**Pygame 官方文档 - pygame.math**

pygame.math

pygame模块关于向量类  
[pygame.math.Vector2](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2) - 一个二维向量  
[pygame.math.Vector3](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3) - 一个三维向量  
[pygame.math.enable\_swizzling](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#enable_swizzling%28%29) - 全局允许对向量进行调配  
[pygame.math.disable\_swizzling](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#disable_swizzling%28%29) - 全局禁用对向量进行调配

        pygame math模块目前分别提供Vector2（二维）和Vector3（三维）的Vector类。  
        它们支持以下数值运算：vec + vec，vec-vec，vec \* number，number \* vec，vec / number，vec // number，vec + = vec，vec- = vec，vec \* = number，vec / = number ，vec // =number。 所有这些操作都将按元素执行。 此外，vec \* vec将执行标量积（又叫点积）。 如果你想将vector v中的每个元素与vector w中的每个元素相乘，你可以使用按元素方法：v.elementwise()\ \* w  
        pygame 1.9.2pre中的新功能。 1.9.4删除实验通知。 1.9.4将构造函数更改为需要2个或3个元素，而不是指定0默认值。 1.9.4允许标量构造如GLSL Vector2(2)== Vector2(2.0,2.0) 1.9.4 pygame.math需要导入。 更方便的pygame.Vector2和pygame.Vector3。

pygame.math.Vector2 [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#title)

一个二维向量  
Vector2() -> Vector2  
Vector2(int) -> Vector2  
Vector2(float) -> Vector2  
Vector2(Vector2) -> Vector2  
Vector2(x, y) -> Vector2  
Vector2((x, y)) -> Vector2

[pygame.math.Vector2.dot](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.dot%28%29) - 使用另一个向量计算点或标量积  
[pygame.math.Vector2.cross](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.cross%28%29) - 计算叉或矢量积  
[pygame.math.Vector2.magnitude](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.magnitude%28%29) - 返回向量的欧几里德矢量大小。  
[pygame.math.Vector2.magnitude\_squared](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.magnitude_squared%28%29) - 返回向量的平方大小。  
[pygame.math.Vector2.length](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.length%28%29) - 返回向量的欧几里德长度。  
[pygame.math.Vector2.length\_squared](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.length_squared%28%29) - 返回向量的平方欧几里德长度。  
[pygame.math.Vector2.normalize](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.normalize%28%29) - 返回方向相同但长度为1的向量。  
[pygame.math.Vector2.normalize\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.normalize_ip%28%29) - 将矢量就原地位置正规化，使其长度为1。  
[pygame.math.Vector2.is\_normalized](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.is_normalized%28%29) - 测试向量是否正规化，即长度== 1。  
[pygame.math.Vector2.scale\_to\_length](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.scale_to_length%28%29) - 将矢量缩放到给定长度。  
[pygame.math.Vector2.reflect](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.reflect%28%29) - 返回给定法线的反射向量。  
[pygame.math.Vector2.reflect\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.reflect_ip%28%29) - 就原地位置给定法线的反射向量。  
[pygame.math.Vector2.distance\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.distance_to%28%29) - 计算给定矢量的欧几里德距离。  
[pygame.math.Vector2.distance\_squared\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.distance_squared_to%28%29) - 计算到给定向量的平方欧几里德距离。  
[pygame.math.Vector2.lerp](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.lerp%28%29) - 返回给定向量的线性插值。  
[pygame.math.Vector2.slerp](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.slerp%28%29) - 返回给定向量的球面插值。  
[pygame.math.Vector2.elementwise](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.elementwise%28%29) - 下一个操作将按元素执行。  
[pygame.math.Vector2.rotate](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.rotate%28%29) - 以角度为单位旋转给定角度的矢量。  
[pygame.math.Vector2.rotate\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.rotate_ip%28%29) - 将矢量旋转一个给定角度（以度为单位）。  
[pygame.math.Vector2.angle\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.angle_to%28%29) - 以角度为单位计算给定矢量的角度。  
[pygame.math.Vector2.as\_polar](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.as_polar%28%29) - 返回一个具有径向距离和方位角的元组。  
[pygame.math.Vector2.from\_polar](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2.from_polar%28%29) - 从极坐标元组中设置x和y。  
关于Vector2类的一些一般信息。

