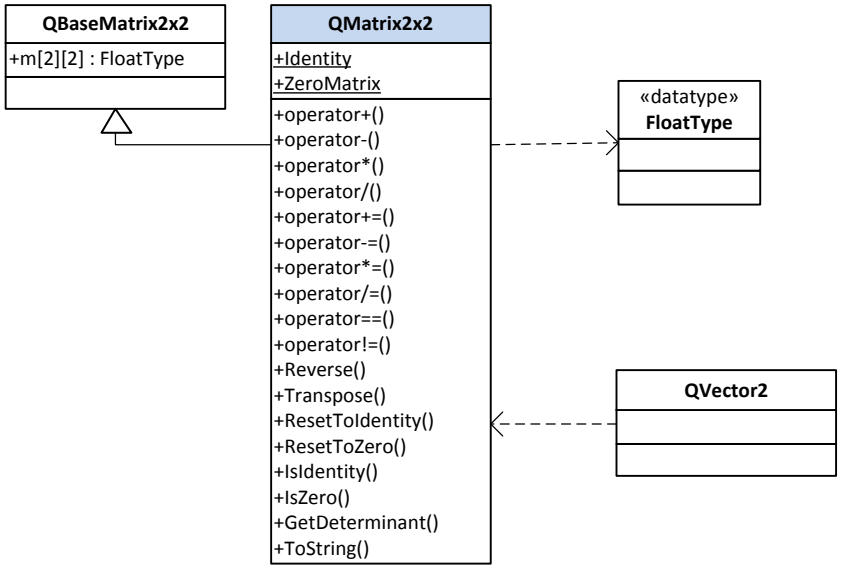
QMatrix2x2

Diagrams



Knowledge Requirements

- Math
- See: Introduction to 3D Game Programming with DirectX 9.0, Part I.
- See: *<Program Files Folder>\Microsoft DirectX SDK (June 2010)\Documentation\DirectX9\windows_graphics.chm*, from DirectX SDK. Search for D3DXMATRIX2 and D3DXMatrix in "Index" tab.
- See: *http://www.zator.com/Cpp/E4_9_18.htm* to refresh operators overloading knowledge.

Functional Specifications

- Override default constructor. Sets attributes to zero.
- Override copy constructor.
- Implement constructor that receives a QBaseMatrix2x2 type.
- Implement constructor that receives 1 FloatType. The matrix must be initialized as if it was a rotation matrix, using input angle.
- Implement constructor that receives a 4-FloatTypes array. Remember that Quimera Engine uses a row x column convention.
- Implement constructor that receives a pointer-to-FloatType. The pointer should point to a dynamically allocated 4-FloatTypes array. Remember that Quimera Engine uses a row x column convention.
- It is not necessary to override default destructor.
- It is not necessary to override assign operator.
- Operator* must offer an overload that receives a FloatType (product by scalar).
- Operator* must offer an overload that receives a QBaseMatrix2x2.
- Operator/ implements "division by scalar" only, there is no division operation between matrices.
- Operator/= implements "division by scalar" only, there is no division operation between matrices.
- A global operator* must be implemented in order to let a FloatType be multiplied by a QMatrix2x2. It's not the same QMatrix2x2 * FloatType than FloatType * QMatrix2x2.

- A global operator/ must be implemented in order to let a FloatType be multiplied by a QMatrix2x2. It's not the same QMatrix2x2 / FloatType than FloatType / QMatrix2x2.
- ResetToZero sets all matrix's elements to 0.
- ToString format: "M2x2(11, 12)(21, 22)". Use STL string.

Design / Technical Requirements

- Use member initialization lists.
- Remember using "explicit" when constructors receive only one parameter.
- No virtual methods.
- Use by-reference parameters always.
- Operator== and Operator!= must have Epsilon value into account.
- Use FloatType constants to store values like 0.
- All methods should be inline.
- No exceptions.
- No error codes.
- No profiling.
- Check for division by zero. Use asserts.
- Respect diagram names.

Support People

- Thund.