Библиотека *Base Math Library*

Библиотека *Base Math Library* предоставляет базовые математические абстракции – *векторы*, *матрицы*, *аффинные преобразования* – и операции над ними. Библиотека имеет богатый набор поддерживаемых функций. Несмотря на то, что многие из них выполняют тривиальные операции, они носят общий характер и применяются в различных алгоритмах компьютерной графики. Больша*я* часть функций носят такие же имена, как функции из стандартных библиотек языков C/C++, но здесь они работают с векторными и матричными аргументами. Поскольку языки C/C++ поддерживают перегрузку функций, указанные функции обычно имеют несколько вариантов с одним и тем же именем, различающихся типами входных и выходных параметров.

1. Классы *Vector{2|3|4}D*

Таблица 1. Поля и методы классов ***Vector{2|3|4}D***

Открытые поля	
Синтаксис	*Описание*
`[2D|3D|4D] float X`	*Компонента x вектора*
`[2D|3D|4D] float Y`	*Компонента y вектора*
`[3D|4D] float Z`	*Компонента z вектора*
`[4D] float W`	*Компонента w вектора*
Конструкторы	
Синтаксис	*Описание*

[2D] Vector2D (float x = 0, float y = 0) [3D] Vector3D (float x = 0, float y = 0, float z = 0) [4D] Vector4D (float x = 0, float y = 0, float z = 0, float w = 0)	Создает новый вектор путем непосредственного перечисления его компонент
[2D] Vector2D (const float[2] v) [3D] Vector3D (const float[3] v) [4D] Vector4D (const float[4] v)	Создает новый вектор путем задания компонент через массив
[2D] Vector2D (const Vector2D& v) [3D] Vector3D (const Vector2D& v, float z = 0) [4D] Vector4D (const Vector2D& v, float z = 0, float w = 0)	Создает новый вектор по заданному двумерному вектору с указанием недостающих компонент
[2D] Vector2D (const Vector3D& v) [3D] Vector3D (const Vector3D& v) [4D] Vector4D (const Vector3D& v, float w = 0)	Создает новый вектор по заданному трехмерному вектору с указанием недостающих или отбрасыванием лишних компонент
[2D] Vector2D (const Vector4D& v) [3D] Vector3D (const Vector4D& v) [4D] Vector4D (const Vector4D& v)	Создает новый вектор по заданному четырехмерному вектору с отбрасыванием лишних компонент

Арифметические операторы

Синтаксис	Описание
[2D] Vector2D operator - (const Vector2D& x) [3D] Vector3D operator - (const Vector3D& x) [4D] Vector4D operator - (const Vector4D& x)	Унарный минус. Возвращает –x для каждой компоненты вектора x
[2D] Vector2D operator + (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator + (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator + (const Vector4D& x, const Vector4D& y)	Оператор сложения. Возвращает x + y для каждой пары компонент векторов x и y
[2D] Vector2D operator + (const Vector2D& x, float y) [3D] Vector3D operator + (const Vector3D& x, float y) [4D] Vector4D operator + (const Vector4D& x, float y)	Оператор сложения. Возвращает x + y для каждой компоненты вектора x и скаляра y
[2D] Vector2D operator + (float x, const Vector2D& y) [3D] Vector3D operator + (float x, const Vector3D& y) [4D] Vector4D operator + (float x, const Vector4D& y)	Оператор сложения. Возвращает x + y для каждой компоненты вектора y и скаляра x
[2D] Vector2D operator - (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator - (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator - (const Vector4D& x, const Vector4D& y)	Оператор вычитания. Возвращает x - y для каждой пары компонент векторов x и y
[2D] Vector2D operator - (const Vector2D& x, float y) [3D] Vector3D operator - (const Vector3D& x, float y) [4D] Vector4D operator - (const Vector4D& x, float y)	Оператор вычитания. Возвращает x - y для каждой компоненты вектора x и скаляра y
[2D] Vector2D operator - (float x, const Vector2D& y) [3D] Vector3D operator - (float x, const Vector3D& y) [4D] Vector4D operator - (float x, const Vector4D& y)	Оператор вычитания. Возвращает x - y для каждой компоненты вектора y и скаляра x
[2D] Vector2D operator * (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator * (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator * (const Vector4D& x, const Vector4D& y)	Оператор умножения. Возвращает x * y для каждой пары компонент векторов x и y
[2D] Vector2D operator * (const Vector2D& x, float y) [3D] Vector3D operator * (const Vector3D& x, float y) [4D] Vector4D operator * (const Vector4D& x, float y)	Оператор умножения. Возвращает x * y для каждой компоненты вектора x и скаляра y
[2D] Vector2D operator * (float x, const Vector2D& y) [3D] Vector3D operator * (float x, const Vector3D& y) [4D] Vector4D operator * (float x, const Vector4D& y)	Оператор умножения. Возвращает x * y для каждой компоненты вектора y и скаляра x
[2D] Vector2D operator / (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator / (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator / (const Vector4D& x, const Vector4D& y)	Оператор деления. Возвращает x / y для каждой пары компонент векторов x и y
[2D] Vector2D operator / (const Vector2D& x, float y)	Оператор деления. Возвращает x / y