pygame.math.Vector2.dot() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

使用另一个向量计算点或标量积  
dot(Vector2) -> float  
[搜索pygame.math.Vector2.dot的示例](https://github.com/search?q=pygame.math.Vector2.dot&type=Code)

pygame.math.Vector2.cross() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

计算叉或矢量积  
cross(Vector2) -> Vector2  
计算叉积的第三个分量。  
[搜索pygame.math.Vector2.cross的示例](https://github.com/search?q=pygame.math.Vector2.cross&type=Code)

pygame.math.Vector2.magnitude() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回向量的欧几里德矢量大小。  
magnitude() -> float  
根据以下定理计算矢量的大小：vec.large() == math.sqrt(vec.x\*\*2+vec.y\*\*2)  
[搜索pygame.math.Vector2.magnitude的示例](https://github.com/search?q=pygame.math.Vector2.magnitude&type=Code)

pygame.math.Vector2.magnitude\_squared() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回向量的平方大小。  
magnitude\_squared() -> float  
根据以下定理计算矢量的大小：vec.large\_squared() == vec.x\*\*2+vec.y\*\*2这比vec.large()快，因为它避免了平方根。  
[搜索pygame.math.Vector2.magnitude\_squared的示例](https://github.com/search?q=pygame.math.Vector2.magnitude_squared&type=Code)

pygame.math.Vector2.length() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回向量的欧几里德长度。  
length() -> float  
根据勾股定理计算向量的欧几里得长度：vec.length() == math.sqrt(vec.x\*\*2+vec.y\*\*2)  
[搜索pygame.math.Vector2.length的示例](https://github.com/search?q=pygame.math.Vector2.length&type=Code)

pygame.math.Vector2.length\_squared() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回向量的平方欧几里德长度。  
length\_squared() -> float  
根据勾股定理计算向量的欧几里得长度：vec.length\_square() == vec.x\*\*2+vec.y\*\*2这比vec.length()更快，因为它避免了平方根。  
[搜索pygame.math.Vector2.length\_squared的示例](https://github.com/search?q=pygame.math.Vector2.length_squared&type=Code)

pygame.math.Vector2.normalize() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回方向相同但长度为1的向量。  
normalize() -> Vector2  
返回一个新的向量，该向量的长度=1，方向与self相同。  
[搜索pygame.math.Vector2.normalize的示例](https://github.com/search?q=pygame.math.Vector2.normalize&type=Code)

pygame.math.Vector2.normalize\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

将矢量就原地位置正规化，使其长度为1。  
normalize\_ip() -> None  
正规化向量，使其长度=1。矢量的方向不变。  
[搜索pygame.math.Vector2.normalize\_ip的示例](https://github.com/search?q=pygame.math.Vector2.normalize_ip&type=Code)

pygame.math.Vector2.is\_normalized() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

测试向量是否正规化，即长度== 1。  
is\_normalized() -> Bool  
如果向量的长度=1，则返回true。否则返回false。  
[搜索pygame.math.Vector2.is\_normalized的示例](https://github.com/search?q=pygame.math.Vector2.is_normalized&type=Code)

pygame.math.Vector2.scale\_to\_length() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

