ЗАО «АЙТИ»

УТВЕРЖДАЮ

Генеральный директор
ЗАО «АйТи».

Бакиев О.Р.

"__" _____ 2011 г.

Создание высокотехнологичного производства комплексных решений в области предметно-ориентированных облачных вычислений для нужд науки, промышленности, бизнеса и социальной сферы

ПРОГРАММНЫЙ КОМПОНЕНТ КЛИЕНТСКОЙ ОБРАБОТКИ И ВИЗУАЛИЗАЦИИ ДАННЫХ CLAVIRE/HYPERLAB

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 63

2011

Инв. № подл.	Подпись и дата	Взам. инв. №	Инв. № дубл.	Подпись и дата

ЗАО «АЙТИ»

УТВЕРЖДЕН
RU.СНАБ.80066-06 13 63-ЛУ

МНОГОПРОФИЛЬНАЯ ИНСТРУМЕНТАЛЬНО-ТЕХНОЛОГИЧЕСКАЯ ПЛАТФОРМА СОЗДАНИЯ И УПРАВЛЕНИЯ РАСПРЕДЕЛЕННОЙ СРЕДОЙ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ CLAVIRE

ПРОГРАММНЫЙ КОМПОНЕНТ КЛИЕНТСКОЙ ОБРАБОТКИ И ВИЗУАЛИЗАЦИИ ДАННЫХ CLAVIRE/HYPERLAB

ОПИСАНИЕ ПРОГРАММЫ

RU.СНАБ.80066-06 13 63

ЛИСТОВ 18

2011

Инв.№ подл.	Подп. и дата	Взам.инв.№	Инв.№ дубл.	Подп. и дата

АННОТАЦИЯ

Документ содержит описание компонента клиентской обработки и визуализации данных CLAVIRE/HyperLab RU.СНАБ.80066-06 01 63, обеспечивающего возможность визуализации данных с использованием доступных программных и технических средств (в т.ч. с использованием многофункциональных и предметно-ориентированных систем визуализации результатов, включая широкоэкранные системы виртуальной реальности типа 3DWall). Программный компонент клиентской обработки и визуализации данных разработан в ходе выполнения проекта «Создание распределенной вычислительной среды на базе облачной архитектуры для построения и эксплуатации высокопроизводительных композитных приложений» в рамках реализации постановления Правительства РФ № 218 «О мерах государственной поддержки развития кооперации российских высших учебных заведений и организаций, реализующих комплексные проекты по созданию высокотехнологичного производства».

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

Компонент клиентской обработки и визуализации данных CLAVIRE/HyperLab RU.СНАБ.80066-06 01 63 позволяет визуализировать данные с использованием доступных программных и технических средств (в т.ч. многофункциональных и предметно-ориентированных систем визуализации результатов).

Компонент клиентской обработки и визуализации данных реализует технологию визуального конструктора, который позволяет пользователю компоновать собственные предметно-ориентированные визуализаторы результатов моделирования путем соединения технологических узлов различного назначения в графическом редакторе. Это позволяет гибко определять объекты и способ их визуализации, обрабатывать различные устройства пользовательского интерфейса, такие как мышь, клавиатура, 3D-мышь, сенсорный экран (или система захвата движения) в системах класса 3DWall, и гибко настраивать реакцию системы на действия пользователя. Как следствие, это дает возможность строить на основе компонента CLAVIRE/HyperLab проблемно-ориентированные среды виртуальной реальности, отображающие результаты расчетов в распределенной среде на основе МИТП CLAVIRE.

Данный компонент разработан на языке C# в виде приложения Windows с применением технологии XNA.

Компонент функционирует на аппаратных системах с архитектурой процессора x86. Для его работы необходимо следующее системное программное обеспечение: ОС семейства Windows NT (версии старше Windows 2000), .NET 4.0, XNA Framework Redistributable 4.0

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

Для решения задачи эффективной организации процессов обработки и визуализации данных в рамках компонента CLAVIRE/HyperLab реализованы следующие функции.