[3D] Vector3D operator / (const Vector3D& x, float y) [4D] Vector4D operator / (const Vector4D& x, float y)	*для каждой компоненты вектора x и скаляра y*
[2D] Vector2D operator / (float x, const Vector2D& y) [3D] Vector3D operator / (float x, const Vector3D& y) [4D] Vector4D operator / (float x, const Vector4D& y)	*Оператор деления. Возвращает x / y для каждой компоненты вектора y и скаляра x*
[2D] Vector2D operator += (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator += (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator += (const Vector4D& x, const Vector4D& y)	*Присваивание со сложением. Выполняет операцию x += y для каждой пары компонент векторов x и y*
[2D] Vector2D operator -= (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator -= (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator -= (const Vector4D& x, const Vector4D& y)	*Присваивание с вычитанием. Выполняет операцию x -= y для каждой пары компонент векторов x и y*
[2D] Vector2D operator += (const Vector2D& x, float y) [3D] Vector3D operator += (const Vector3D& x, float y) [4D] Vector4D operator += (const Vector4D& x, float y)	*Присваивание со сложением. Выполняет операцию x += y для каждой компоненты вектора x и скаляра y*
[2D] Vector2D operator -= (const Vector2D& x, float y) [3D] Vector3D operator -= (const Vector3D& x, float y) [4D] Vector4D operator -= (const Vector4D& x, float y)	*Присваивание с вычитанием. Выполняет операцию x -= y для каждой компоненты вектора x и скаляра y*
[2D] Vector2D operator *= (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator *= (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator *= (const Vector4D& x, const Vector4D& y)	*Присваивание с умножением. Выполняет операцию x *= y для каждой пары компонент векторов x и y*
[2D] Vector2D operator /= (const Vector2D& x, const Vector2D& y) [3D] Vector3D operator /= (const Vector3D& x, const Vector3D& y) [4D] Vector4D operator /= (const Vector4D& x, const Vector4D& y)	*Присваивание с делением. Выполняет операцию x /= y для каждой пары компонент векторов x и y*
[2D] Vector2D operator *= (const Vector2D& x, float y) [3D] Vector3D operator *= (const Vector3D& x, float y) [4D] Vector4D operator *= (const Vector4D& x, float y)	*Присваивание с умножением. Выполняет операцию x *= y для каждой компоненты вектора x и скаляра y*
[2D] Vector2D operator /= (const Vector2D& x, float y) [3D] Vector3D operator /= (const Vector3D& x, float y) [4D] Vector4D operator /= (const Vector4D& x, float y)	*Присваивание с делением. Выполняет операцию x /= y для каждой компоненты вектора x и скаляра y*

Логические операторы

Синтаксис	*Описание*
[2D] bool operator < (const Vector2D& x, const Vector2D& y) [3D] bool operator < (const Vector3D& x, const Vector3D& y) [4D] bool operator < (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если неравенство x < y выполняется для каждой пары компонент векторов x и y*
[2D] bool operator > (const Vector2D& x, const Vector2D& y) [3D] bool operator > (const Vector3D& x, const Vector3D& y) [4D] bool operator > (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если неравенство x > y выполняется для каждой пары компонент векторов x и y*
[2D] bool operator < (const Vector2D& x, float y) [3D] bool operator < (const Vector3D& x, float y) [4D] bool operator < (const Vector4D& x, float y)	*Возвращает **true**, если неравенство x < y выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator > (const Vector2D& x, float y) [3D] bool operator > (const Vector3D& x, float y) [4D] bool operator > (const Vector4D& x, float y)	*Возвращает **true**, если неравенство x > y выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator < (float x, const Vector2D& y) [3D] bool operator < (float x, const Vector3D& y) [4D] bool operator < (float x, const Vector4D& y)	*Возвращает **true**, если неравенство x < y выполняется для каждой компоненты вектора y и скаляра x*
[2D] bool operator > (float x, const Vector2D& y) [3D] bool operator > (float x, const Vector3D& y) [4D] bool operator > (float x, const Vector4D& y)	*Возвращает **true**, если неравенство x > y выполняется для каждой компоненты вектора y и скаляра x*
[2D] bool operator <= (const Vector2D& x, const Vector2D& y) [3D] bool operator <= (const Vector3D& x, const Vector3D& y) [4D] bool operator <= (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если неравенство x ≤ y выполняется для каждой пары компонент векторов x и y*