将矢量缩放到给定长度。  
scale\_to\_length(float) -> None  
缩放向量，使其具有给定的长度。矢量的方向不变。也可以缩放到长度0。如果矢量为零矢量（即长度为0，因此没有方向），则会产生ZeroDivisionError 异常。  
[搜索pygame.math.Vector2.scale\_to\_length的示例](https://github.com/search?q=pygame.math.Vector2.scale_to_length&type=Code)

pygame.math.Vector2.reflect() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回给定法线的反射向量。  
reflect(Vector2) -> Vector2  
返回一个新的向量，该向量指向的方向就像self将在具有给定surface法向特征的surface上反弹一样。新向量的长度与self的长度相同。  
[搜索pygame.math.Vector2.reflect的示例](https://github.com/search?q=pygame.math.Vector2.reflect&type=Code)

pygame.math.Vector2.reflect\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

就原地位置给定法线的反射向量。  
reflect\_ip(Vector2) -> None  
改变自己的方向，就好像它是一个具有给surface面法向的曲面的反射。  
[搜索pygame.math.Vector2.reflect\_ip的示例](https://github.com/search?q=pygame.math.Vector2.reflect_ip&type=Code)

pygame.math.Vector2.distance\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

计算给定矢量的欧几里德距离。  
distance\_to(Vector2) -> float  
[搜索pygame.math.Vector2.distance\_to的示例](https://github.com/search?q=pygame.math.Vector2.distance_to&type=Code)

pygame.math.Vector2.distance\_squared\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

计算到给定向量的平方欧几里德距离。  
distance\_squared\_to(Vector2) -> float  
[搜索pygame.math.Vector2.distance\_squared\_to的示例](https://github.com/search?q=pygame.math.Vector2.distance_squared_to&type=Code)

pygame.math.Vector2.lerp() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回给定向量的线性插值。  
lerp(Vector2, float) -> Vector2  
返回一个向量，该向量是自己和给定向量之间的线性插值。第二个参数决定了结果在自己和其它之间的距离。它必须是介于0和1之间的值，其中0表示自身，1表示将返回其他值。  
[搜索pygame.math.Vector2.lerp的示例](https://github.com/search?q=pygame.math.Vector2.lerp&type=Code)

pygame.math.Vector2.slerp() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回给定向量的球面插值。  
slerp(Vector2, float) -> Vector2  
计算从自己到给定向量的球面插值。第二个参数（通常称为t）必须在范围内[-1，1]。它参数化了两个向量之间的结果。如果给定一个负值，则插值不会取最短路径的补数。  
[搜索pygame.math.Vector2.slerp的示例](https://github.com/search?q=pygame.math.Vector2.slerp&type=Code)

pygame.math.Vector2.elementwise() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

下一个操作将按元素执行。  
elementwise() -> VectorElementwiseProxy  
将以下操作应用于向量的每个元素。  
[搜索pygame.math.Vector2.elementwise的示例](https://github.com/search?q=pygame.math.Vector2.elementwise&type=Code)

pygame.math.Vector2.rotate() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

以角度为单位旋转给定角度的矢量。  
rotate(float) -> Vector2  
返回一个与self具有相同长度但向逆时针旋转给定角度的矢量。  
[搜索pygame.math.Vector2.rotate的示例](https://github.com/search?q=pygame.math.Vector2.rotate&type=Code)

pygame.math.Vector2.rotate\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

将矢量旋转一个给定角度（以度为单位）。  
rotate\_ip(float) -> None  
逆时针旋转矢量给定角度（以角度为单位）。矢量的长度不会改变。  
[搜索pygame.math.Vector2.rotate\_ip的示例](https://github.com/search?q=pygame.math.Vector2.rotate_ip&type=Code)

pygame.math.Vector2.angle\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

以角度为单位计算给定矢量的角度。  
angle\_to(Vector2) -> float  
返回self和给定向量之间的角度。  
[搜索pygame.math.Vector2.angle\_to的示例](https://github.com/search?q=pygame.math.Vector2.angle_to&type=Code)

pygame.math.Vector2.as\_polar() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

返回一个具有径向距离和方位角的元组。  
as\_polar() -> (r, phi)  
返回元组(r, phi)，其中r是径向距离，phi是方位角。  
[搜索pygame.math.Vector2.as\_polar的示例](https://github.com/search?q=pygame.math.Vector2.as_polar&type=Code)

pygame.math.Vector2.from\_polar() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector2)

