О согласовании вычислительного эксперимента при интерактивном моделировании гидромеханики корабля в штормовом море

1 – анализ постановки задачи

2 – система функций-алгоритмов

3 – краткий обзор текущего задела

www.ShipDesign.ru/SoftWare

Проектирование эксперимента

интерактивного вычислительного комплекса

Логический иероглиф - *как программирование прямого вычислительного эксперимента*

Трехмерное логическое пространство представляется тремя независимыми (ортогональными) направлениями работ с три уровнями логической градации

Вычислительное окружение
/ ***virtual polygon*** / направления изысканий

- ***Аналитика*** *(задел)*: геометрия и пространственное движение корпуса судна, его гидростатика на взволнованной поверхности моря с учетом распределенных источников излучения корабельных и отраженных волн (*в том числе с возможностью перестроения затапливаемых отсеков*);
- ***Физика*** *(проблема)*: потоки энергии в трохоидальном волнении с оценками распределенного силового воздействия на обшивку корпуса с учетом динамической трансформации и разрушения групповых структур свободных волн;
- ***Эмпирика*** *(поиск)*: непрерывный и повсеместный контроль для динамической адаптации объектов и процессов в системе критериев гидромеханики, с вовлечением результатов из опытовых, эмпирических и асимптотических моделей.

Структура системы моделирования

/ ***virtual polygon*** / - математический иероглиф

«Морской виртуальный полигон» = ***предметное моделирование явления***

Корпус: → ***Волнение:*** → ***Визуализация:***
аналитика – физика – эмпирика .

Window~Place

- Разделение вычислительного комплекса на условно независимые моделирующие блоки

Логические действия в матрице

/ ***virtual polygon*** / - математический иероглиф

Если в проекте программы полагается декларативное или условно статическое описание физических явлений (*компиляция*), то, как следствие, потребуется непрерывный поиск критериев состояния решения (*интерпретация*), с последующими рекурсивными перестроениями операций во времени в зависимости от текущего состояния моделируемых явлений.

Таким образом обеспечивается разделение или независимость декларативного описания и функциональной трансформации виртуальных структур данных, где осуществляется контекстное управление вычислениями по столбцам логического иероглифа, и последовательное формирование результатов моделирования по направлениям строк матрицы.

