешном* и *неудачном* завершении работы грид-потока. Действия данных обработчиков сводятся к вызову описанных выше методов из *основного* потока приложения

Задание 5 – Запуск грид-потоков в грид

- Продолжение:
  - реализовать метод для запуска грид-потоков в вычислительной сети с использованием класса **`GApplication`**. В нашем приложении данный метод называется **`void RenderImageGrid()`** *Открыть*
  - предусмотреть уничтожение экземпляра грид-приложения при выходе из программы. Если этого не сделать, на Менеджере не освободятся ресурсы, связанные с приложением (в том числе ресурсы оперативной памяти) *Открыть*

Задание 6 – Тестирование приложения

- Для того чтобы откомпилировать и запустить приложение выполните команду **Debug | Start Debugging** или нажмите клавишу **F5**
- Перед тем, как работать с программой, на машине разработчика следует создать минимальную “грид” из одного Менеджера и одного Исполнителя. При наличии многоядерной или многопроцессорной машины желательно запускать несколько Исполнителей для более эффективного использования вычислительных ресурсов
- Выберите файл сцены для рендеринга с помощью команды **File | Open Scene**. Напомним, что множество готовых сцен хранятся в подкаталоге **scenes** установочного каталога программы POV-Ray

Для обработки сцены в вычислительной сети щелкните на кнопке ***Render Image on Grid***

Заключение…

- В данной лабораторной работе рассматривался вопрос внедрения в грид существующего приложения на примере инструментария Alchemi
- В качестве программы для внедрения был выбран известный пакет для рендеринга компьютерных сцен POV-Ray с предустановленной надстройкой MegaPOV
- Инструментарий Alchemi предлагает хорошие возможности для внедрения существующих приложений в грид, которые вместе с тем являются довольно прозрачными в сравнении, например, с инструментарием GPE
- Тем не менее, процесс внедрения приложений в грид с помощью модели грид-потоков является *низкоуровневым*, т.к. требует разработки специального грид-приложения “с нуля”

Отметим некоторые преимущества и недостатки Alchemi, связанные с внедрением приложений в грид

Заключение…

- К *преимуществам* инструментария Alchemi можно отнести:
  - ***простота реализации и компактность кода***: обеспечивается благодаря развитой архитектуре высокоуровневых классов Microsoft .NET Framework
  - ***возможность быстро и удобно проектировать пользовательский интерфейс:*** инструментарий Alchemi использует все возможности библиотеки пользовательского интерфейса Windows Forms
  - ***легкость в отладке и использовании приложений:*** разработанные классы грид-потоков можно сначала отладить на локальной машине, и лишь затем осуществлять их запуск в грид
  - ***возможность запуска разработанных приложений на локальных компьютерах:*** разработанные приложения можно запускать на локальных компьютерах без установленных компонент инструментария Alchemi. Любая программа для инструментария Alchemi может использоваться как обычное приложение для операционной системы Microsoft Windows с установленной платформой Microsoft .NET Framework

Заключение

- К *недостаткам* инструментария Alchemi можно отнести:
  - ***отсутствие переносимости кода и ориентация на единственную платформу**:* на сегодняшний день отсутствует *полноценная* замена платформы Microsoft .NET Framework для других операционных систем. Данный факт делает невозможным запуск программ, написанных с использованием инструментария Alchemi, под другими операционными системами (однако прогресс в этом направлении не стоит на месте – межплатформенная разработка Mono предлагает неплохую совместимость);
  - ***ограниченный класс задач из-за отсутствия обмена данными между грид-потоками**: н*а сегодняшний день инструментарий Alchemi не имеет поддержки средств обмена данными между грид-потоками. Данный факт несколько ограничивает круг применения инструментария;
  - ***слабая масштабируемость инструментария***: инструментарий Alchemi лучше всего подходит для объединения в вычислительную сеть нескольких десятков компьютеров

Вопросы для самопроверки…

1. Какие модели программирования предоставляет инструментарий Alchemi? В чем их особенности?
2. В каких ситуациях следует применять модель грид-потоков?
3. В чем состоят основные этапы разработки приложения с помощью модели грид-потоков?
4. Каким требованиям должна удовлетворять вычислительная схема?
5. Опишите два стандартных вида параллелизма.
6. Опишите два стандартных вида вычислительных схем.
7. В чем состоят основные этапы разработки класса грид-потока?
8. Каким образом лучше всего отлаживать класс грид-потока?

Дополнительные упражнения

1. Добавить возможность остановки вычислений при их запуске как на локальном компьютере, так и в вычислительной грид
2. Реализовать корректную обработку неудачно завершившихся грид-потоков и последующий их перезапуск на доступных Исполнителях

Литература

1. Никулин Е. А. Компьютерная геометрия и алгоритмы машинной графики. – СПб.: БХВ-Петербург, 2003. – 560 с.
2. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal. Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework.

   http://www.gridbus.org/~alchemi/files/alchemi_bookchapter.pdf
4. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal. Alchemi: A .NET-based Enterprise Grid Computing System.

   http://www.gridbus.org/%7Eraj/papers/alchemi_icomp05.pdf
5. Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal. Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids.

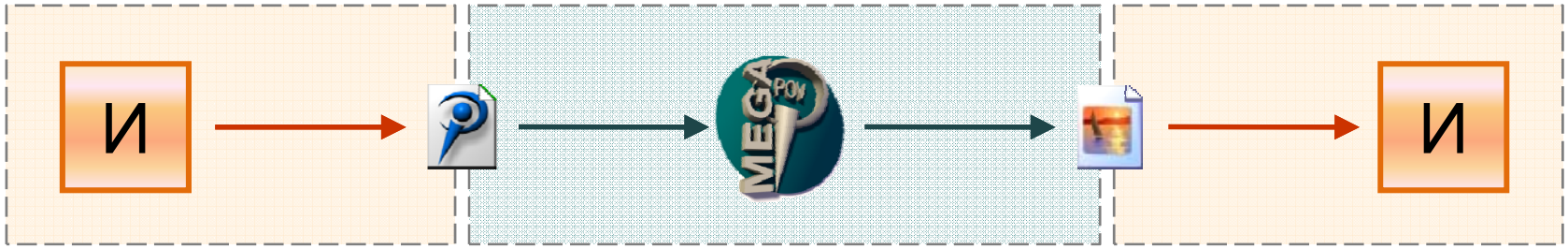
   http://www.gridbus.org/~alchemi/files/alchemi_techreport.pdf

Литература

6. http://www.alchemi.net – официальный сайт проекта Alchemi
7. http://www.povray.org – официальный сайт проекта POV-Ray
8. http://megapov.inetart.net – официальный сайт проекта MegaPOV

Реализация метода Start грид-потока

- Реализация метода **`void Start()`** разбивается на три этапа:
  - сохранение файла компьютерной сцены во временный каталог;
  - запуск приложения MegaPOV для обработки определенной части данной сцены и ожидание окончания его работы;
  - загрузка из временного каталога сгенерированного изображения и извлечение из него полезной части *Открыть*



Вернуться