Laboratory work 2

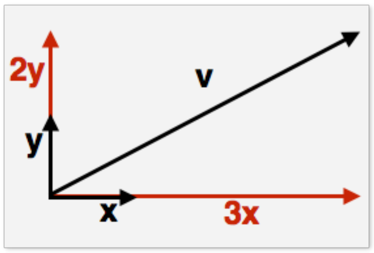
1 Objectives

The objective of this laboratory is to implement specific C++ classes for handling vectors.

2 Vectors

In the field of computer graphics vectors are used to determine the angle between edges, the orientation of surfaces, the relative position of a point to a surface, in the computation of lighting models, and many other things.

We represent vectors by a list of numbers and graphically in a Cartesian coordinate system. We represent vectors as arrows and we name them by using bold letters. Geometrically a vector is described by direction and length. In 2D, a vector can be written as a combination of two not parallel vectors (with a length different from 0). A vector **v** is represented by $\boldsymbol{v} = v_x\boldsymbol{x} + v_y\boldsymbol{y}$, where v_x, v_y are the Cartesian coordinates of the vector. We can write vectors horizontally and call them **row vectors** or we can write them vertically and call them **column vectors**. For the previous example we can write: $\boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$, or $\boldsymbol{v}^T = \begin{bmatrix} v_x & v_y \end{bmatrix}$.



In the previous figure vector **v** can be defined as a combination of the basis vector **x** and **y**.

$$\boldsymbol{v} = 3\boldsymbol{x} + 2\boldsymbol{y} = 3\begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

In computer graphics we are interested in 2D, 3D and 4D vectors and we refer to vector elements by:

- x, y (in 2D)
- x, y, z (in 3D)
- x, y, z, w (in 4D)

Vectors can be used to store displacement (the offset between two points) or locations (represented as displacement from a well known origin). Note however that locations are not vectors (we cannot add “Cluj” to “Bucharest”).

The difference between two points is a vector ($\boldsymbol{v} = Q - P$), and the sum between a point and a vector is a point ($\boldsymbol{v} + P = Q$).

3 Operations

3.1 Vector length

The **length** is denoted by $\|\boldsymbol{v}\|$ and equals the square root of the sum of the square of vector elements. In the 2D case the length is $\|\boldsymbol{v}\| = \sqrt{v_x^2 + v_y^2}$.

A **unit vector** is a vector with length equal to 1. The **zero vector** has the length equal to 0 (in this case the direction is undefined).

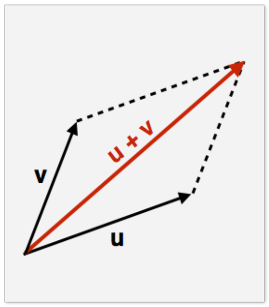
3.2 Vector normalization

We can **normalize** any nonzero vector by dividing the vector by its length. The new vector points in the same direction as the original vector but the length equals 1.

$$\boldsymbol{v} = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$$

3.3 Vector addition

We can **add** two vectors and we sum the corresponding elements from the initial vectors. If we have two 2D vectors $\boldsymbol{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ and $\boldsymbol{v} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$ the result is a new vector equals to:



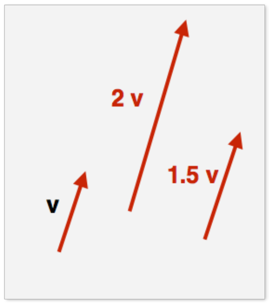
$$\boldsymbol{u} + \boldsymbol{v} = \begin{bmatrix} u_x + v_x \\ u_y + v_y \end{bmatrix}$$

This vector addition operation is commutative.

$$\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$$

3.4 Vector scaling

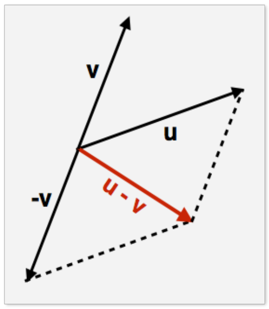
We can scale a vector by a scalar value in order to change the vector length. For this we are multiplying the vector elements with the scalar value. If we scale a vector by -1 we change the vector's direction.



$$\beta \boldsymbol{u} = \begin{bmatrix} \beta u_x \\ \beta u_y \end{bmatrix}$$

3.5 Vector subtraction

Subtraction of two vectors is similar to the addition operation, we just scale with -1 the second vector.

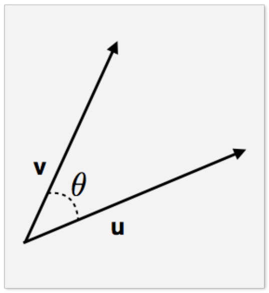


$$\boldsymbol{u} - \boldsymbol{v} = \boldsymbol{u} + (-\boldsymbol{v}) = \begin{bmatrix} u_x - v_x \\ u_y - v_y \end{bmatrix}$$

3.6 Vector multiplication

3.6.1 Dot product

The dot product of two vectors returns a scalar value related to the vectors' length and the angle between them.

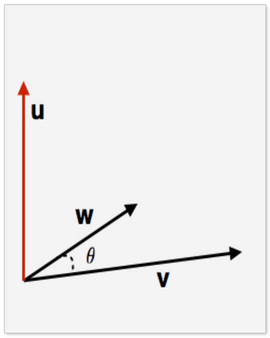


$$\boldsymbol{u} \cdot \boldsymbol{v} = \|\boldsymbol{u}\|\|\boldsymbol{v}\| cos\theta$$

If two vectors **u** and **v** are represented in Cartesian coordinate then $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}_x\boldsymbol{v}_x + \boldsymbol{u}_y\boldsymbol{v}_y$. Similar in 3D the dot product is $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}_x\boldsymbol{v}_x + \boldsymbol{u}_y\boldsymbol{v}_y + \boldsymbol{u}_z\boldsymbol{v}_z$.

3.6.2 Cross product

In computer graphics we are mainly using this product only on 3D vectors, but the product can be generalized. The result is a vector perpendicular to the two vector. The length of the resulting vector is $\|\boldsymbol{v} \times \boldsymbol{w}\| = \|\boldsymbol{v}\|\|\boldsymbol{w}\| sin\theta$. The direction of the vector generates a right-handed coordinate system.



$$\boldsymbol{v} \times \boldsymbol{w} = \begin{bmatrix} \boldsymbol{v}_y\boldsymbol{w}_z - \boldsymbol{v}_z\boldsymbol{w}_y \\ \boldsymbol{v}_z\boldsymbol{w}_x - \boldsymbol{v}_x\boldsymbol{w}_z \\ \boldsymbol{v}_x\boldsymbol{w}_y - \boldsymbol{v}_y\boldsymbol{w}_x \end{bmatrix}$$

4 Assignment

Download the source code from the web repository. You have to implement the methods inside the source files (vec2.cpp, vec3.cpp, and vec4.cpp). The header files contain the definition of classes and the methods that should be implemented.