Штормовая гидромеханика корабля

/ 1.3. ***Window~Place*** – 3D графика OpenGL /

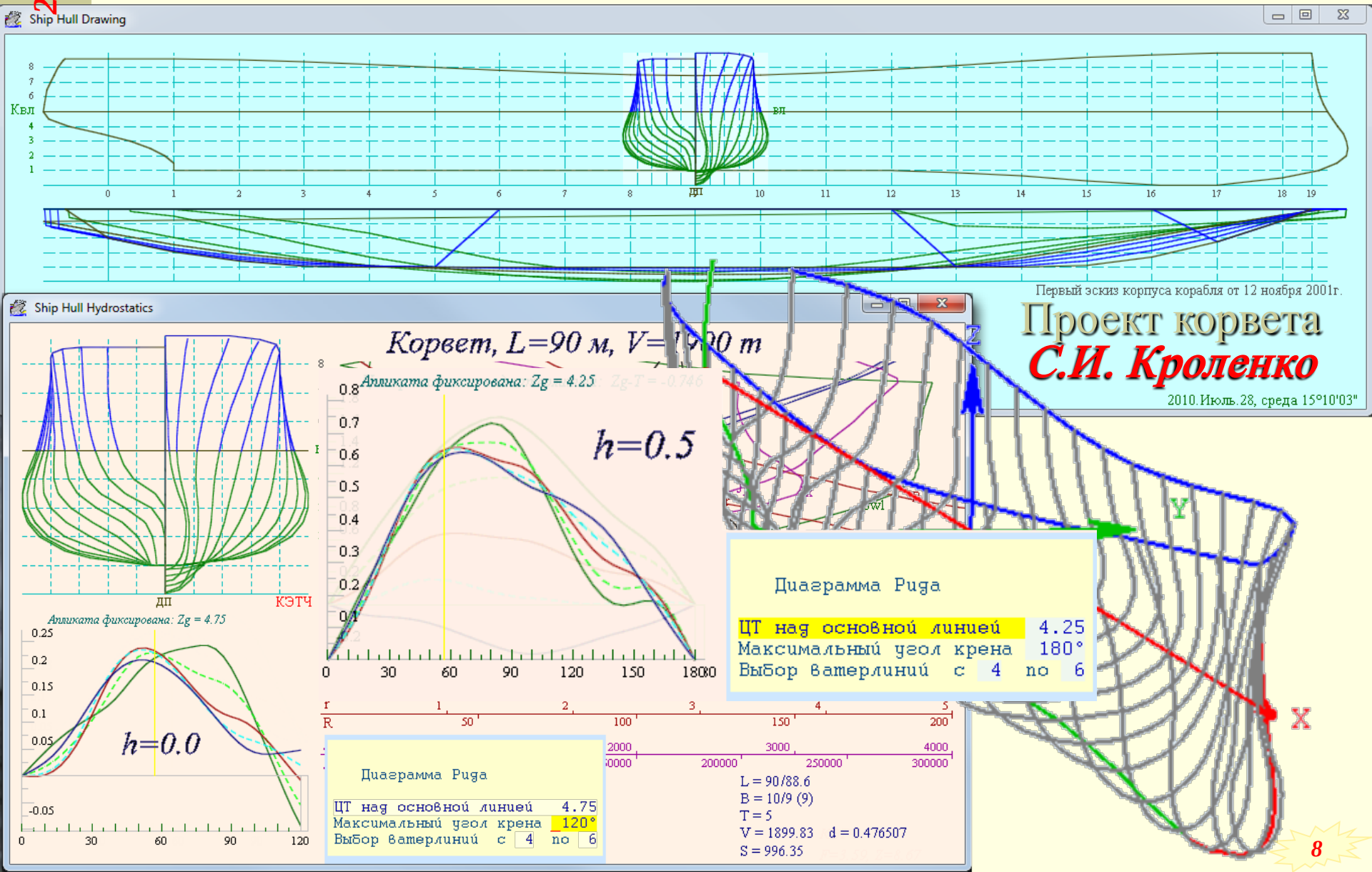
- интерактивная программная среда для проведения, визуализации и динамического управления вычислительным экспериментом;
- Независимая графическая визуализация по таймеру или с приостановкой вычислительных циклов для управляемого перестроения алгоритмов;
- прерывания при возникновении особых условий:
  - сбор аналитической информации в точке;
  - настройка и управление визуализацией;
  - контроль и перенастройка математической модели под локальные условия моделирования.

1.2. Трохоидальное волнение

/ *virtual polygon* /

- *Структура трохоидальной волны*

1.1. Корпус корабля / *virtual polygon* /



3.1. Волнообразование

/ *virtual polygon* /

- *Структура трохоидальной волны*

3.1. Волнообразование / *virtual polygon*

(*излучение ↔ воздействие волн*)

Уравнения Мичелла приводятся к размерным физическим аргументам, что необходимо для анализа корабельного волнообразования как волнопоглощения на корпусе корабля

Минимизируется спектр корабельных волн, включая закритически большие числа Фруда.

