PBMath

P. Baillehache

October 30, 2018

Contents

Introduction

PBMath is a C library providing mathematical structures and functions.

The `VecFloat` structure and its functions can be used to manipulate vectors of float values.

The `VecShort` structure and its functions can be used to manipulate vectors of short values.

The `MatFloat` structure and its functions can be used to manipulate matrices of float values.

The `Gauss` structure and its functions can be used to get values of the Gauss function and random values distributed accordingly with a Gauss distribution.

The `Smoother` functions can be used to get values of the SmoothStep and SmootherStep functions.

The `EqLinSys` structure and its functions can be used to solve systems of linear equation.

It uses the `PBErr` library.

1 Definitions

1.1 Vector

1.1.1 Distance between two vectors

For `VecShort`:

$$\begin{array}{l} Dist(\overrightarrow{v},\overrightarrow{w}) = \sum_i |v_i - w_i| \\ HamiltonDist(\overrightarrow{v},\overrightarrow{w}) = \sum_i |v_i - w_i| \\ PixelDist(\overrightarrow{v},\overrightarrow{w}) = \sum_i |v_i - w_i| \end{array} \tag{1}$$

For `VecFloat`:

$$\begin{array}{l} Dist(\overrightarrow{v},\overrightarrow{w}) = \sum_i (v_i - w_i)^2 \\ HamiltonDist(\overrightarrow{v},\overrightarrow{w}) = \sum_i |v_i - w_i| \\ PixelDist(\overrightarrow{v},\overrightarrow{w}) = \sum_i |\lfloor v_i \rfloor - \lfloor w_i \rfloor| \end{array} \tag{2}$$

1.1.2 Angle between two vectors

The problem is as follow: given two vectors $\vec{V}$ and $\vec{W}$ not null, how to calculate the angle θ from $\vec{V}$ to $\vec{W}$.

Let's call M the rotation matrix: $M\vec{V} = \vec{W}$, and the components of M as follow:

$$M = \begin{bmatrix} Ma & Mb \\ Mc & Md \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{3}$$

Then, $M\vec{V} = \vec{W}$ can be written has

$$\begin{cases} W_x = M_a V_x + M_b V_y \\ W_y = M_c V_x + M_d V_y \end{cases} \tag{4}$$

Equivalent to

$$\begin{cases} W_x = M_a V_x + M_b V_y \\ W_y = -M_b V_x + M_a V_y \end{cases} \tag{5}$$

where $M_a = cos(\theta)$ and $M_b = -sin(\theta)$.
If $Vx \neq 0.0$, we can write

$$\begin{cases} M_b = \frac{M_a V_y - W_y}{V_x} \\ M_a = \frac{W_x + W_y V_y / V_x}{V_x + V_y^2 / V_x} \end{cases} \tag{6}$$

Or, if $Vx = 0.0$, we can write

$$\begin{cases} Ma = \frac{W_y + M_b V_x}{V_y} \\ Mb = \frac{W_x - W_y V_x / V_y}{V_y + V_x^2 / V_y} \end{cases} \tag{7}$$

Then we have $\theta = \pm cos^{-1}(M_a)$ where the sign can be determined by verifying that the sign of $sin(\theta)$ matches the sign of $-M_b$: if $sin(cos^{-1}(M_a)) * M_b > 0.0$ then multiply $\theta = -cos^{-1}(M_a)$ else $\theta = cos^{-1}(M_a)$.

1.1.3 Rotation

Rotation if a vector is only defined in 2D and 3D. In 2D, for a right-handed rotation of angle θ the rotation matrix is equal to:

$$R = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{8}$$

In 3D, for a right-handed rotation of angle θ around axis $\overrightarrow{u}$ the rotation is equal to (to shorten notation θ is not written in the matrix below):

$$R = \begin{bmatrix} cos + u_x^2(1-cos) & u_xu_y(1-cos) - u_zsin & u_xu_z(1-cos) + u_ysin \\ u_xu_y(1-cos) + u_zsin & cos + u_y^2(1-cos) & u_yu_z(1-cos) - u_xsin \\ u_xu_z(1-cos) - u_ysin & u_yu_z(1-cos) + u_xsin & cos + u_z^2(1-cos) \end{bmatrix} \tag{9}$$

1.2 Matrix

1.2.1 Inverse matrix

The inverse of a matrix is only implemented for square matrices less than 3x3. It is computed directly, based on the determinant and the adjoint matrix.

For a 2x2 matrix M:

$$M^{-1} = \frac{1}{det} \begin{bmatrix} M_3 & -M_2 \\ -M_1 & M_0 \end{bmatrix} \tag{10}$$

where

$$M = \begin{bmatrix} M_0 & M_2 \\ M_1 & M_3 \end{bmatrix} \tag{11}$$

and

$$det = M_0M_3 - M_1M_2 \tag{12}$$

For a 3x3 matrix M:

$$M^{-1} = \frac{1}{det} \begin{bmatrix} (M_4M_8 - M_5M_7) & -(M_3M_8 - M_5M_6) & (M_3M_7 - M_4M_6) \\ -(M_1M_8 - M_2M_7) & (M_0M_8 - M_2M_6) & -(M_0M_7 - M_1M_6) \\ (M_1M_5 - M_2M_4) & -(M_0M_5 - M_2M_3) & (M_0M_4 - M_1M_3) \end{bmatrix} \tag{13}$$

where

$$M = \begin{bmatrix} M_0 & M_3 & M_6 \\ M_1 & M_4 & M_7 \\ M_2 & M_5 & M_8 \end{bmatrix} \tag{14}$$

and

$$\begin{aligned} det = \; & M_0(M_4M_8 - M_5M_7) - \\ & M_3(M_1M_8 - M_2M_7) + \\ & M_6(M_1M_5 - M_2M_4) \end{aligned} \tag{15}$$

2 Interface

3 Code

3.1 pbmath.c

3.2 pbmath-inline.c

4 Makefile

5 Unit tests

6 Unit tests output

7 Examples

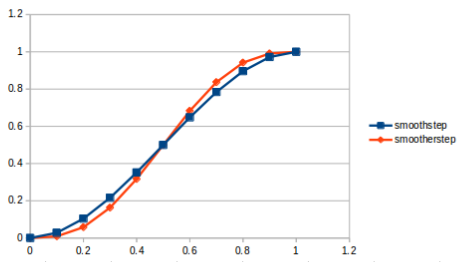
UnitTestVecShortLoadSave.txt:

UnitTestVecLongLoadSave.txt:

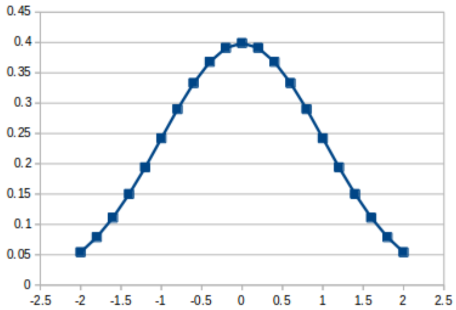
UnitTestVecFloatLoadSave.txt:

matfloat.txt:

smoother functions:



gauss function (mean:0.0, sigma:1.0):



gauss rand function (mean:1.0, sigma:0.5):