[2D] bool operator >= (const Vector2D& x, const Vector2D& y) [3D] bool operator >= (const Vector3D& x, const Vector3D& y) [4D] bool operator >= (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если неравенство $x \geq y$ выполняется для каждой пары компонент векторов x и y*
[2D] bool operator <= (const Vector2D& x, float y) [3D] bool operator <= (const Vector3D& x, float y) [4D] bool operator <= (const Vector4D& x, float y)	*Возвращает **true**, если неравенство $x \leq y$ выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator >= (const Vector2D& x, float y) [3D] bool operator >= (const Vector3D& x, float y) [4D] bool operator >= (const Vector4D& x, float y)	*Возвращает **true**, если неравенство $x \geq y$ выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator <= (float x, const Vector2D& y) [3D] bool operator <= (float x, const Vector3D& y) [4D] bool operator <= (float x, const Vector4D& y)	*Возвращает **true**, если неравенство $x \leq y$ выполняется для каждой компоненты вектора y и скаляра x*
[2D] bool operator >= (float x, const Vector2D& y) [3D] bool operator >= (float x, const Vector3D& y) [4D] bool operator >= (float x, const Vector4D& y)	*Возвращает **true**, если неравенство $x \geq y$ выполняется для каждой компоненты вектора y и скаляра x*
[2D] bool operator == (const Vector2D& x, const Vector2D& y) [3D] bool operator == (const Vector3D& x, const Vector3D& y) [4D] bool operator == (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если равенство x = y выполняется для каждой пары компонент векторов x и y*
[2D] bool operator != (const Vector2D& x, const Vector2D& y) [3D] bool operator != (const Vector3D& x, const Vector3D& y) [4D] bool operator != (const Vector4D& x, const Vector4D& y)	*Возвращает **true**, если неравенство $x \neq y$ выполняется для каждой пары компонент векторов x и y*
[2D] bool operator == (const Vector2D& x, float y) [3D] bool operator == (const Vector3D& x, float y) [4D] bool operator == (const Vector4D& x, float y)	*Возвращает **true**, если равенство x = y выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator != (const Vector2D& x, float y) [3D] bool operator != (const Vector3D& x, float y) [4D] bool operator != (const Vector4D& x, float y)	*Возвращает **true**, если неравенство $x \neq y$ выполняется для каждой компоненты вектора x и скаляра y*
[2D] bool operator == (float x, const Vector2D& y) [3D] bool operator == (float x, const Vector3D& y) [4D] bool operator == (float x, const Vector4D& y)	*Возвращает **true**, если равенство x = y выполняется для каждой компоненты вектора y и скаляра x*
[2D] bool operator != (float x, const Vector2D& y) [3D] bool operator != (float x, const Vector3D& y) [4D] bool operator != (float x, const Vector4D& y)	*Возвращает **true**, если неравенство $x \neq y$ выполняется для каждой компоненты вектора y и скаляра x*
Операторы ввода/вывода	
Синтаксис	**Описание**
[2D] istream& operator >> (istream&, Vector2D& x) [3D] istream& operator >> (istream&, Vector3D& x) [4D] istream& operator >> (istream&, Vector4D& x)	*Выполняет ввод вектора x из стандартного потока ввода. Компоненты вектора могут быть произвольными числами, разделенными символом пробела*
[2D] ostream& operator << (ostream&, const Vector2D& x) [3D] ostream& operator << (ostream&, const Vector3D& x) [4D] ostream& operator << (ostream&, const Vector4D& x)	*Выполняет вывод вектора x в стандартный поток вывода. Компоненты вектора разделяются символом пробела*
Дополнительные операторы	
Синтаксис	**Описание**
[2D|3D|4D] operator float * ( )	*Преобразование вектора к массиву вещественных чисел с плавающей точкой*
[2D|3D|4D] operator const float * ( ) const	*Преобразование вектора к*

	константному массиву вещественных чисел с плавающей точкой
`[2D|3D|4D] float& operator [] ( int index )`	*Обеспечивает доступ к компонентам вектора в стиле массива*
Функции общего назначения	
Синтаксис	*Описание*

`[2D] Vector2D Abs ( const Vector2D& x )`
`[3D] Vector3D Abs ( const Vector3D& x )`
`[4D] Vector4D Abs ( const Vector4D& x )`
	Возвращает x, если $x \geq 0$; в противном случае возвращает –x для каждой компоненты вектора x