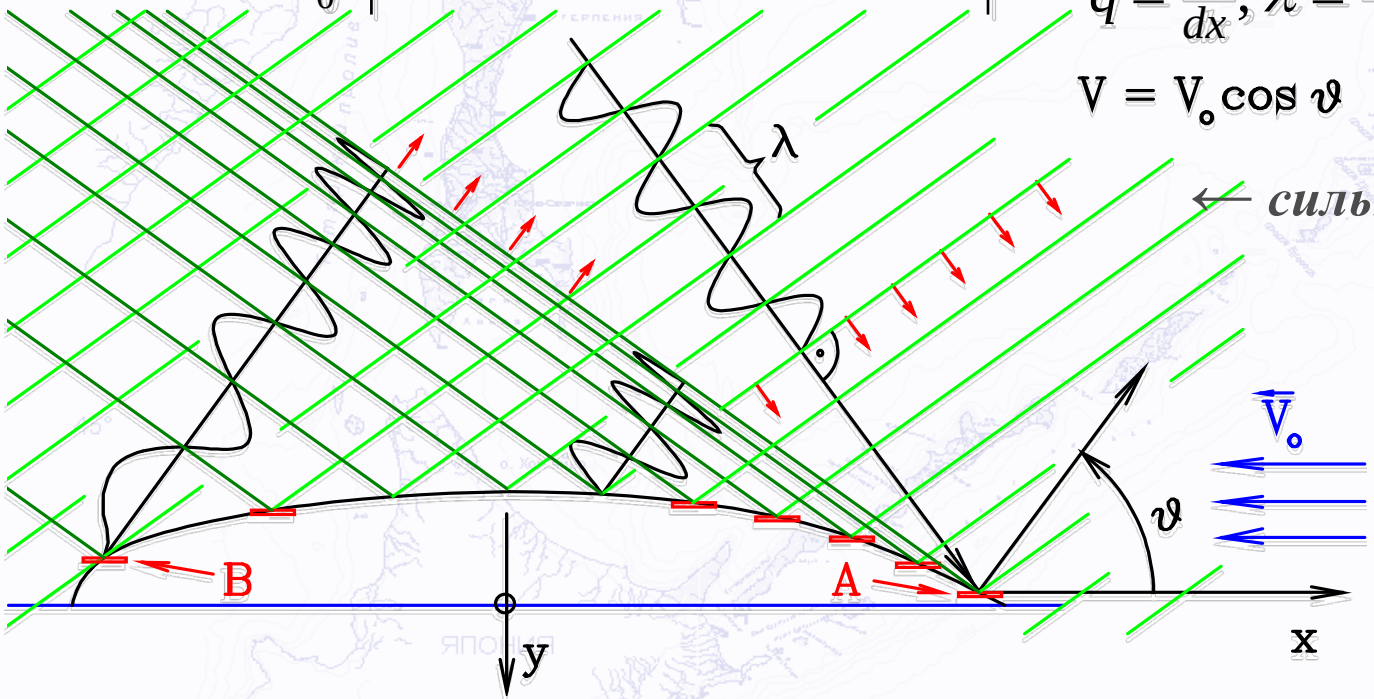
$$R_x = \pi\rho V_0^2 \int_0^{\frac{\pi}{2}} A^2(\vartheta)\cdot\cos^3\vartheta\, d\vartheta$$

$$A(\vartheta) = \frac{g}{2\pi V_0^2}\left|\int_\Omega q(x_0,y_0)\frac{e^{k\cdot(-z_0+i\omega_0)}}{\cos^3\vartheta}\,d\Omega\right|$$

$$\omega_0 = x_0\cos\vartheta + y_0\sin\vartheta$$

$$q = \frac{dy}{dx};\ \lambda = \frac{2\pi}{k} = 2\pi\frac{V_0^2\cos^2\vartheta}{g}$$

