Vector2D: A Basic 2D Vector Class

Vector2D.h — Introduction

```cpp
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

© Dr Markus Lumpe, 2022

Vector2D: Private Members

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

vector coordinates

Line: 2:16 C++ Tab Size: 4

© Dr Markus Lumpe, 2022

Vector2D: Public Members

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

Basic vector operations

© Dr Markus Lumpe, 2022

Vector2D: Friends

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

Support for C++ stream-based I/O

© Dr Markus Lumpe, 2022

Vector2D: Ad hoc Definitions

Vector2D.h — Introduction

```
4   #pragma once
5
6   #include <iostream>
7
8   class Vector2D
9   {
10  private:
11
12      float fX;
13      float fY;
14
15  public:
16
17      Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
18      Vector2D( std::istream& aIStream ) { aIStream >> *this; }
19
20      float getX() const;
21      float getY() const;
22
23      Vector2D operator+( const Vector2D& aVector ) const;
24      Vector2D operator-( const Vector2D& aVector ) const;
25
26      Vector2D operator*( const float aScalar ) const;
27      float dot( const Vector2D& aVector ) const;
28      float cross( const Vector2D& aVector ) const;
29
30      float length() const;
31      Vector2D normalize() const;
32
33      float direction() const;
34      Vector2D align( float aAngleInDegrees ) const;
35
36      friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
37      friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
38  };
39
40  Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

Support for scalar * vector

© Dr Markus Lumpe, 2022

Everything that should or must not change is marked with the **const** keyword.

© Dr Markus Lumpe, 2022

Initialized Vector2D objects are read-only.

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

const member functions

© Dr Markus Lumpe, 2022

References prevent copies from being made.

© Dr Markus Lumpe, 2022

Reference Parameters (C++-98)

- C++ uses call-by-value as default parameter passing mechanism.

```
void Assign( int aPar, int aVal ) { aPar = aVal; }

Assign( val, 3 );                    // val unchanged
```

- A reference parameter yields call-by-reference:

```
void AssignR( int& aPar, int aVal ) { aPar = aVal; }

AssignR( val, 3 );                   // val is set to 3
```

- A const reference parameter yields call-by-reference, but the value of the parameter is read-only:

```
void AssignCR( const int& aPar, int aVal ) { aPar = aVal; } // error
```

© Dr Markus Lumpe, 2022

Initialized Vector2D objects are read-only.

Vector2D.h — Introduction

```
4   #pragma once
5
6   #include <iostream>
7
8   class Vector2D
9   {
10  private:
11
12      float fX;
13      float fY;
14
15  public:
16
17      Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
18      Vector2D( std::istream& aIStream ) { aIStream >> *this; }
19
20      float getX() const;
21      float getY() const;
22
23      Vector2D operator+( const Vector2D& aVector ) const;
24      Vector2D operator-( const Vector2D& aVector ) const;
25
26      Vector2D operator*( const float aScalar ) const;
27      float dot( const Vector2D& aVector ) const;
28      float cross( const Vector2D& aVector ) const;
29
30      float length() const;
31      Vector2D normalize() const;
32
33      float direction() const;
34      Vector2D align( float aAngleInDegrees ) const;
35
36      friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
37      friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
38  };
39
40  Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

const reference argument:
- call-by-reference
- value not copied
- assignment not possible

reference argument:
- call-by-reference
- value not copied
- assignment possible

© Dr Markus Lumpe, 2022

Class Implementation

© Dr Markus Lumpe, 2022

Include File: Vector2D.h

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{


    ...


};
```

Line: 1 C++ Tab Size: 4

We do not select any namespace yet!

The implementation goes to Vector2D.cpp

© Dr Markus Lumpe, 2022

Implementation File: Vector2D.cpp

Macro definition to make math definitions available (e.g., M_PI)

Vector2D.cpp — Introduction

```
#define _USE_MATH_DEFINES     // must be defined before any #include

#include "Vector2D.h"

#include <cmath>

using namespace std;

Vector2D Vector2D::operator+( const Vector2D& aVector ) const
{
    return Vector2D( fX + aVector.fX, fY + aVector.fY );
}
```

Line: 1 C++ Tab Size: 4 Symbols

Implementation

© Dr Markus Lumpe, 2022

Member Implementation

- When implementing a member function of a class in C++ we must explicitly specify the class name using a scope identifier within the signature of the member function.
- A scope identifier is a name followed by two colons (e.g. Vector2D::).