从极坐标元组中设置x和y。  
from\_polar((r, phi)) -> None  
从元组 (r, phi)设置x和y，其中r是径向距离，phi是方位角。  
[搜索pygame.math.Vector2.from\_polar的示例](https://github.com/search?q=pygame.math.Vector2.from_polar&type=Code)

pygame.math.Vector3 [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#title)

一个三维向量  
Vector3() -> Vector3  
Vector3(int) -> Vector2  
Vector3(float) -> Vector2  
Vector3(Vector3) -> Vector3  
Vector3(x, y, z) -> Vector3  
Vector3((x, y, z)) -> Vector3

[pygame.math.Vector3.dot](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.dot%28%29)-用另一个向量计算点或标量积  
[pygame.math.Vector3.cross](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.cross%28%29)-计算叉积或矢量积  
[pygame.math.Vector3.magnitude](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.magnitude%28%29)-返回向量的欧几里得大小  
[pygame.math.Vector3.magnitude\_squared](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.magnitude_squared%28%29)-返回向量的欧几里得平方大小  
[pygame.math.Vector3.length](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.length%28%29)-返回向量的欧几里得长度  
[pygame.math.Vector3.length\_squared](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.length_squared%28%29)-返回向量的欧几里得平方长度  
[pygame.math.Vector3.normalize](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.normalize%28%29)-返回一个方向相同但长度为1的向量  
[pygame.math.Vector3.normalize\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.normalize_ip%28%29)-在适当位置正规化向量，使其长度为1  
[pygame.math.Vector3.is\_normalized](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.is_normalized%28%29)-如果向量是正规化的，即长度=1，则测试  
[pygame.math.Vector3.scale\_to\_length](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.scale_to_length%28%29)-将向量缩放到给定的长度  
[pygame.math.Vector3.reflect](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.reflect%28%29)-返回给定法线的反射向量  
[pygame.math.Vector3.reflect\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.reflect_ip%28%29)-就地反映给定法向量  
[pygame.math.Vector3.distance\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.distance_to%28%29)-计算到给定向量的欧几里得距离  
[pygame.math.Vector3.distance\_squared\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.distance_squared_to%28%29)-计算到给定向量的欧几里得平方距离  
[pygame.math.Vector3.lerp](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.lerp%28%29)-返回给定向量的线性插值  
[pygame.math.Vector3.slerp](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.slerp%28%29)-返回给定向量的球面插值  
[pygame.math.Vector3.elementwise](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.elementwise%28%29)-下一步操作将按元素执行  
[pygame.math.Vector3.rotate](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate%28%29)-按给定角度以度旋转向量  
[pygame.math.Vector3.rotate\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_ip%28%29)-将矢量旋转给定角度（以度为单位）  
[pygame.math.Vector3.rotate\_x](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_x%28%29)-将一个矢量绕X轴旋转角度  
[pygame.math.Vector3.rotate\_x\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_x_ip%28%29)-将矢量绕X轴原处旋转一定角度  
[pygame.math.Vector3.rotate\_y](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_y%28%29)-以度为单位围绕y轴旋转向量  
[pygame.math.Vector3.rotate\_y\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_y_ip%28%29)-将矢量绕y轴原处旋转一定角度  
[pygame.math.Vector3.rotate\_z](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_z%28%29)-围绕z轴旋转一个以角度为单位的角度  
[pygame.math.Vector3.rotate\_z\_ip](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.rotate_z_ip%28%29)-将矢量绕Z轴原处旋转角度（以角度为单位）  
[pygame.math.Vector3.angle\_to](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.angle_to%28%29)-计算到给定向量的角度  
[pygame.math.Vector3.as\_spherical](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.as_spherical%28%29)-返回一个具有径向距离、倾角和方位角的元组  
[pygame.math.Vector3.from\_spherical](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3.from_spherical%28%29)-从球面坐标3元组设置x、y和z