$$V = V_0\cos\vartheta \qquad \vec{r}_w = \frac{1}{2}A\cdot e^{-k(z+z_0)}$$



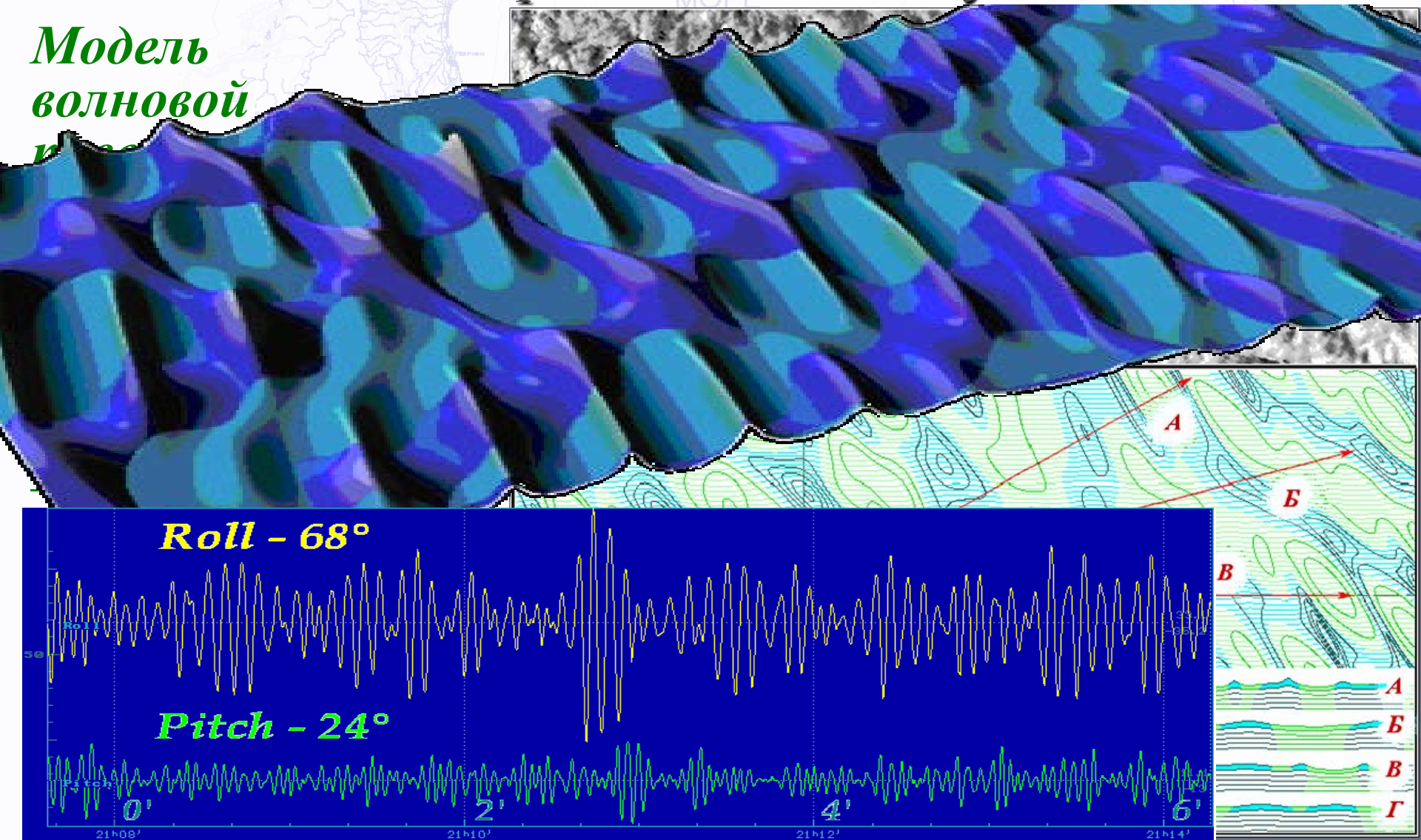
← *сильнейшие искажения отраженных волн на корпусе судна*

В оконечностях **A** *и* **B** *– обрушающийся гребень и отрыв потока. Энергия – суммируется.*

2018-09-11

3.2. Эмпирическое построение нестационарных процессов зарождения и распространения свободных морских волн на глубокой воде

Модель волновой



3.3. Гидромеханика корабля на море

/ **virtual polygon** / - *прямые численные модели*

Строка 1 п.2. *Генерация на свободных границах расчетной области трех пакетов волн в произведении с «огибающими» секторами, смещающимися с половинной скоростью;*

Строка 2. *Инерционное смещение корпуса по приращениям скорости – моделирование качки (без вовлечения в расчеты присоединенных масс)*

Строка 3. *Генерация корабельной волны в результате отражения фронтов коротких фазовых волн и собственно корабельное волнообразование из точечных излучателей в каждой элементарной площадке обшивки.*

Вычислительный эксперимент

/ **virtual polygon** / - *результат*

Заключение

1. *Рассмотрены вопросы логического построения и внутреннего согласования объектов и процессов прикладного вычислительного эксперимента.*
2. *Показан вариант разделения алгоритмов для описания конкретных физических объектов, корпус корабля и штормовое волнение, механика взаимодействия которых управляется функциональными операциями с учетом текущего состояния моделируемой физической среды.*
3. *Особым блоком 1.3 выделено графическое сопровождение вычислительного эксперимента, пронизывающее весь программный комплекс и обеспечивающее интерактивное управление численным моделированием*

Контекстная графическая среда пространственной визуализации результатов вычислительных экспериментов в механике сплошных сред

1 -построение ресурсоемкой задачи

2 - визуализация моделирования

3 - пространственная геометрия

shipdesign.ru/SoftWare Window-Place

Window-Place – *визуализация*

1» множество независимых окон Window с собственным доступом к графической среде OpenGL, к таймерам и клавиатуре.

2» контекстно зависимые графические и текстовые площадки Place в графической среде Window, управляемые в видимой области с помощью указателя «мышь».

3» привязка всех графических площадок к трехмерной тензорной математике !!!

