Xilinx Zynq FPGA, TI DSP, MCU 기반의 프로그래밍 및 회로 설계 전문가 과정 #55

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1.배운내용 복습.

Vector

벡터의 내적

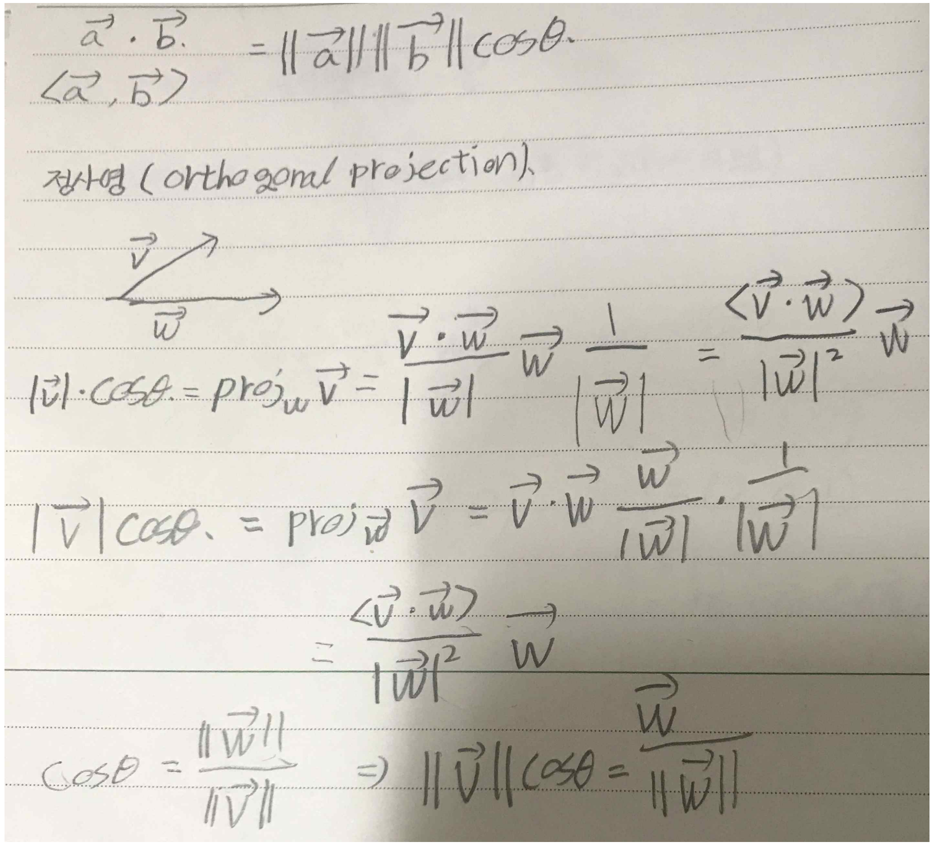
$$\overrightarrow{A} \bullet \overrightarrow{B} = a_x b_x + a_y b_y + a_z b_z$$

백터의 외적

$$A \times B = \begin{bmatrix} i & j & k \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{bmatrix}$$

$$(a_y b_z - a_z b_y)i + (a_z b_x - a_x b_z)j + (a_x b_y - a_y b_x)k$$

그람슈미트



$\vec{a}\cdot\vec{b}.$

$\langle\vec{a},\vec{b}\rangle \quad = \|\vec{a}\|\|\vec{b}\|\cos\theta.$

정사영 (orthogonal projection).