`[2D] Vector2D Sign ( const Vector2D& x )`
`[3D] Vector3D Sign ( const Vector3D& x )`
`[4D] Vector4D Sign ( const Vector4D& x )`
	Возвращает 1.0, если $x > 0$, 0.0, если $x = 0$ и -1.0, если $x < 0$ для каждой компоненты вектора x

`[2D] Vector2D Floor ( const Vector2D& x )`
`[3D] Vector3D Floor ( const Vector3D& x )`
`[4D] Vector4D Floor ( const Vector4D& x )`
	Возвращает ближайшее целое число, меньшее или равное x для каждой компоненты вектора x

`[2D] Vector2D Ceil ( const Vector2D& x )`
`[3D] Vector3D Ceil ( const Vector3D& x )`
`[4D] Vector4D Ceil ( const Vector4D& x )`
	Возвращает ближайшее целое число, большее или равное x для каждой компоненты вектора x

`[2D] Vector2D Fract ( const Vector2D& x )`
`[3D] Vector3D Fract ( const Vector3D& x )`
`[4D] Vector4D Fract ( const Vector4D& x )`
	Возвращает $x - \mathbf{floor}(x)$ для каждой компоненты вектора x

`[2D] Vector2D Mod ( const Vector2D& x, const Vector2D& y )`
`[3D] Vector3D Mod ( const Vector3D& x, const Vector3D& y )`
`[4D] Vector4D Mod ( const Vector4D& x, const Vector4D& y )`
	Возвращает $x - y \cdot \mathbf{floor}(x / y)$ для каждой пары компонент векторов x и y

`[2D] Vector2D Mod ( const Vector2D& x, float y )`
`[3D] Vector3D Mod ( const Vector3D& x, float y )`
`[4D] Vector4D Mod ( const Vector4D& x, float y )`
	Возвращает $x - y \cdot \mathbf{floor}(x / y)$ для каждой компоненты вектора x и скаляра y

`[2D] Vector2D Mod ( float x, const Vector2D& y )`
`[3D] Vector3D Mod ( float x, const Vector3D& y )`
`[4D] Vector4D Mod ( float x, const Vector4D& y )`
	Возвращает $x - y \cdot \mathbf{floor}(x / y)$ для каждой компоненты вектора y и скаляра x

`[2D] Vector2D Min ( const Vector2D& x, const Vector2D& y )`
`[3D] Vector3D Min ( const Vector3D& x, const Vector3D& y )`
`[4D] Vector4D Min ( const Vector4D& x, const Vector4D& y )`
	Возвращает y, если $y < x$; в противном случае возвращает x для каждой пары компонент векторов x и y

`[2D] Vector2D Min ( const Vector2D& x, float y )`
`[3D] Vector3D Min ( const Vector3D& x, float y )`
`[4D] Vector4D Min ( const Vector4D& x, float y )`
	Возвращает y, если $y < x$; в противном случае возвращает x для каждой компоненты вектора x и скаляра y

`[2D] Vector2D Min ( float x, const Vector2D& y )`
`[3D] Vector3D Min ( float x, const Vector3D& y )`
`[4D] Vector4D Min ( float x, const Vector4D& y )`
	Возвращает y, если $y < x$; в противном случае возвращает x для каждой компоненты вектора y и скаляра x

`[2D] Vector2D Max ( const Vector2D& x, const Vector2D& y )`
`[3D] Vector3D Max ( const Vector3D& x, const Vector3D& y )`
`[4D] Vector4D Max ( const Vector4D& x, const Vector4D& y )`
	Возвращает y, если $y > x$; в противном случае возвращает x для каждой пары компонент векторов x и y

`[2D] Vector2D Max ( const Vector2D& x, float y )`
`[3D] Vector3D Max ( const Vector3D& x, float y )`
`[4D] Vector4D Max ( const Vector4D& x, float y )`
	Возвращает y, если $y > x$; в противном случае возвращает x для каждой компоненты вектора x и скаляра y

`[2D] Vector2D Max ( float x, const Vector2D& y )`
`[3D] Vector3D Max ( float x, const Vector3D& y )`
`[4D] Vector4D Max ( float x, const Vector4D& y )`
	Возвращает y, если $y > x$; в противном случае возвращает x для каждой компоненты вектора y и скаляра x