Scope identifier Vector2D::

r2D.cpp — Introduction

```
11
12  float Vector2D::getX() const
13  {
14      return fX;
15  }
16
17  float Vector2D::getY() const
18  {
19      return fY;
20  }
21
```

Line: 7 C++ Tab Size: 4 _USE...

Getters for Vector2D

© Dr Markus Lumpe, 2022

C++ Code Organization

- Classes are defined in include files (i.e., .h).
- Class members are implemented in source files (i.e., .cpp).
- There are exceptions when working with templates.

© Dr Markus Lumpe, 2022

#pragma once (Visual Studio)

```
#pragma once

/* Body of Header */
```

Guard against repeated inclusion

© Dr Markus Lumpe, 2022

Class Vector2D - Constructors

Vector2D.h — Introduction

```
#pragma once

#include <iostream>

class Vector2D
{
private:

    float fX;
    float fY;

public:

    Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
    Vector2D( std::istream& aIStream ) { aIStream >> *this; }

    float getX() const;
    float getY() const;

    Vector2D operator+( const Vector2D& aVector ) const;
    Vector2D operator-( const Vector2D& aVector ) const;

    Vector2D operator*( const float aScalar ) const;
    float dot( const Vector2D& aVector ) const;
    float cross( const Vector2D& aVector ) const;

    float length() const;
    Vector2D normalize() const;

    float direction() const;
    Vector2D align( float aAngleInDegrees ) const;

    friend std::istream& operator>>( std::istream& aIStream, Vector2D& aVector );
    friend std::ostream& operator<<( std::ostream& aOStream, const Vector2D& aVector );
};

Vector2D operator*( const float aScalar, const Vector2D& aVector );
```

Line: 2:16 C++ Tab Size: 4

Default argument

Note, we use inlined trivial constructors here.

© Dr Markus Lumpe, 2022

Constructor Initializer

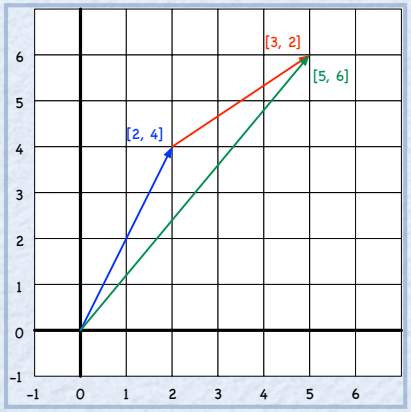
- A constructor initializer is a comma-separated list of member initializers, which is declared between the signature of the constructor and its body.
- Constructor initializers take the form of function calls where the name of the function coincides with name of the instance variable being initialized.

```
Vector2D( float aX = 1.0f, float aY = 0.0f ) : fX(aX), fY(aY) {}
```

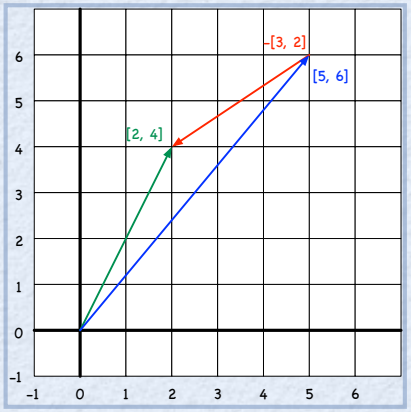

© Dr Markus Lumpe, 2022

Addition & Subtraction

[5,6] = [2,4] + [3,2]



[2,4] = [5,6] - [3,2]



Vector2D.cpp — Introduction

```
Vector2D Vector2D::operator+( const Vector2D& aVector ) const
{
    return Vector2D( fX + aVector.fX, fY + aVector.fY );
}

Vector2D Vector2D::operator-( const Vector2D& aVector ) const
{
    return Vector2D( fX - aVector.fX, fY - aVector.fY );
}
```

Tab Size: 4 _USE_MATH_DEFINES

Member operators

© Dr Markus Lumpe, 2022

Scalar Multiplication, Dot Product, and Cross Product

- We use scalar multiplication to scale a vector uniformly.
- The dot product (inner product) is a measure of the difference between the directions in which the two vectors point.
- The 2D cross product yields a scalar that we use it to determine wether consecutive line segments turn left of right.