$\vec{v}$

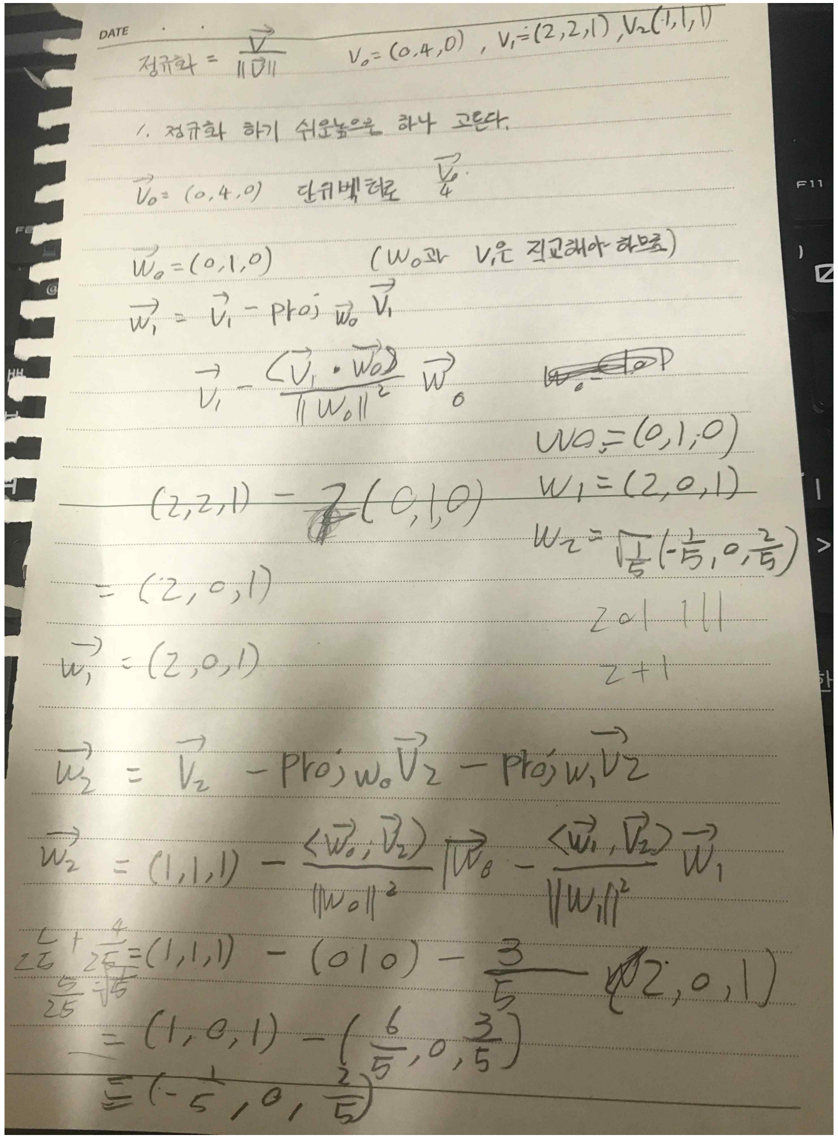
$\vec{w}$

$$|\vec{v}|\cdot\cos\theta. = proj_w\vec{v} = \frac{\vec{v}\cdot\vec{w}}{|\vec{w}|}\vec{w}\,\frac{1}{|\vec{w}|} = \frac{\langle\vec{v}\cdot\vec{w}\rangle}{|\vec{w}|^2}\vec{w}$$

$$|\vec{v}|\cos\theta. = proj_{\vec{w}}\vec{v} = \vec{v}\cdot\vec{w}\,\frac{\vec{w}}{|\vec{w}|}\cdot\frac{1}{|\vec{w}|}$$

$$= \frac{\langle\vec{v}\cdot\vec{w}\rangle}{|\vec{w}|^2}\vec{w}$$

$$\cos\theta = \frac{\|\vec{w}\|}{\|\vec{v}\|} \Rightarrow \|\vec{v}\|\cos\theta = \frac{\vec{w}}{\|\vec{w}\|}$$



DATE . .

정규화 = $\frac{\vec{v}}{\|\vec{v}\|}$ $V_0=(0,4,0)$, $V_1=(2,2,1)$, $V_2(1,1,1)$

1. 정규화 하기 쉬운놈으로 하나 고른다.