`[2D] Vector2D Clamp ( const Vector2D& x,`
`  const Vector2D& a, const Vector2D& b )`
`[3D] Vector3D Clamp ( const Vector3D& x,`
`  const Vector3D& a, const Vector3D& b )`
`[4D] Vector4D Clamp ( const Vector4D& x,`
	Возвращает min (max (x, a), b) для каждой группы компонент векторов x, a и b

const Vector4D& a, const Vector4D& b)	
[2D] Vector2D Clamp (const Vector2D& x, float a, float b) [3D] Vector3D Clamp (const Vector3D& x, float a, float b) [4D] Vector4D Clamp (const Vector4D& x, float a, float b)	Возвращает min (max (x, a), b) для каждой компоненты вектора x и скаляров a и b
[2D] Vector2D Mix (const Vector2D& x, const Vector2D& y, const Vector2D& a) [3D] Vector3D Mix (const Vector3D& x, const Vector3D& y, const Vector3D& a) [4D] Vector4D Mix (const Vector4D& x, const Vector4D& y, const Vector4D& a)	Возвращает x · (1 - a) + y · a для каждой группы компонент векторов x, y и a
[2D] Vector2D Mix (const Vector2D& x, const Vector2D& y, float a) [3D] Vector3D Mix (const Vector3D& x, const Vector3D& y, float a) [4D] Vector4D Mix (const Vector4D& x, const Vector4D& y, float a)	Возвращает x · (1 - a) + y · a для каждой пары компонент векторов x, y и скаляра a
[2D] Vector2D Step (const Vector2D& x, const Vector2D& a) [3D] Vector3D Step (const Vector3D& x, const Vector3D& a) [4D] Vector4D Step (const Vector4D& x, const Vector4D& a)	Возвращает 0.0, если $x \le a$; в противном случае возвращает 1.0 для каждой пары компонент векторов x и a
[2D] Vector2D Step (const Vector2D& x, float a) [3D] Vector3D Step (const Vector3D& x, float a) [4D] Vector4D Step (const Vector4D& x, float a)	Возвращает 0.0, если $x \le a$; в противном случае возвращает 1.0 для каждой компоненты вектора x и скаляра a
[2D] Vector2D Smooth (const Vector2D& x, const Vector2D& a, const Vector2D& b) [3D] Vector3D Smooth (const Vector3D& x, const Vector3D& a, const Vector3D& b) [4D] Vector4D Smooth (const Vector4D& x, const Vector4D& a, const Vector4D& b)	Возвращает 0.0, если $x \le a$, 1.0, если $x \ge b$ и выполняет плавную интерполяцию Хермита между 0.0 и 1.0, если $a < x < b$ для каждой группы компонент векторов x, y и a
[2D] Vector2D Smooth (const Vector2D& x, float a, float b) [3D] Vector3D Smooth (const Vector3D& x, float a, float b) [4D] Vector4D Smooth (const Vector4D& x, float a, float b)	Возвращает 0.0, если $x \le a$, 1.0, если $x \ge b$ и выполняет плавную интерполяцию Хермита между 0.0 и 1.0, если $a < x < b$ для каждой компоненты вектора x и скаляров a и b

Геометрические функции

Синтаксис	Описание
[2D] float Length (const Vector2D& x) [3D] float Length (const Vector3D& x) [4D] float Length (const Vector4D& x)	Возвращает длину вектора $\|x\|$
[2D] float Square (const Vector2D& x) [3D] float Square (const Vector3D& x) [4D] float Square (const Vector4D& x)	Возвращает квадрат длины вектора $\|x\|^2$
[2D] float Distance (const Vector2D& x, const Vector2D& y) [3D] float Distance (const Vector3D& x, const Vector3D& y) [4D] float Distance (const Vector4D& x, const Vector4D& y)	Возвращает расстояние между двумя точками $\|x - y\|$
[2D] float Dot (const Vector2D& x, const Vector2D& y) [3D] float Dot (const Vector3D& x, const Vector3D& y) [4D] float Dot (const Vector4D& x, const Vector4D& y)	Возвращает скалярное произведение векторов x и y
[3D] Vector3D Cross (const Vector3D& x, const Vector3D& y)	Возвращает векторное произведение векторов x и y
[2D] Vector2D Normalize (const Vector2D& x) [3D] Vector3D Normalize (const Vector3D& x) [4D] Vector4D Normalize (const Vector4D& x)	Возвращает вектор с тем же направлением, что x, но с единичной длиной
[2D] Vector2D Reflect (const Vector2D& i, const Vector2D& n) [3D] Vector3D Reflect (const Vector3D& i, const Vector3D& n) [4D] Vector4D Reflect (const Vector4D& i, const Vector4D& n)	Для падающего вектора i и ориентации поверхности n возвращает направление отражения

