Body Tracker

1.0.0

Generated by Doxygen 1.8.8

Mon Mar 16 2015 16:16:06

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Matriz Class Reference

Classe que representa uma matriz e suas operações.

`#include <matriz.h>`

Public Member Functions

- Matriz ()

  Construtor padrão.
- Matriz (int linha, int coluna)

  Contrutor utilizado para inicializar uma matriz com suas dimensões.
- Matriz * Multiplicar (Matriz *m)

  Método utilizado para multiplica duas matrizes.
- void MultiplicaTransformacao (Matriz *m)

  Semelhante ao método Multiplicar.
- Matriz * Somar (Matriz *m)

  Somar duas matrizes.
- Matriz * Transposta ()

  Calculo da transposta da matriz atual.
- Matriz * Subtrair (Matriz *m)

  Subtrair duas matrizes.
- void CarregarIdentidade ()

  Preencher a matriz atual com a matriz identidade.
- float RetornaValor (int indice)

  Buscar valor no vetor que representa a matriz.
- void AlterarValor (int i, double valor)

  Alterar valor no vetor que representa a matriz.
- void AlterarValor (int x, int y, double valor)

  Alterar valor utilizando coordenadas da matriz.
- float BuscarValor (int x, int y)

  Buscar valor utilizando coordenadas da matriz.
- int GetLinha ()

  Método para acessar a quantidade de linhas de uma matriz.
- int GetColuna ()

  Método para acessar a quantidade de colunas de uma matriz.
- GLdouble * MatrixToGLDArray ()

  Converter matriz atual um array.

Public Attributes

- std::vector< GLdouble > _m

  Vetor que representa a matriz.

2.1.1 Detailed Description

Classe que representa uma matriz e suas operações.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Matriz::Matriz (int *linha*, int *coluna*)

Contrutor utilizado para inicializar uma matriz com suas dimensões.

Parameters

linha	Número de linhas
coluna	Número de colunas

2.1.3 Member Function Documentation

2.1.3.1 void Matriz::AlterarValor (int *i*, double *valor*)

Alterar valor no vetor que representa a matriz.

Parameters

i	Índice direto do vetor
valor	Valor a ser inserido no índice do vetor

2.1.3.2 void Matriz::AlterarValor (int *x*, int *y*, double *valor*)

Alterar valor utilizando coordenadas da matriz.

Parameters

x	Linha
y	Coluna
valor	Valor a ser inserido

2.1.3.3 float Matriz::BuscarValor (int *x*, int *y*)

Buscar valor utilizando coordenadas da matriz.

Parameters

x	Linha
y	Coluna

Returns

Valor armazenado nas coordenadas indicadas

2.1.3.4 void Matriz::CarregarIdentidade ( )

Preencher a matriz atual com a matriz identidade.

[1][0][0]
[0][1][0]
[0][0][1]
Podendo variar de acordo com as dimenções da matriz.

2.1.3.5 int Matriz::GetColuna ( )

Método para acessar a quantidade de colunas de uma matriz.

Returns

Quantidade de colunas da matriz

2.1.3.6 int Matriz::GetLinha ( )

Método para acessar a quantidade de linhas de uma matriz.

Returns

Quantidade de linhas da matriz

2.1.3.7 GLdouble * Matriz::MatrixToGLDArray ( )

Converter matriz atual um array.

Este método é utilizado quando for necessário um vetor GLdouble

Returns

2.1.3.8 Matriz * Matriz::Multiplicar (Matriz * *m*)

Método utilizado para multiplica duas matrizes.

Parameters

m	Matriz que será multiplicada pela matriz atual

Returns

Uma nova matriz contendo o resultado da multiplicaçao

2.1.3.9 void Matriz::MultiplicaTransformacao (Matriz * *m*)

Semelhante ao método Multiplicar.

Neste método o resultado da multiplicação é salvo na matriz atual.

See also

Multiplicar();

Parameters

m	Matriz que será multiplicada pela matriz atual

2.1.3.10 float Matriz::RetornaValor (int *indice*)

Buscar valor no vetor que representa a matriz.

Parameters

indice	Índice direto do vetor

Returns

Valor armazenado no índice indicado

2.1.3.11 Matriz * Matriz::Somar (Matriz * *m*)

Somar duas matrizes.