1. Формирование графового представления композитной процедуры визуализации. Узлы графа при этом представляют собой функциональные элементы, отвечающие за атомарные операции обработки или визуализации данных.
2. Автоматическая обработка и визуализация данных, доступных в хранилище МИТП CLAVIRE.

3. Визуализация данных с использованием разнородных доступных программных и технических средств (в т.ч. многофункциональных и предметно-ориентированных систем визуализации данных).
4. Взаимодействие с управляющим ядром CLAVIRE в процессе визуализации и обработки данных. При этом вызов рассматривается как элемент цепочки заданий, реализуемой в МИТП CLAVIRE.
5. Взаимодействие с пользователем с применением доступных технических и программных средств (интерактивная визуализация), с использованием специфических интерактивных средств виртуальной реальности в системах 3DWall (3D мышь, система трекинга, сенсорная поверхность).
6. Поддержка широкоэкранной стерео-визуализации за счет формирования стерео-пары в режиме Dual-Screen (на левом экране формируется изображение для левого глаза, на правом - для правого) и трансляции ее на оборудование системы класса 3DWall.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Принципы функционирования

Вычислительный граф состоит из узлов и связей между ними. Каждый узел имеет несколько входов и/или выходов. Связи соединяют выходы одних узлов со входами других узлов и являются ребрами графа, вдоль которых передаются данные. При перерасчете графа перерасчет осуществляется только для тех узлов, входные данные которых изменились. Такой подход, в частности, используется в пакете Autodesk Maya, позволяя осуществлять гибкую визуализацию и физические расчеты, также он был опробован при обработке и визуализации уровня воды Финского залива.

Вычислительный граф реализован на языке C# с использованием технологии XNA, которая предоставляет возможности интерактивной визуализации и взаимодействия с различными устройствами ввода, такими как клавиатура, мышь, джойстик и тач-скрин. Реализация узла графа представляет собой класс на языке C#, наследованный от класса Node (предоставляемый МИТП).

Для определения входов и выходов узла используются атрибуты .NET: атрибут In помечает поле класса как входное, Out – как выходное.

При реализации класса необходимо:

1) определить метод Compute, описывающий основную операцию обработки данных, реализуемую узлом;
2) если узел помимо вычислительных задач выполняет фоновую работу, то необходимо дополнительно определить метод Update, который вызывается на каждом кадре (шаге обработки);
3) если узел выполняет задачи визуализации средствами XNA, то необходимо переопределить методы LoadContent (вызывается при создании узла, а также восстановлении устройства воспроизведения) и Draw (выполняет визуализацию данных в соответствии с функциональностью узла).

Приведем пример реализации узла, который складывает два числа, на языке C#.

```
public class Add : Node {
  [In]      public double a;
  [In]      public double b;
  [Out] public double c;
  public override void Compute () {
    c = a + b;
  }
}
```

Граф создается пользователем с помощью визуального интерфейса путем перемещения блоков, изображающих узлы, мышью и их соединения (см. рис. 3.1). При этом у пользователя имеется возможность запуска, остановки и прекращения обработки построенного графа при помощи соответствующих кнопок интерфейса.