[2D] Vector2D Refract (const Vector2D& i, const Vector2D& n, float index) [3D] Vector3D Refract (const Vector3D& i, const Vector3D& n, float index) [4D] Vector4D Refract (const Vector4D& i, const Vector4D& n, float index)	*Для падающего вектора i, ориентации поверхности n и относительного коэффициента преломления index возвращает направление преломления*
Угловые и тригонометрические функции	
Синтаксис	*Описание*
[2D] Vector2D Radians (const Vector2D& x) [3D] Vector3D Radians (const Vector3D& x) [4D] Vector4D Radians (const Vector4D& x)	*Для каждой компоненты вектора x переводит градусы в радианы и возвращает результат*
[2D] Vector2D Degrees (const Vector2D& x) [3D] Vector3D Degrees (const Vector3D& x) [4D] Vector4D Degrees (const Vector4D& x)	*Для каждой компоненты вектора x переводит радианы в градусы и возвращает результат*
[2D] Vector2D Sin (const Vector2D& x) [3D] Vector3D Sin (const Vector3D& x) [4D] Vector4D Sin (const Vector4D& x)	*Возвращает sin (x) для каждой компоненты вектора x*
[2D] Vector2D Cos (const Vector2D& x) [3D] Vector3D Cos (const Vector3D& x) [4D] Vector4D Cos (const Vector4D& x)	*Возвращает cos (x) для каждой компоненты вектора x*
[2D] Vector2D Tan (const Vector2D& x) [3D] Vector3D Tan (const Vector3D& x) [4D] Vector4D Tan (const Vector4D& x)	*Возвращает tan (x) для каждой компоненты вектора x*
[2D] Vector2D Asin (const Vector2D& x) [3D] Vector3D Asin (const Vector3D& x) [4D] Vector4D Asin (const Vector4D& x)	*Возвращает arcsin (x) для каждой компоненты вектора x*
[2D] Vector2D Acos (const Vector2D& x) [3D] Vector3D Acos (const Vector3D& x) [4D] Vector4D Acos (const Vector4D& x)	*Возвращает arccos (x) для каждой компоненты вектора x*
[2D] Vector2D Atan (const Vector2D& x) [3D] Vector3D Atan (const Vector3D& x) [4D] Vector4D Atan (const Vector4D& x)	*Возвращает arctan (x) для каждой компоненты вектора x*
Экспоненциальные и логарифмические функции	
Синтаксис	*Описание*
[2D] Vector2D Pow (const Vector2D& x, const Vector2D& y) [3D] Vector3D Pow (const Vector3D& x, const Vector3D& y) [4D] Vector4D Pow (const Vector4D& x, const Vector4D& y)	*Возвращает x^y для каждой пары компонент векторов x и y*
[2D] Vector2D Pow (const Vector2D& x, float y) [3D] Vector3D Pow (const Vector3D& x, float y) [4D] Vector4D Pow (const Vector4D& x, float y)	*Возвращает x^y для каждой компоненты вектора x и скаляра y*
[2D] Vector2D Exp (const Vector2D& x) [3D] Vector3D Exp (const Vector3D& x) [4D] Vector4D Exp (const Vector4D& x)	*Возвращает e^x для каждой компоненты вектора x*
[2D] Vector2D Log (const Vector2D& x) [3D] Vector3D Log (const Vector3D& x) [4D] Vector4D Log (const Vector4D& x)	*Возвращает ln x для каждой компоненты вектора x*
[2D] Vector2D Sqrt (const Vector2D& x) [3D] Vector3D Sqrt (const Vector3D& x) [4D] Vector4D Sqrt (const Vector4D& x)	*Возвращает для каждой компоненты вектора x*