Window-Place – привязки площадок

```
//!  Конструктор сразу прикрепляется к активному
//  окну Window с его контекстом для настроек
Place( byte = PlaceOrtho|PlaceAbove );
Place& Capture(Window *Win=Act->Base);
virtual ~Place(); // с очисткой всех полей и адресов

virtual void Active();  // активизация контекста новой площадки
void Active( void(*)( int W,int H ) ); // => Reshape
void(*inActive)( int W,int H );     // + пересыльная процедура
void Area( int X,int Y,int W,int H ); // установка размеров
//  если X,Y > 0 - отсчеты от левого верхнего угла, иначе - правого нижнего
//  если Width,Height > 0 - отсчеты в символах, если 0 - до границы окна,
//                если < 0 - в пикселях и нормальных отсчетах Y  - снизу вверх

static Place *Act; // Адрес активной страницы – контекстной среды
```

Window-Place – создание окна

```
Window Tvm  // ( null,0,0,0,0 )
( "Example № 1",    // null – без рамки
   ix,iy,640,480 );  // числа можно опустить

// создание, позиционирование, активизация и очистка фрагментов
// +левый/-правый +верхний/-нижний относительно экрана
Tvm.Locate( 50,100,640,480 ); // пересоздание на новом месте
// к Window разумно привязывать лишь клавиатуру и таймер
// и/или вовлекать Window базовым классом для визуализации
//   графических сцен и построения интерактивных приложений

#define Xpm( X )( GetSystemMetrics( SM_CXSCREEN ) * Real( X )/100 )
#define Ypm( Y )( GetSystemMetrics( SM_CYSCREEN ) * Real( Y )/100 )
```

Window-Place – клавиатура

```
Window::                    // внутри класса окна
 Windows
 byte WaitKey()             // ожидание нового
 символа
 byte ScanKey()             // запрос без остановки
 byte ScanStatus()          // запрос кода из буфера
```

```
enum{ RIGHT=1,LEFT,SHIFT,  LCTRL=4,RCTRL=8,CTRL=12,
                           L_ALT=16,R_ALT=32,ALT=48 };

enum Course
{ _North_West=3, _North=1,_North_East=9,  _Home=3,   _Up =1,_PgUp =9,
        _West=2,_Zenith=0,      _East=8,  _Left=2,_Center=0,_Right=8,
  _South_West=6, _South=4,_South_East=12, _End =6, _Down =4,_PgDn=12,
  _Enter=13,_BkSp,_F1,_F2,_F3,_F4,_F5,_F6,_F7,_F8,_F9,_F10,_F11,_F12,
  _Esc=27, _Ins,_Del,_Tab, _Space=32      // +5,+7,+10,+31 - запас
};
```

Window-Place – вывод текста

```
SIZE Alfabet( int size=16,      // привязка
  шрифтов
  const char* font="Courier",  // к площадке Place
  byte weight=FW_DONTCARE, byte italic=false )
SIZE AlfaBit( byte*=CCCP )  // старый растр: 8x08,14,16
SIZE AlfaRect( const char* )  // размер текста
int Text( Course,Real x,Real y,Real z,const char*,...)
void Print( int x,int y,const char*,...)// если y/x<=0, то
void Print( const char*, ... )      // — снизу/справа
<ConIO.h>
      // Включены также все операции вывода
      //   раскрашенных текстов и запросов
      //   с клавиатуры в режиме для консоли
```

Window-Place – часы и таймер

```
// выполняется привязка к целому экрану, т.е. все параметра в динамике
// таймер срабатывает от достигнутого времени, с надбавкой mSec
// -- сам таймер, мышка, и даже клавиатура -- могут разрушить задачу

virtual void Timer(); // виртуальная – полный контроль контекста
void SetTimer( DWORD mSec, bool(*inTimer)()=NULL );
                  // интервал времени mSec и адрес исполнителя
                  // inTimer – аккуратно встраивается в графическую среду
void KillTimer(){ SetTimer( 0,NULL ); } // при любом нуле
bool(*inTimer)(); // внешняя, на выходе признак обновления картинки
bool isTimer;  // метка запущенного по таймеру расчетного процесса
bool isMouse;  // и мышка!, которую надо отводить от таймера
                  // отчего основные процессы уходят в застой/пропуски
```

(Windows – не может обслуживать высокоточную аналоговую аппаратуру)