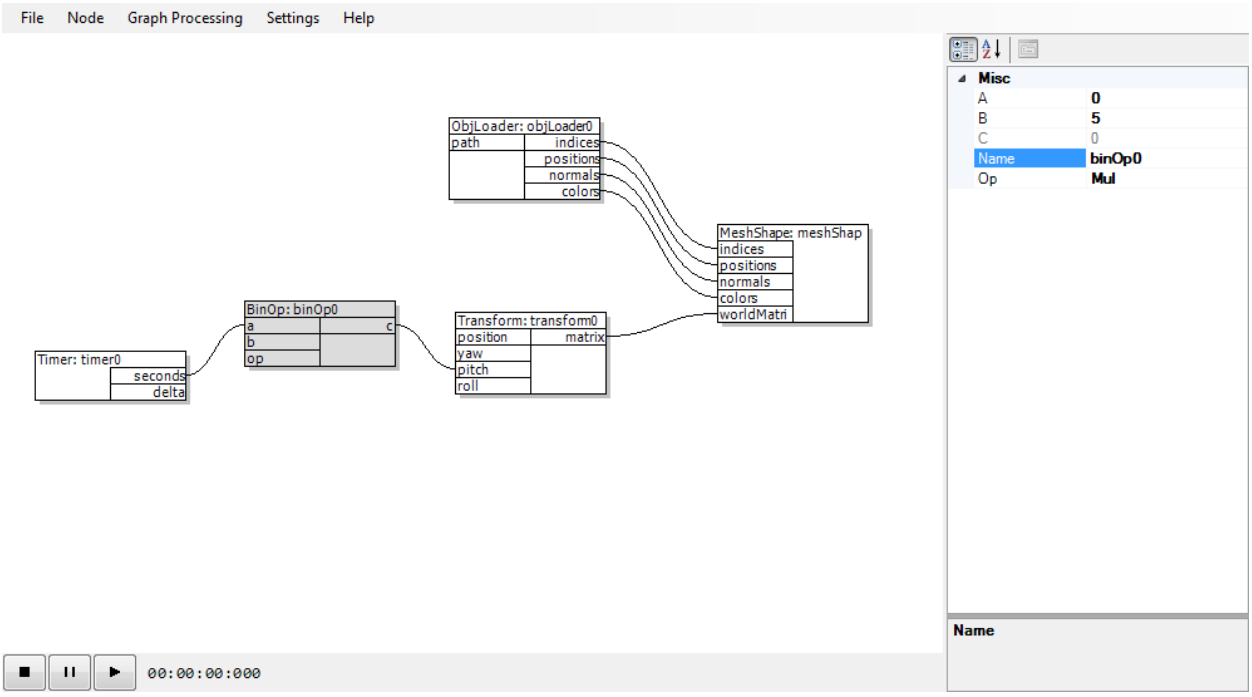

Рисунок 3.1– Визуальный интерфейс редактора вычислительного графа

Алгоритм функционирования графа приведен на рис. 3.2. Такой подход позволяет выполнять вычисления только, если данные изменились. В противном случае данные кэшируются на входах и выходах. Следует отметить, что данные могут быть переданы только между выходами и входами, совместимыми по типу. Если типы входа и выхода, требующих совмещения, различаются, необходимы реализация конвертера соответствующих типов, регистрация его в компоненте и использование в качестве элемента графа.