2. Классы `Matrix{2|3|4}D`

Таблица 2. Поля и методы классов `Matrix{2|3|4}D`

Конструкторы	
Синтаксис	Описание
[2D] Matrix2D (float d = 0) [3D] Matrix3D (float d = 0) [4D] Matrix4D (float d = 0)	Создает новую матрицу с одинаковыми элементами на главной диагонали
[2D] Matrix2D (const float[2] d) [3D] Matrix3D (const float[3] d) [4D] Matrix4D (const float[4] d)	Создает новую матрицу с заданным массивом элементов на главной диагонали
[2D] Matrix2D (const Vector2D& d) [3D] Matrix3D (const Vector3D& d) [4D] Matrix4D (const Vector4D& d)	Создает новую матрицу с заданным вектором элементов на главной диагонали
[2D] Matrix2D (const float[2][2] m) [3D] Matrix3D (const float[3][3] m) [4D] Matrix4D (const float[4][4] m)	Создает новую матрицу путем задания всех ее элементов в виде массива
[2D] Matrix2D (const Matrix2D& m) [3D] Matrix3D (const Matrix3D& m) [4D] Matrix4D (const Matrix4D& m)	Создает новую матрицу как точную копию заданной
Арифметические операторы	
Синтаксис	Описание
[2D] Matrix2D operator - (const Matrix2D& x) [3D] Matrix3D operator - (const Matrix3D& x) [4D] Matrix4D operator - (const Matrix4D& x)	Унарный минус. Возвращает –x для каждого элемента матрицы x
[2D] Matrix2D operator + (const Matrix2D& x, const Matrix2D& y) [3D] Matrix3D operator + (const Matrix3D& x, const Matrix3D& y) [4D] Matrix4D operator + (const Matrix4D& x, const Matrix4D& y)	Оператор сложения. Возвращает $x_{ij} + y_{ij}$ для каждой пары элементов матриц x и y
[2D] Matrix2D operator - (const Matrix2D& x, const Matrix2D& y) [3D] Matrix3D operator - (const Matrix3D& x, const Matrix3D& y) [4D] Matrix4D operator - (const Matrix4D& x, const Matrix4D& y)	Оператор вычитания. Возвращает $x_{ij} - y_{ij}$ для каждой пары элементов матриц x и y
[2D] Matrix2D operator * (const Matrix2D& x, const Matrix2D& y) [3D] Matrix3D operator * (const Matrix3D& x, const Matrix3D& y) [4D] Matrix4D operator * (const Matrix4D& x, const Matrix4D& y)	Оператор вычитания. Возвращает результат ***алгебраического*** умножения матриц x и y
[2D] Matrix2D operator * (const Matrix2D& x, const Vector2D& y) [3D] Matrix3D operator * (const Matrix3D& x, const Vector3D& y) [4D] Matrix4D operator * (const Matrix4D& x, const Vector4D& y)	Оператор вычитания. Возвращает результат ***алгебраического*** умножения матрицы x на вектор y
[2D] Matrix2D operator * (const Matrix2D& x, float y) [3D] Matrix3D operator * (const Matrix3D& x, float y) [4D] Matrix4D operator * (const Matrix4D& x, float y)	Оператор вычитания. Возвращает $x_{ij} \cdot y$ для каждого элемента матрицы x и скаляра y
[2D] Matrix2D operator / (const Matrix2D& x, float y) [3D] Matrix3D operator / (const Matrix3D& x, float y) [4D] Matrix4D operator / (const Matrix4D& x, float y)	Оператор вычитания. Возвращает x_{ij} / y для каждого элемента матрицы x и скаляра y
[2D] Matrix2D operator * (float x, const Matrix2D& y) [3D] Matrix3D operator * (float x, const Matrix3D& y) [4D] Matrix4D operator * (float x, const Matrix4D& y)	Оператор вычитания. Возвращает $x \cdot y_{ij}$ для каждого элемента матрицы y и скаляра x
[2D] Matrix2D operator / (float x, const Matrix2D& y) [3D] Matrix3D operator / (float x, const Matrix3D& y) [4D] Matrix4D operator / (float x, const Matrix4D& y)	Оператор вычитания. Возвращает x / y_{ij} для каждого элемента матрицы y и скаляра x
Операторы ввода/вывода	
Синтаксис	Описание

[2D] istream& operator >> (istream&, Matrix2D & x) [3D] istream& operator >> (istream&, Matrix3D & x) [4D] istream& operator >> (istream&, Matrix4D & x)	*Выполняет ввод матрицы x **по строкам** из стандартного потока ввода. Элементы матрицы могут быть произвольными числами, разделенными символом пробела*
[2D] ostream& operator << (ostream&, const Matrix2D & x) [3D] ostream& operator << (ostream&, const Matrix3D & x) [4D] ostream& operator << (ostream&, const Matrix4D & x)	*Выполняет вывод матрицы x **по строкам** в стандартный поток вывода. Элементы матрицы разделяются символом пробела*

Дополнительные операторы

Синтаксис	*Описание*
[2D|3D|4D] float * operator [] (int index)	*Обеспечивает доступ к элементам матрицы как к двумерному массиву*
[2D|3D|4D] const float * operator [] (int index) const	*Обеспечивает доступ к элементам матрицы как к константному двумерному массиву*