关于Vector3类的一些一般信息。

pygame.math.Vector3.dot() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

用另一个向量计算点或标量积  
dot(Vector3) -> float  
[搜索pygame.math.Vector3.dot的示例](https://github.com/search?q=pygame.math.Vector3.dot&type=Code)

pygame.math.Vector3.cross() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

计算叉积或矢量积  
cross(Vector3) -> Vector3  
计算叉积  
[搜索pygame.math.Vector3.cross的示例](https://github.com/search?q=pygame.math.Vector3.cross&type=Code)

pygame.math.Vector3.magnitude() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回向量的欧几里得大小  
magnitude() -> float  
根据以下定理计算矢量的大小：vec.large() == math.sqrt(vec.x\*\*2+vec.y\*\*2+vec.z\*\*2)  
[搜索pygame.math.Vector3.magnitude的示例](https://github.com/search?q=pygame.math.Vector3.magnitude&type=Code)

pygame.math.Vector3.magnitude\_squared() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回向量的欧几里得平方大小  
magnitude\_squared() -> float  
根据以下定理计算矢量的大小：vec.large\_squared() == vec.x\*\*2+vec.y\*\*2+vec.z\*\*2这比vec.large()快，因为它避免了平方根。  
[搜索pygame.math.Vector3.magnitude\_squared的示例](https://github.com/search?q=pygame.math.Vector3.magnitude_squared&type=Code)

pygame.math.Vector3.length() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回向量的欧几里得长度  
length() -> float  
根据勾股定理计算向量的欧几里得长度：  
vec.length() == math.sqrt(vec.x\*\*2+vec.y\*\*2+vec.z\*\*2)  
[搜索pygame.math.Vector3.length的示例](https://github.com/search?q=pygame.math.Vector3.length&type=Code)

pygame.math.Vector3.length\_squared() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回向量的欧几里得平方长度  
length\_squared() -> float  
根据勾股定理计算向量的欧几里得长度：vec.length\_squared() ==vec.x\*\*2+vec.y\*\*2+vec.z\*\*2这比vec.length()快，因为它避免了平方根。  
[搜索pygame.math.Vector3.length\_squared的示例](https://github.com/search?q=pygame.math.Vector3.length_squared&type=Code)

pygame.math.Vector3.normalize() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回一个方向相同但长度为1的向量  
normalize() -> Vector3  
返回一个长度为== 1且方向与self相同的新向量。  
[搜索pygame.math.Vector3.normalize的示例](https://github.com/search?q=pygame.math.Vector3.normalize&type=Code)

pygame.math.Vector3.normalize\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

在适当位置正规化向量，使其长度为1  
normalize\_ip() -> None  
正规化向量，使其长度=1。矢量的方向不变。  
[搜索pygame.math.Vector3.normalize\_ip的示例](https://github.com/search?q=pygame.math.Vector3.normalize_ip&type=Code)

pygame.math.Vector3.is\_normalized() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

如果向量是正规化的，即长度=1，则测试  
is\_normalized() -> Bool  
如果向量的长度为1，则返回True。否则返回False。  
[搜索pygame.math.Vector3.is\_normalized的示例](https://github.com/search?q=pygame.math.Vector3.is_normalized&type=Code)

pygame.math.Vector3.scale\_to\_length() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将向量缩放到给定的长度  
scale\_to\_length(float) -> None  
缩放矢量，使其具有给定的长度。矢量的方向不会改变。您也可以缩放到长度0.如果向量是零向量（即长度为0因此没有方向），则引发ZeroDivisionError异常。  
[搜索pygame.math.Vector3.scale\_to\_length的示例](https://github.com/search?q=pygame.math.Vector3.scale_to_length&type=Code)

pygame.math.Vector3.reflect() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回给定法线的反射向量  
reflect(Vector3) -> Vector3  
返回一个新的向量，该向量指向的方向就像self将在具有给定surface法向特征的surface上反弹一样。新向量的长度与self的长度相同。  
[搜索pygame.math.Vector3.reflect的示例](https://github.com/search?q=pygame.math.Vector3.reflect&type=Code)

pygame.math.Vector3.reflect\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