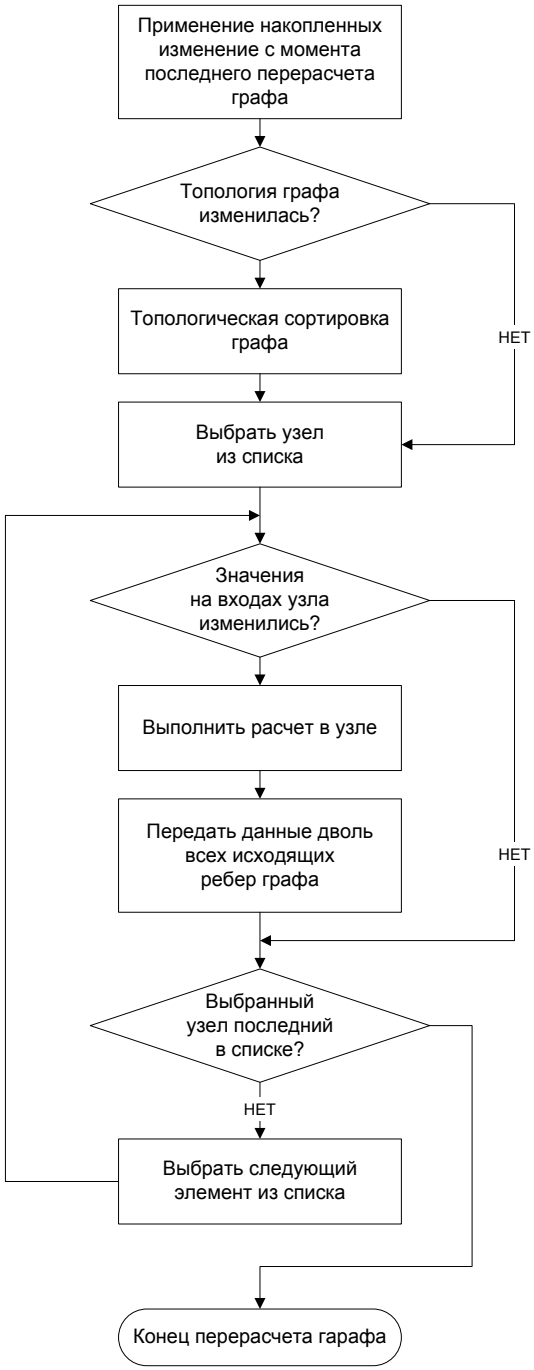

Рисунок 3.2 – Алгоритм перерасчета графа

3.2. Программная архитектура

Программно компонент клиентской обработки и визуализации данных состоит из модуля редактора графа (см. рис. 3.1), модуля обработки и визуализации графа и коллекции классов узлов графа (которые могут находиться в подключаемых модулях), которые реализуют те или иные функции. Тексты всех модулей расположены в файлах с расширением *.cs. Диаграмма основных классов компонента представлена на рис. 3.3.