Window-Place – мышка на площадке

```
// в случае с мышкой возможны виртуальные процедуры производного
//     класса, если ж таких нет, или в них нет переопределений, то
//     исполнятся базовые и внешние функции в контекстной среде
virtual void MouseMove( int x, int y );
virtual void MousePress( int st, int x, int y );
void MouseMove ( void(*Pass)( int, int ) )
               { inPass=Pass; }
void MousePress( void(*Push)( int, int, int ) )
               { inPush=Push; }
void(*inPass )( int X, int Y );    // внешние и независимые
void(*inPush )( int State, int X, int Y );  // процедуры
                                   // обработки прерываний от мышки
//     виртуальные функции отменяют все настройки графической среды
//   позволяя, формально, упрощать или ускорять обработку прерываний
```

Window-Place – многозадачность

```
struct _One: public Window
{ _One():Window( "First OpenGL window" ){};
  void MouseMove( int x, int y ){ ... cprint( ... ); }
  void MousePress( int State,int x,int y ){ cprint... ); }
 ~_One(){ cprint( 1,14,"Destructor._One \n" ); }
} One;                          //! многозадачность и многооконность
void main()                     //! исполняется либо асинхронно по таймеру,
{ Window Second( "Second" ); // либо в цикле с запросом ScanKey()
          Second.MouseMove( ::MouseMove );  // OpenMP пока
          Second.MousePress( ::MousePress );// пока в отладке
          Second.SetTimer( 50,SecondTimer ); // есть «тормозок» с
             One.SetTimer( 500,OneTimer );   // идентификатором
// все это начинает работать само по себе, т. е. под контролем мышки
    while( One.Peak && (Key=WaitKey())!=_Esc )Sleep( 20 );
}
```

Window-Place – разные утилиты

```
//
//! запросы в виде наложенных на графическое поле текстовых страничек
//
byte Help( const char *N[],const char *C[],const char *P[], int X=-1,int Y=1 );
//  Name[0,1-3] Заголовок и строки расширенного названия
//  Text       Парное описание основных команд
//  Plus       Парное описание дополнительных команд
//    ++  определение каждого блока строк заканчивается нулевым адресом
//
struct Mlist{ short skip,lf; const char *Msg; void *dat; };
#define Mlist( L ) L,( sizeof( L )/sizeof( Mlist ) )
int TMenu( Mlist *M, int Nm, int x=1, int y=1, int ans=0 );
//
//     Mlist - Список параметров одного для запроса на терминал
//     skip    :пропуск строк              --> номер сроки
//     lf      : 0 - запрос не производится --> длина входного сообщения
//     Msg     :NULL - чистое входное поле  --> выходной формат
//     dat     :NULL & lf<>0 - меню-запрос  --> адрес изменяемого объекта
//
```

++1. Base~Space~Screen / 2014-12-08

- Пакет процедур визуализации графических объектов (**Basis**) в трехмерном криволинейном функциональном пространстве (**Space**) с представлением изображения в перспективных проекциях экранных сцен (**Screen**) контекстной среды программирования **OpenGL**.
- Многооконный интерфейс **Window** со стековым наложением графических и текстовых фрагментов/площадок **Place** с поддержкой клавиатуры, мышки и таймера.

Base\Space+Screen ← Window\Place

- **Volume** – программный пакет интерактивной визуализации графических объектов (**Basis**) в трехмерном криволинейном функциональном пространстве (**Space**) с представлением изображения в перспективных проекциях экранных сцен (**Screen**) в контекстной среде программирования **OpenGL**.
- ++ многооконный интерфейс (**Window**) с наложением графических и текстовых фрагментов/площадок (**Place**) с поддержкой клавиатуры, мышки и таймера.

Base\Space+Screen / формулы

Tensor – № 2013.619727, «Программа для построения числовых объектов и функций трехмерной тензорной математики при реализации вычислительных экспериментов в гидромеханике»,