Parameters

m	Matriz que será multiplicada a matriz atual

Returns

Uma nova matriz contendo o resultado da soma

2.1.3.12 Matriz * Matriz::Subtrair (Matriz * *m*)

Subtrair duas matrizes.

Parameters

m	Matriz que será subtraída a matriz atual

Returns

Uma nova matriz contendo o resultado da soma

2.1.3.13 Matriz * Matriz::Transposta ( )

Calculo da transposta da matriz atual.

Returns

Matriz transposta da matriz atual

The documentation for this class was generated from the following files:

- matriz.h
- matriz.cpp

2.2 Objeto3d Class Reference

Classe que representa um objeto 3D.

`#include <objeto3d.h>`

Public Member Functions

- Objeto3d ()
  Construtor padrão.
- void RotacionaX (double teta)
  Rotacionar o objeto no eixo X.
- void RotacionaY (double teta)
  Rotacionar o objeto no eixo Y.
- void RotacionaZ (double teta)
  Rotacionar o objeto no eixo Z.
- void Desenha ()
  Desenhar objeto.
- GLdouble * GetMatrizRotacao ()
  Recuperar matriz de rotação atual.
- void WriteData (char *texto, GLfloat x, GLfloat y, GLfloat z)
  Escrever informações do objeto em um ponto da janela.
- void AumentarEscala ()
  Aumentar o fator de escala do objeto.
- void DiminuirEscala ()
  Diminuir o fator de escala do objeto.

2.2.1 Detailed Description

Classe que representa um objeto 3D.

Esta classe emgloba propriedades e operações disponíveis para um objeto 3D

2.2.2 Member Function Documentation

2.2.2.1 GLdouble * Objeto3d::GetMatrizRotacao ( )

Recuperar matriz de rotação atual.

Returns

Vetor contendo os valores da matriz de rotação

2.2.2.2 void Objeto3d::RotacionaX (double *teta*)

Rotacionar o objeto no eixo X.

Gerar a matrix de rotação no eixo X com o ângulo indicado

Parameters

Angulo	de rotação

2.2.2.3 void Objeto3d::RotacionaY (double *teta*)

Rotacionar o objeto no eixo Y.

Gerar a matrix de rotação no eixo Y com o ângulo indicado

Parameters

Angulo	de rotação

2.2.2.4 void Objeto3d::RotacionaZ (double *teta*)

Rotacionar o objeto no eixo Z.

Gerar a matrix de rotação no eixo Z com o ângulo indicado

Parameters

Angulo	de rotação

2.2.2.5 void Objeto3d::WriteData (char * *texto*, GLfloat *x*, GLfloat *y*, GLfloat *z*)

Escrever informações do objeto em um ponto da janela.

Parameters

Texto	a ser escrito
x	Posição X onde texto objeto será desenhado
y	Posição Y onde texto objeto será desenhado
z	Posição Z onde texto objeto será desenhado

The documentation for this class was generated from the following files:

- objeto3d.h
- objeto3d.cpp

2.3 Utils Class Reference

Class com método e propriedades úteis a todo o sistema.

`#include <utils.h>`

Public Types

- enum RotationEnum { ROT_X, ROT_Y, ROT_Z, ROT_STOP }

  Enum que representa os tipos de rotações.
- enum GroundTypeEnum { EMPTY, SQUARES, POLYGON, CUBE }

  Enum que representa os tipos de solo que podem ser desenhados.

Public Member Functions

- Utils ()

  Construtor padrão.

2.3.1 Detailed Description

Class com método e propriedades úteis a todo o sistema.

2.3.2 Member Enumeration Documentation

2.3.2.1 enum Utils::GroundTypeEnum

Enum que representa os tipos de solo que podem ser desenhados.

Enumerator

EMPTY Utilizado para representar o chão vazio.

SQUARES Utilizado para representar o chão preenchido com quadrados.

POLYGON Utilizado para representar o chão preenchido com polígonos sólidos.

CUBE Utilizado para representar o chão, teto e paredes feitos com linhas.

2.3.2.2 enum Utils::RotationEnum

Enum que representa os tipos de rotações.

Enumerator

ROT_X Utilizado para representar rotação no eixo X.

ROT_Y Utilizado para representar rotação no eixo Y.

ROT_Z Utilizado para representar rotação no eixo Z.

ROT_STOP Utilizado para representar o fim da rotação. O objeto permanece parado.

The documentation for this class was generated from the following files:

- utils.h
- utils.cpp

Index