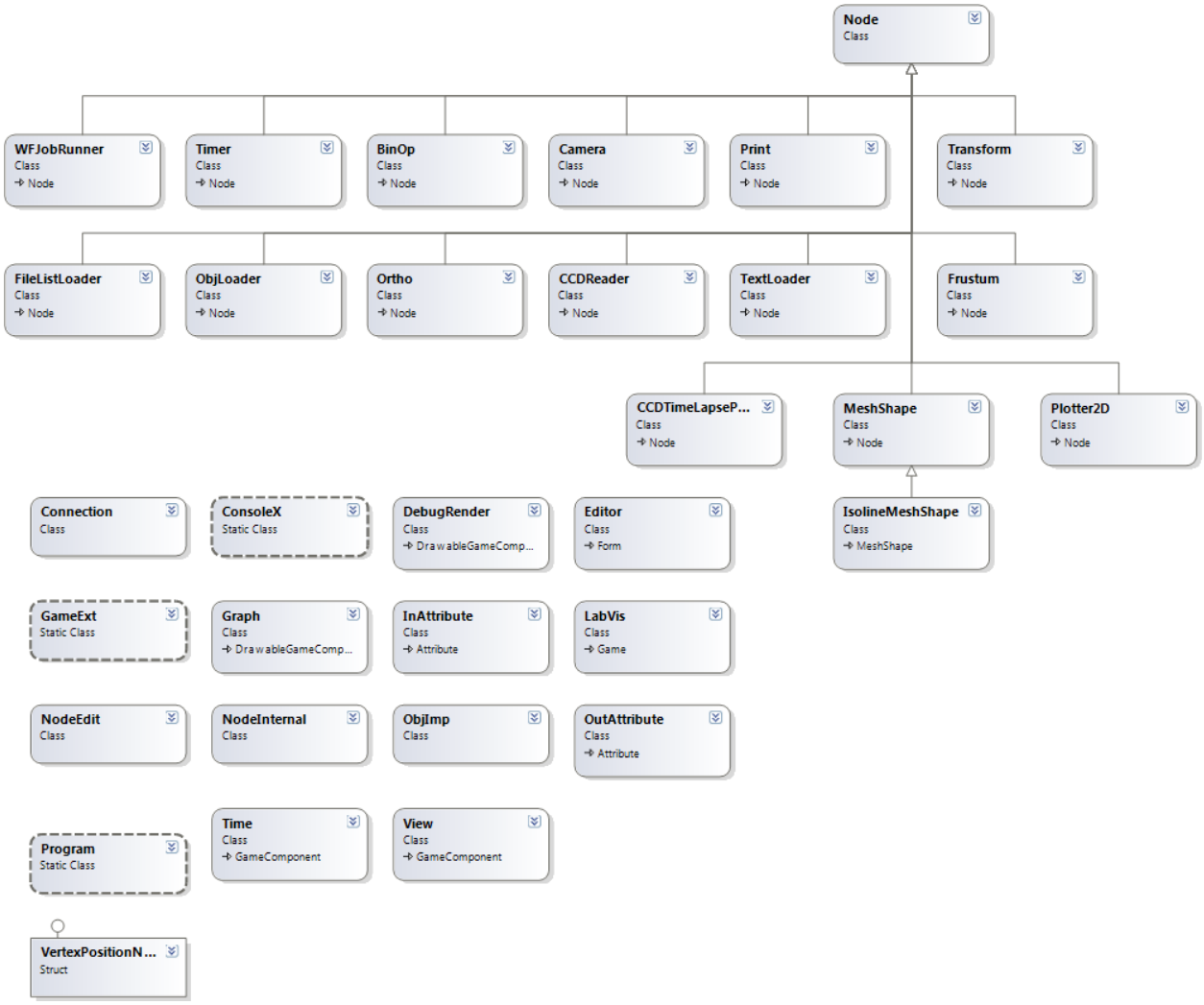

Рисунок 3.3 – Диаграмма классов

Более подробно структура классов, используемых программным компонентом, рассмотрена в разделах 3.3–3.4.

3.3. Основные классы графа

Ниже приводятся сокращенные описания структуры и методов основных классов компонента клиентской обработки и визуализации данных.

3.3.1. Класс Graph

- void Update (GameTime gameTime) – вызывается приблизительно на 60 раз в секунду, для выполнения фоновых действий
  a) gameTime – хранит текущее время, и время, прошедшее с предыдущего вызова функции Update().
- void Draw (GameTime gameTime) – вызывается на приблизительно 60 раз в секунду, для визуализации узлов.
  a) gameTime – хранит текущее время, и время, прошедшее с предыдущего вызова функции Draw()
- void RenewGraph () – удаляет все узлы и ребра между ними.
- void Retopologize () – удаляет дерево топологической сортировки.
- void AddNode (string name, string className) – создает узел указанного класса и указанными именем
  a) name – имя узла;
  b) className – имя класса узла.
- string AddNode (string className) – создает узел указанного класса, уникальное имя узла присваивается автоматически
  a) className – имя класса узла;
  b) возвращаемое значение – имя созданного узла.
- void AddNode (Node node) – добавляет уже созданный узел;
  a) node – узел, которые будет добавлен в граф.
- void Set (string dst, object value) – устанавливает значение атрибута узла. Изменение вступит в силу после вызова функции CommitChanges()
  a) dst – адрес атрибута узла, значение которого надо установить, вида: "имяУзла.имяАтрибута".
- string GenerateName (string className) – генерирует уникальное имя вида "имяКлассаXXXX", где XXXX – число, обеспечивающее уникальность имени,
  a) className – имя класса узла;
  b) возвращаемое значение – уникальное имя узла.
- void CommitChanges () – вносит изменения.
- void Wire (string src, string dst) – соединяет атрибуты узлов
  a) src – адрес выходного узла;

b) dst – адрес входного узла.

- void Break (string src, string dst) – разрывает соединение между узлами
  a) src – адрес выходного узла;
  b) dst – адрес входного узла.
- void RemoveNode (string name) – удаляет узел с указанным именем
  a) name – имя узла.
- void RenameNode (string oldName, string newName) – переименовывает указанный узел. Если указанное новое имя не уникально, генерируется уникальное
  a) oldName – имя узла, который будет переименован;
  b) newName – новое имя узла.
- void Evaluate (bool force=false) – выполняет перерасчет графа
  a) force – если «ложь», то перерасчет будет осуществлен только для тех узлов, атрибуты которых изменились, в противном случае – для всех узлов.