就地反映给定法向量  
reflect\_ip(Vector3) -> None  
改变self的方向，好像它会被给定surface法线的表面反射一样。  
[搜索pygame.math.Vector3.reflect\_ip的示例](https://github.com/search?q=pygame.math.Vector3.reflect_ip&type=Code)

pygame.math.Vector3.distance\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

计算到给定向量的欧几里得距离  
distance\_to(Vector3) -> float  
[搜索pygame.math.Vector3.distance\_to的示例](https://github.com/search?q=pygame.math.Vector3.distance_to&type=Code)

pygame.math.Vector3.distance\_squared\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

计算到给定向量的欧几里得平方距离  
distance\_squared\_to(Vector3) -> float  
[搜索pygame.math.Vector3.distance\_squared\_to的示例](https://github.com/search?q=pygame.math.Vector3.distance_squared_to&type=Code)

pygame.math.Vector3.lerp() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回给定向量的线性插值  
lerp(Vector3, float) -> Vector3  
返回一个向量，该向量是自己和给定向量之间的线性插值。第二个参数决定了结果在自己和其它之间的距离。它必须是介于0和1之间的值，其中0表示自己，1表示将返回其他值。  
[搜索pygame.math.Vector3.lerp的示例](https://github.com/search?q=pygame.math.Vector3.lerp&type=Code)

pygame.math.Vector3.slerp() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回给定向量的球面插值  
slerp(Vector3, float) -> Vector3  
计算从自身到给定向量的球面插值。第二个参数（通常称为t）必须在范围内[-1，1]。它参数化了两个向量之间的结果。如果给定一个负值，则插值不会取最短路径的补数。  
[搜索pygame.math.Vector3.slerp的示例](https://github.com/search?q=pygame.math.Vector3.slerp&type=Code)

pygame.math.Vector3.elementwise() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

下一步操作将按元素执行  
elementwise() -> VectorElementwiseProxy  
将以下操作应用于向量的每个元素。  
[搜索pygame.math.Vector3.elementwise的示例](https://github.com/search?q=pygame.math.Vector3.elementwise&type=Code)

pygame.math.Vector3.rotate() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

按给定角度以度旋转向量  
rotate(Vector3, float) -> Vector3  
返回一个与self具有相同长度但向逆时针旋转给定角度的给定角度的矢量。  
[搜索pygame.math.Vector3.rotate的示例](https://github.com/search?q=pygame.math.Vector3.rotate&type=Code)

pygame.math.Vector3.rotate\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将矢量旋转给定角度（以度为单位）  
rotate\_ip(Vector3, float) -> None  
以给定的角度（以度为单位）围绕给定轴逆时针旋转矢量。矢量的长度不会改变。  
[搜索pygame.math.Vector3.rotate\_ip的示例](https://github.com/search?q=pygame.math.Vector3.rotate_ip&type=Code)

pygame.math.Vector3.rotate\_x() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将一个矢量绕X轴旋转角度  
rotate\_x(float) -> Vector3  
返回一个矢量，该矢量与self具有相同的长度，但绕x轴逆时针旋转给定角度（以角度为单位）。  
[搜索pygame.math.Vector3.rotate\_x的示例](https://github.com/search?q=pygame.math.Vector3.rotate_x&type=Code)

pygame.math.Vector3.rotate\_x\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将矢量绕X轴原处旋转一定角度  
rotate\_x\_ip(float) -> None  
围绕x轴逆时针旋转矢量给定角度（以角度为单位）。矢量的长度不会改变。  
[搜索pygame.math.Vector3.rotate\_x\_ip的示例](https://github.com/search?q=pygame.math.Vector3.rotate_x_ip&type=Code)

pygame.math.Vector3.rotate\_y() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

