Assignment: A9 (return to main page)

Vector Class

This assignment introduces some basic vector math. We will use this in the next assignments to model 2D collisions of circular objects and in the rotation of objects such as jet tubes and gun turrets.

Vector addition and subtraction:

http://en.wikipedia.org/wiki/Vector_addition#Addition_and_subtraction

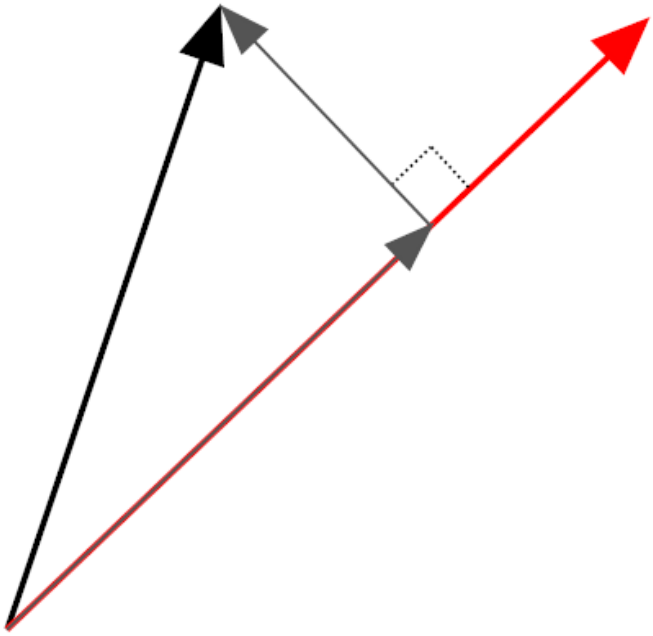
All basic vector arithmetic is handled with components (no use of trigonometry). Vectors are instantiated with x and y components (no angle in the definition).

Vector projection and the dot product:

http://en.wikipedia.org/wiki/Dot_product

http://en.wikipedia.org/wiki/Vector_projection

The essentials of this are illustrated in the drawing to the right. Here the black vector is resolved into the two grey components, one along the length of (parallel to) the red vector and one perpendicular to the red vector. The parallel component is calculated using vector projection which is based on the dot product. The perpendicular component can be calculated by projecting onto a 90° rotated version of the red vector or by subtracting the parallel component from the original black vector.



Note that we will be using a version of vector projection that does not involve the square root operation. This is computationally most efficient. This approach uses two dot products and one scalar operation.

$$\vec{a}_{parallel\_to\_b} = \frac{\vec{a} \cdot \vec{b}}{\vec{b} \cdot \vec{b}} \vec{b}$$

Vector rotation:

http://en.wikipedia.org/wiki/Rotation_%28mathematics%29

This reference on rotation describes a general transformation for 2D rotation. This requires the use of the sine and cosine function in the Python math module. Our vector class will also need some specific rotation methods for 90° and 180° rotations; these can be done simply with component manipulations (no trig).

Python language topics:

- Python modules
  http://docs.python.org/2/tutorial/modules.html

Problem statement:

(Another clean slate here; we won't be building onto the previous assignment.)

- Make a separate module file that contains this vector class. Import this class from the module file into a demo/test program.
- Try this from the Python command line. Import the module and then run some live vector manipulations from the command line.
- The test program should illustrate the following operations:
  - Vector addition and subtraction using component operations.
  - Finding a vector's parallel component along the direction of a second vector (projection).
    - Find the corresponding perpendicular component using projection.
    - Find the corresponding perpendicular component using subtraction.
  - General vector rotation using a trigonometric transform.
    - Specific 90° and 180° rotations using only component manipulations.

Algorithmic description:

- See the vector class description and reference links above. The class will have an __init__ function that defines the *x* and *y* components of the vector. Vector operations will be defined in the vector class methods.

Python code: (see source code on web page…)

Obfuscated code or incremental code (again) didn't seem useful for this assignment, so there are no screen shots of code here.

This is another case where it's probably best to start out looking at the source code for the class module and then do some reading in the reference links above. Next, try to make your own test/demo program. Then look at the Vector Sandbox code (and video) provided on the web page.

Here's a command line session:

```
>>> from vec2d_jdm import Vec2D
>>> a = Vec2D(1,2)
>>> print a + a
Vec2D(2.0, 4.0)
>>> print a.rotated(90)
Vec2D(-2.0, 1.0)
>>> print a.rotate90()
Vec2D(-2.0, 1.0)
>>> print a.rotated(90).rotated(-90)
Vec2D(1.0, 2.0)
>>> print a.projection_onto(Vec2D(1,1))
Vec2D(1.5, 1.5)
>>> print Vec2D(2,1).projection_onto(Vec2D(1,1))
Vec2D(1.5, 1.5)
>>>
>>> a.length()
2.236067977499789
>>> print a * 10.0
Vec2D(10.0, 20.0)
```