3.3.2. Класс Node

Node – базовый для всех классов узлов.

- void Compute () – выполняет расчет значений выходных атрибутов узла.
- void LoadContent () – загружает графический контент.
- void Update (GameTime gameTime) – выполняет фоновую работу
  a) gameTime – хранит текущее время и время прошедшее с предыдущего вызова функции Update ()
- void Draw (GameTime gameTime) – визуализирует узел.
  a) gameTime – хранит текущее время, и время, прошедшее с предыдущего вызова функции Draw()

3.4. Классы узлов графа

Программные средства для реализации вычислительного графа включают поддержку узлов разных типов:

1) общего назначения;
2) поддержки визуализации;
3) взаимодействия со средствами ввода/вывода;
4) связи с CLAVIRE;
5) реализующие пользовательские функции.

Каждый тип узлов поддерживает свой набор методов и включает уникальные поля данных. В табл. 3.1 приведено описание параметров для основных узлов общего назначения. К узлам этого типа относятся:

- Timer: узел возвращает время с момента последнего нажатия на кнопку Play. Используется для анализа на клиентской машине процессов, развивающихся во времени.
- BinOp: бинарная арифметическая операция.
- TextLoader: загружает текст из файла.
- FileListLoader: загружает текст из набора файлов.

Таблица 3.1

Параметры узлов общего назначения

Timer		
Seconds	Out, double	Время с момента нажатия кнопки Play в интерфейсе в секундах
Delta	Out, double	Интервал между кадрами (в секундах)
BinOp		
A	In, double	Аргумент A
B	In, double	Аргумент B
C	Out, double	Результат C
Op	In, enum	Код операции (Add, Sub, Mul, Div)
TextLoader		
path	In, string	Путь к файлу
Text	In, string	Текст, загруженный из указанного файла
FileListLoader		
Paths	In, List<string>	Путь к файлам
Names	In, List<string>	Имена файлов
outFiles	Dictionary<string, List<byte>>	Словарь с данными загруженных из файлов, индексируемый именами файлов

В табл. 3.2 приведено описание параметров для основных узлов поддержки визуализации. К узлам этого типа относятся:

- Camera: устанавливает матрицы вида и трансформации камеры для окна просмотра трехмерной сцены.
- Frustum: формирует перспективную матрицу проецирования.
- Ortho: формирует ортогональную матрицу проецирования.

- Transform: формирует мировою или видовую матрицу по позиции в пространстве и трем углам.
- MeshShape: визуализирует геометрическую фигуру, заданную массивом индексов и вершин.
- IsolineMeshShape: визуализирует геометрическую фигуру с нанесенными на нее изолиниями. Значения, по которым строятся изолинии, берутся из цвета в вершинах геометрической фигуры.
- ObjMeshLoader: загружает полигональную сеток (массив вершин и индексом) из файла в формате OBJ.
- Plotter2D: построитель двумерных графиков

Таблица 3.2

Параметры узлов поддержки визуализации

Camera		
ViewMatrix	In, Matrix	Матрица вида
ProjMatrix	In, Matrix	Матрица проецирования
Frustum		
Near	In, float	Ближняя плоскость отсечения
Far	In, float	Дальняя плоскость отсечения
Fov	In, float	Поле зрения (в градусах) по вертикали, поле зрения по горизонтали определяется соотношением сторон окна просмотра
Matrix	Out, Matrix	Результирующая матрица
Ortho		
Near	In, float	Ближняя плоскость отсечения
Far	In, float	Дальняя плоскость отсечения
Width	In, float	Ширина видовой области
Matrix	Out, Matrix	Результирующая матрица
Transform		
Yaw	In, float	Угол поворота вокруг поперечной вертикальной оси
Pitch	In, float	Угол поворота вокруг поперечной горизонтальной оси
Roll	In, float	Угол поворота вокруг продольной оси
Position	In, float	Позиция в пространстве
Matrix	Out, Matrix	Результирующая матрица
MeshShape		
Indices	In, List<int>	Массив индексов
Positions	In, List<Vector3>	Массив позиций вершин