以度为单位围绕y轴旋转向量  
rotate\_y(float) -> Vector3  
返回一个矢量，该矢量与self具有相同的长度，但是以y度轴逆时针旋转给定角度（以度为单位）。  
[搜索pygame.math.Vector3.rotate\_y的示例](https://github.com/search?q=pygame.math.Vector3.rotate_y&type=Code)

pygame.math.Vector3.rotate\_y\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将矢量绕y轴原处旋转一定角度  
rotate\_y\_ip(float) -> None  
围绕y轴逆时针旋转矢量给定角度（以度为单位）。矢量的长度不会改变。  
[搜索pygame.math.Vector3.rotate\_y\_ip的示例](https://github.com/search?q=pygame.math.Vector3.rotate_y_ip&type=Code)

pygame.math.Vector3.rotate\_z() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

围绕z轴旋转一个以角度为单位的角度  
rotate\_z(float) -> Vector3  
返回一个矢量，该矢量与self具有相同的长度，但是以z度轴逆时针旋转给定角度（以度为单位）。  
[搜索pygame.math.Vector3.rotate\_z的示例](https://github.com/search?q=pygame.math.Vector3.rotate_z&type=Code)

pygame.math.Vector3.rotate\_z\_ip() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

将矢量绕Z轴原处旋转角度（以角度为单位）  
rotate\_z\_ip(float) -> None  
围绕z轴逆时针旋转矢量给定角度（以度为单位）。矢量的长度不会改变。  
[搜索pygame.math.Vector3.rotate\_z\_ip的示例](https://github.com/search?q=pygame.math.Vector3.rotate_z_ip&type=Code)

pygame.math.Vector3.angle\_to() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

计算到给定向量的角度  
angle\_to(Vector3) -> float  
返回self和给定向量之间的角度。  
[搜索pygame.math.Vector3.angle\_to的示例](https://github.com/search?q=pygame.math.Vector3.angle_to&type=Code)

pygame.math.Vector3.as\_spherical() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

返回一个具有径向距离、倾角和方位角的元组  
as\_spherical() -> (r, theta, phi)  
返回一个元组(r, theta, phi)，其中r是径向距离，θ是倾角，phi是方位角。  
[搜索pygame.math.Vector3.as\_spheral的示例](https://github.com/search?q=pygame.math.Vector3.as_spheral&type=Code)

pygame.math.Vector3.from\_spherical() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#Vector3)

从球面坐标3元组设置x、y和z  
from\_spherical((r, theta, phi)) -> None  
从元组(r, theta, phi)设置x，y和z，其中r是径向距离，θ是倾斜角度，phi是方位角。  
[搜索pygame.math.Vector3.from\_spherical的示例](https://github.com/search?q=pygame.math.Vector3.from_spherical&type=Code)

pygame.math.enable\_swizzling() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#title)

全局允许对向量进行调配  
enable\_swizzling() -> None  
弃用：不再需要了。将在以后的版本中删除。 在调用disable\_swizzling()之前，可以对所有向量进行调配。默认情况下，禁用swizzling。  
[搜索pygame.math.enable\_swizzling的示例](https://github.com/search?q=pygame.math.enable_swizzling&type=Code)

pygame.math.disable\_swizzling() [↶](https://blog.csdn.net/Enderman_xiaohei/article/details/88396698#title)

全局禁用对向量进行调配  
disable\_swizzling() -> None  
弃用：不再需要了。将在以后的版本中删除。 禁用所有向量的调配，直到调用enable\_swizzling()。默认情况下，禁用swizzling。  
[搜索pygame.math.disable\_swizzling的示例](https://github.com/search?q=pygame.math.disable_swizzling&type=Code)

以上文档，自己翻译，可能有误，可参考：[pygame.math](https://www.pygame.org/docs/ref/math.html)