Функции общего назначения

Синтаксис	*Описание*
[2D] float Determinant (const Matrix2D& x) [3D] float Determinant (const Matrix3D& x) [4D] float Determinant (const Matrix4D& x)	*Возвращает определитель матрицы x*
[2D] Matrix2D Transpose (const Matrix2D& x) [3D] Matrix3D Transpose (const Matrix3D& x) [4D] Matrix4D Transpose (const Matrix4D& x)	*Возвращает результат транспонирования матрицы x*
[2D] Matrix2D Adjugate (const Matrix2D& x) [3D] Matrix3D Adjugate (const Matrix3D& x) [4D] Matrix4D Adjugate (const Matrix4D& x)	*Возвращает присоединенную матрицу для матрицы x*
[2D] Matrix2D Inverse (const Matrix2D& x) [3D] Matrix3D Inverse (const Matrix3D& x) [4D] Matrix4D Inverse (const Matrix4D& x)	*Возвращает обратную матрицу для матрицы x*

Функции общего назначения

Синтаксис	*Описание*
[3D] Matrix3D MirrorX (void)	*Возвращает матрицу отражения относительно плоскости YZ*
[3D] Matrix3D MirrorY (void)	*Возвращает матрицу отражения относительно плоскости XZ*
[3D] Matrix3D MirrorZ (void)	*Возвращает матрицу отражения относительно плоскости XY*
[3D] Matrix3D RotateX (float a)	*Возвращает матрицу поворота вокруг оси X на угол a (в радианах)*
[3D] Matrix3D RotateY (float a)	*Возвращает матрицу поворота вокруг оси Y на угол a (в радианах)*
[3D] Matrix3D RotateZ (float a)	*Возвращает матрицу поворота вокруг оси Z на угол a (в радианах)*
[3D] Matrix3D Rotate (float a, const Vector3D& v)	*Возвращает матрицу поворота вокруг заданного направления v на угол a (в радианах)*
[3D] Matrix3D Rotate (const Vector3D& a)	*Возвращает матрицу поворота вокруг осей системы координат на заданные*

	вектором a углы (в радианах)
`[3D] Matrix3D ScaleX ( float s )`	*Возвращает матрицу масштабирования вдоль оси X с коэффициентом s*
`[3D] Matrix3D ScaleY ( float s )`	*Возвращает матрицу масштабирования вдоль оси Y с коэффициентом s*
`[3D] Matrix3D ScaleZ ( float s )`	*Возвращает матрицу масштабирования вдоль оси Z с коэффициентом s*
`[3D] Matrix3D Scale ( const Vector3D& s )`	*Возвращает матрицу масштабирования вдоль осей системы координат с заданными вектором s коэффициентами*

3. Класс *Transform*

Таблица 3. Поля и методы класса ***Transform***

Конструктор	
Синтаксис	*Описание*

`Transform ( const Vector3D& t = Vector3D :: Zero,`
`const Vector3D& o = Vector3D :: Zero,`
`const Vector3D& s = Vector3D :: Unit )`
	Создает новое аффинное преобразование для заданного вектора параллельного переноса t, углов поворота o и коэффициентов масштабирования s: $F(p) = R \cdot S \cdot p + T,\ p \in R^3,$ *где S – матрица масштабирования, R – матрица поворота, T – вектор параллельного переноса*

Функции преобразования координат	
Синтаксис	*Описание*
`Vector3D ForwardPoint ( const Vector3D& p )`	*Возвращает результат преобразования* ***точки*** *p:* $F(p) = R \cdot S \cdot p + T$
`Vector3D ForwardVector ( const Vector3D& d )`	*Возвращает результат преобразования* ***направления*** *d:* $F(p) = R \cdot S \cdot p$
`Vector3D ForwardNormal ( const Vector3D& n )`	*Возвращает результат преобразования* ***нормали*** *n:* $F(p) = R \cdot S^{-1} \cdot p$
`Vector3D BackwardPoint ( const Vector3D& p )`	*Возвращает результат* ***обратного*** *преобразования* ***точки*** *p:* $F^{-1}(p) = S^{-1} \cdot R^{-1} \cdot (p - T)$
`Vector3D BackwardVector ( const Vector3D& d )`	*Возвращает результат* ***обратного*** *преобразования* ***направления*** *d:* $F^{-1}(p) = S^{-1} \cdot R^{-1} \cdot d$
Функции чтения / установки параметров преобразования	

Синтаксис	Описание
`Vector3D GetTranslation ( void )`	Возвращает вектор параллельного переноса
`Vector3D GetOrientation ( void )`	Возвращает вектор с углами поворота вокруг осей системы координат
`Vector3D GetScale ( void )`	Возвращает вектор с коэффициентами масштабирования вдоль осей системы координат
`void SetTranslation ( const Vector3D& t )`	Устанавливает новый вектор параллельного переноса
`void SetOrientation ( const Vector3D& o )`	Устанавливает новые углы поворота вокруг осей системы координат
`void SetScale ( const Vector3D& s )`	Устанавливает новые коэффициенты масштабирования вдоль осей системы координат