Normals	In, List<Vector3>	Массив нормалей в вершинах
Colors	In, List<Color>	Массив цветов в вершинах
worldMatrix	In, Matrix	Мировая матрица
IsolineMeshShape		
Indices	In, List<int>	Массив индексов
Positions	In, List<Vector3>	Массив позиций вершин
Normals	In, List<Vector3>	Массив нормалей в вершинах
Colors	In, List<Color>	Массив цветов в вершинах
worldMatrix	In, Matrix	Мировая матрица
lowValue	In, double	Минимальное значение
highValue	In, double	Максимальное значение
palettePath	In, string	Путь к палитре
ObjMeshLoader		
Indices	Out, List<int>	Массив индексов
Positions	Out, List<Vector3>	Массив позиций вершин
Normals	Out, List<Vector3>	Массив нормалей в вершинах
Colors	Out, List<Color>	Массив цветов в вершинах
Path	In, Matrix	Путь к файлу
Plotter2D		
xPos	In, int	Позиция графика по оси X
yPos	In, int	Позиция графика по оси Y
Width	In, int	Ширина графика
Height	In, int	Высота графика

В табл. 3.3 приведено описание параметров для основных узлов взаимодействия со средствами ввода/вывода. К узлам этого типа относятся:

- Keyboard: возвращает массив нажатых клавиш.
- Mouse: возвращает состояние мыши.
- Mouse3D: возвращает состояние 3D-мыши или положение объекта в системе трекинга (величина с шестью степенями свободы).
- TouchScreen: возвращает состояние сенсорного экрана.

Таблица 3.3

Параметры узлов взаимодействия со средствами ввода/вывода

Keyboard		
Keys	Out, List<Keys>	Массив нажатых клавиш

Mouse		
X	Out, int	Координаты мыши по оси X
Y	Out, int	Координаты мыши по оси Y
Left	Out, bool	Флаг нажатия на левую кнопку мыши
Right	Out, bool	Флаг нажатия на правую кнопку мыши
Middle	Out, bool	Флаг нажатия на среднюю кнопку мыши
Mouse3D		
X	Out, double	Смещение мыши по оси X
Y	Out, double	Смещение мыши по оси Y
Z	Out, double	Смещение мыши по оси Z
RX	Out, double	Поворот мыши по оси X
RY	Out, double	Поворот мыши по оси Y
RZ	Out, double	Поворот мыши по оси Z
TouchScreen		
Touches	Out, List<Vector2>	Координаты мыши по оси X
Y	Out, int	Координаты мыши по оси Y

В табл. 3.4 приведено описание параметров для основных узлов взаимодействия с CLAVIRE. К узлам этого типа относятся:

- WFJobRunner: запускает workflow на исполнение.
- WFReadChannel: канал приема данных LRWF.
- WFWriteChannel: возвращает состояние 3D-мыши. 3D-мышь представляет собой манипулятор с шестью степенями свободы [3DMouse].

Таблица 3.4

Параметры узлов взаимодействия со средствами ввода/вывода

WFJobRunner		
Script	In, string	Текст WF
InFiles	In, Dictionary<string, List<byte>>	Входные файлы в бинарном виде
OutFiles	Out, Dictionary<string, List<byte>>	Входные файлы в бинарном виде
WFReadChannel		
EndPoint	In, string	Адрес конечной точки
DataName	In, string	Имя канала данных

Data	Out, object	Принимаемый объект
WFWriteChannel		
EndPoint	In, string	Адрес конечной точки
DataName	In, string	Имя канала данных
Data	Out, object	Отправляемый объект

Узлы, реализующие пользовательские функции, не имеют строгой нотации. Реализация пользовательских функций в узлах вычислительного графа на примере чтения данных полей уровня воды (результат гидродинамического моделирования) показана в табл. 3.5.