Vector2D.cpp — Introduction

```cpp
Vector2D Vector2D::operator*( const float aScalar ) const
{
    return Vector2D( fX * aScalar, fY * aScalar );
}

float Vector2D::dot( const Vector2D& aVector ) const
{
    return fX * aVector.fX + fY * aVector.fY;
}

float Vector2D::cross( const Vector2D& aVector ) const
{
    return fX * aVector.fY - fY * aVector.fX;
}
```

Line: 7 C++ Tab Size: 4 _USE_MATH_DEFINES

© Dr Markus Lumpe, 2022

Vector Length and Unit Vector of a Vector

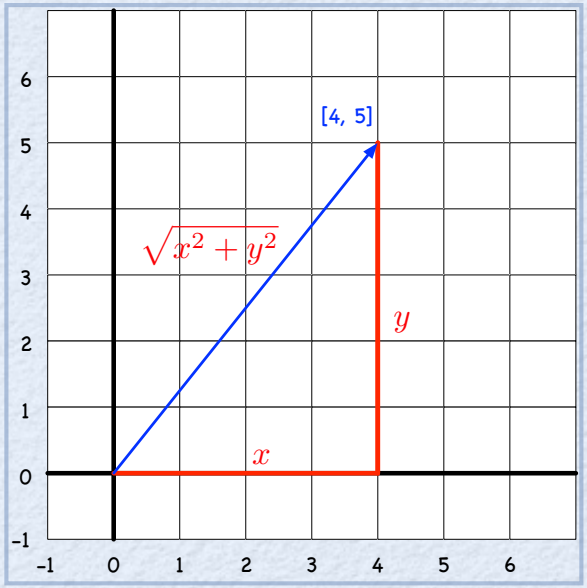
- The length of a vector (magnitude) is the hypotenuse of the right-angled triangle formed by the vector coordinates x and y.
- The unit vector of a vector is a vector with length 1. (In the code below *this refers to the this object, that is, the vector object for which we calculate the unit vector.)

Vector2D.cpp — Introduction

```cpp
float Vector2D::length() const
{
    float val = sqrt(fX * fX + fY * fY);

    return round( val * 100.0f ) / 100.0f;
}

Vector2D Vector2D::normalize() const
{
    return *this * (1.0f/length());
}
```

Line: 11 C++ Tab Size: 4 std

vector length



© Dr Markus Lumpe, 2022

Direction and Align

- The direction of a vector is the arctangent of the right-angled triangle formed by the vector coordinates x and y.
- We can align/rotate a vector, without changing its length, by multiplying its length with the sine and the cosine of the direction angle to obtain the new x and y coordinates, respectively.

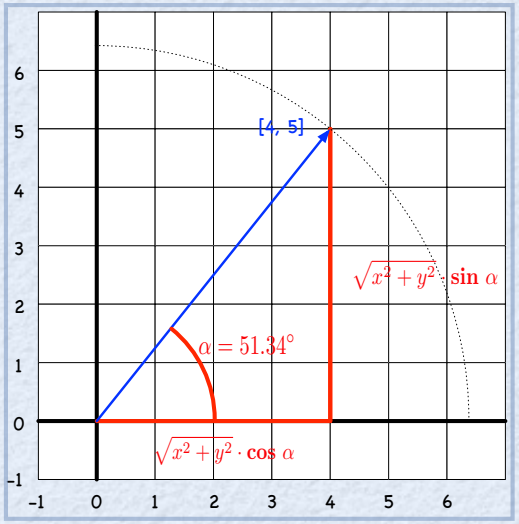
C++ cast M_PI to float

Vector2D.cpp

```cpp
float Vector2D::direction() const
{
    float val = atan2( fY, fX ) * 180.0f / static_cast<float>(M_PI);

    return round( val * 100.0f ) / 100.0f;
}

Vector2D Vector2D::align( float aAngleInDegrees ) const
{
    float lRadians = aAngleInDegrees * static_cast<float>(M_PI) / 180.0f;

    return length() * Vector2D( cos( lRadians ), sin( lRadians ) );
}
```

Line: 11 C++ Tab Size: 4 std

direction/align



© Dr Markus Lumpe, 2022

Ad hoc Operator *

- The member operator for scalar multiplication only allows for vector * scalar. However, multiplication is commutative, that is changing the order of the operands does not change the result.
- We can recover the commutativity of scalar multiplication by defining an ad hoc multiplication operator that takes a scalar as first argument and a vector as the second:

Vector2D.cpp — Introduction

```
83  Vector2D operator*( const float aScalar, const Vector2D& aVector )
84  {
85      return aVector * aScalar;
86  }
87
```

Line: 7 C++ Tab Size: 4 _USE_MATH_DEFINES

© Dr Markus Lumpe, 2022