Vector = Point – Point ←#→ Point = Vector + Point …

Base = Tensor + Point – *(разномасштабные сущности)*

Space = Base ~ Function(Point) – *(механика пространств)*

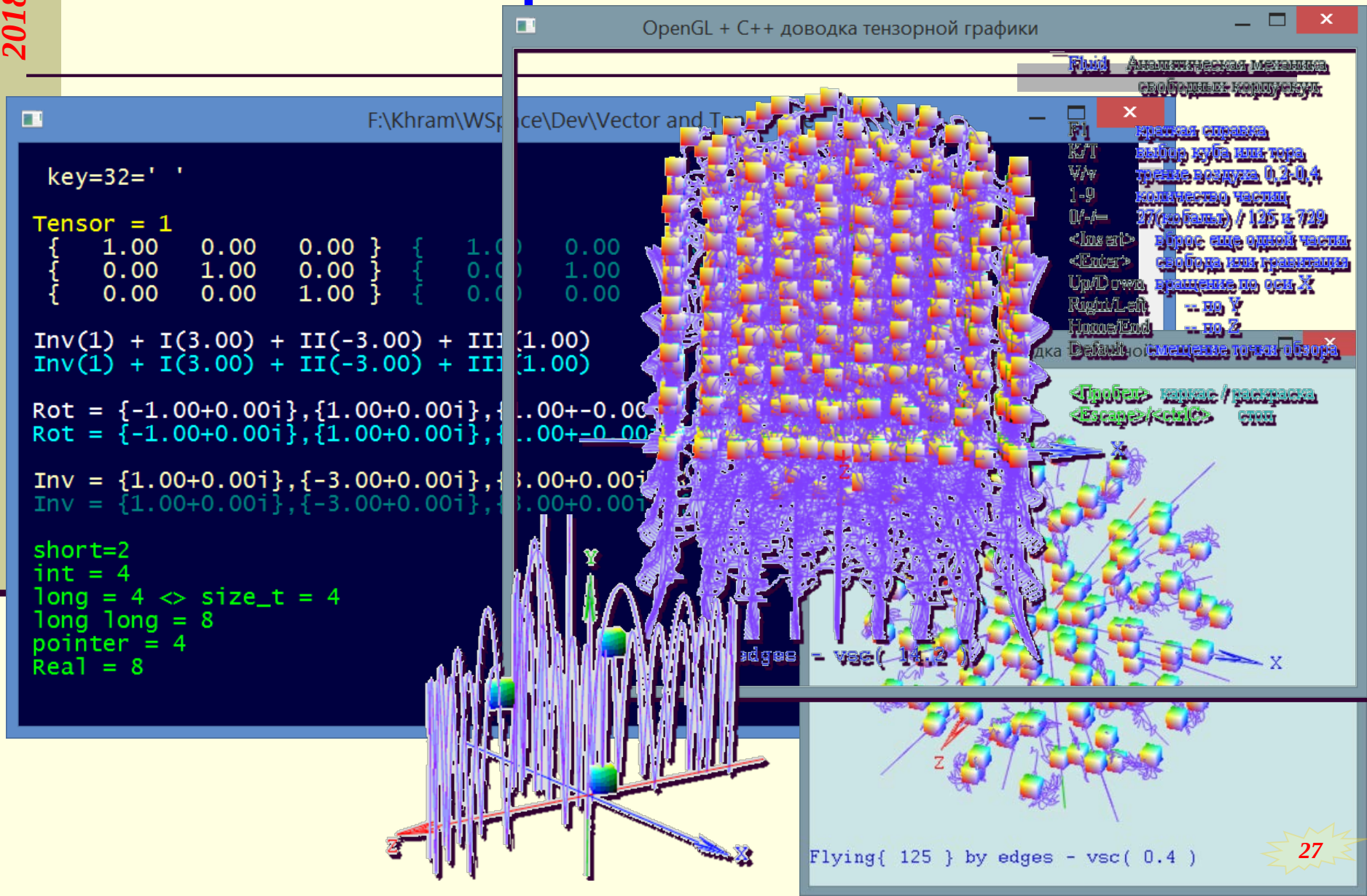
Space<<Point и Space>>Point – *(четыре варианта)*

Screen – Window/Place

«функциональная среда» – *объемная математика*

Текстовая консоль *(странное окошко)* →

Tensor → Base\Space+Screen ← Window\Place



Заключение Grid-2018

- Рассмотрен вариант построения программного комплекса на базе графической среды программирования OpenGL, окружаемой инструментальными средствами для работы со временем и интервальными таймерами, устройствами ввода информации и представления текстовых данных на предельно низком уровне прямого ввода/вывода информации и обработки прерываний в SDK Windows (… перестраивается на GLFW).

Благодарю за внимание

Морской виртуальный полигон СПбГУ:

- *О согласовании вычислительного эксперимента при интерактивном моделировании гидромеханики корабля в штормовом море*
- *Контекстная графическая среда пространственной визуализации результатов вычислительных экспериментов в механике сплошных сред*