- CCDReader: загружает данные из файла с данными уровня в списке, соответствующего времени, указанному на входе time.
- CCDTimeLapsePointReader: загружает данные из CCD-файла в списке, соответствующего времени указанном на входе time.

Таблица 3.5

Параметры узлов чтения данных уровня моря (пример пользовательского узла)

CCDReader		
Time	In, Time	Время
InFiles	In, Dictionary<string, List<byte>>	Входные файлы в бинарном виде
Indices	Out, List<int>	Массив индексов
Positions	Out, List<Vector3>	Массив позиций вершин
Normals	Out, List<Vector3>	Массив нормалей в вершинах
Colors	Out, List<Color>	Массив цветов в вершинах, канал R хранит значение уровня воды по Балтийской системе высот (в сантиметрах)
CCDTimeLapsePointReader		
InFiles	In, Dictionary<string, List<byte>>	Входные файлы и бинарном виде
pointX	In, int	Точка, в которой необходимо считать значения
pointY	In, int	Точка, в которой необходимо считать значения
values	Out, List<Vector2>	Значения уровня воды во времени

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Программный компонент предназначен для работы на пользовательской ЭВМ со следующей конфигурацией:

- архитектура процессора – x86;
- свободное место на диске – не менее 10 ГБ;
- размер оперативной памяти – не менее 2 ГБ;
- тип видеоускорителя – nVidia GeForce GTX260;
- объем видеопамяти – не менее 1 ГБ;
- операционная система – Windows Vista, Windows 7;
- установленные программные компоненты – Microsoft XNA Framework Redistributable 4.0, .NET Framework 4.0.

Программный компонент ориентирован на работу с системами виртуальной реальности класса 3DWall на основе следующих технических средств:

- Профессиональный DLP проектор HDTV модель Roxar Projectiondesign (4 или 6 проекторов);
- Модуль обработки 3D-изображения Roxar XPO;
- Модуль управления проекторами и согласования проекторов Roxar XED;
- Профессиональный проекционный экран обратной проекции Roxar;
- Программируемая система управления оборудованием Roxar CDC;
- Масштабатор/преобразователь видеосигналов Kramer VP-729.

5. ВЫЗОВ И ЗАГРУЗКА

Компонент клиентской обработки и визуализации данных представляет собой приложение Windows. Вызов компонента может осуществляться в двух режимах:

1) в режиме интерактивного редактирования графа – посредством прямого запуска файла HyperLab.exe без параметров, что приводит к появлению окна графического редактора (см. рис. 3.1);
2) в пакетном режиме –запуск обработки графа без вывода окна редактора. Запуск в этом режиме выглядит следующим образом:

```
HyperLab.exe -batch <имя файла для запуска>
```

6. ВХОДНЫЕ ДАННЫЕ

Входными данными для компонента является структура графа визуализации. В режиме интерактивного редактирования входные данные формируются пользователем в процессе построения графа в визуальном редакторе. В случае запуска приложения в пакетном режиме входными данными служат бинарные файлы, из которых загружается (десериализуется) структура графа.

7. ВЫХОДНЫЕ ДАННЫЕ

Выходными данными компонента являются динамические графические трехмерные и двумерные образы, формируемые в процесс выполнения расчетов и визуализации с использованием сформированной графовой структуры. Кроме того, выходными данными являются бинарные файлы, в которые сохраняется сформированная или модифицированная пользователем структура графа.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

CCD	Формат хранения данных об уровне воды
PNG	Portable Network Graphics — растровый формат хранения графической информации, использующий сжатие без потерь по алгоритму Deflate
RGBA	Red Green Blue Alpha — аддитивная цветовая модель, расширенная наличием альфа-канала, как правило, используемого для обозначения прозрачности
ОС	Операционная система

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Изм.	**Номера листов (страниц)**				**Всего листов (страниц) в докум.**	**№ документа**	**Входящий № сопроводительного докум. и дата**	**Подп.**	**Дата**
	измененных	**замененных**	**новых**	**аннулированных**					