$\vec{V_0}=(0,4,0)$ 단위벡터로 $\frac{\vec{V_0}}{4}$

$\vec{W_0}=(0,1,0)$ (W_0과 V_1은 직교해야 하므로)

$\vec{w_1}=\vec{V_1}-\mathrm{proj}_{\vec{W_0}}\vec{V_1}$

$\vec{V_1}-\frac{\langle\vec{V_1}\cdot\vec{W_0}\rangle}{\|W_0\|^2}\vec{W_0}$

$W_0=(0,1,0)$

$W_1=(2,0,1)$

$W_2=\sqrt{\frac{1}{5}}(-\frac{1}{5},0,\frac{2}{5})$

~~$(2,2,1)-2(0,1,0)$~~

$=(2,0,1)$

2의 111

2+1

$\vec{w_1}=(2,0,1)$

$\vec{w_2}=\vec{V_2}-\mathrm{Proj}_{w_0}\vec{V_2}-\mathrm{Proj}_{w_1}\vec{V_2}$

$\vec{w_2}=(1,1,1)-\frac{\langle\vec{w_0},\vec{V_2}\rangle}{\|w_0\|^2}\vec{W_0}-\frac{\langle\vec{w_1},\vec{V_2}\rangle}{\|w_1\|^2}\vec{W_1}$

$\frac{1}{25}+\frac{4}{25}$ $\frac{5}{25}$ $\sqrt{5}$

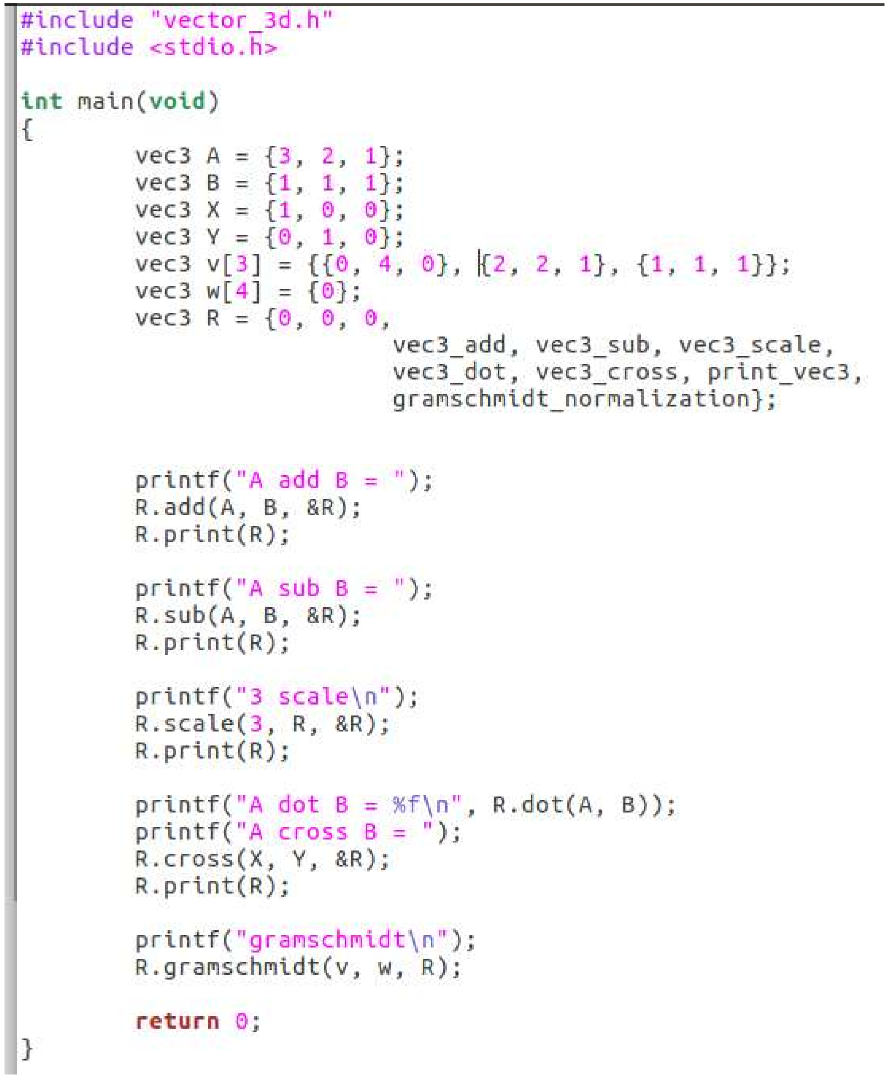
$=(1,1,1)-(010)-\frac{3}{5}(2,0,1)$

$=(1,0,1)-(\frac{6}{5},0,\frac{3}{5})$

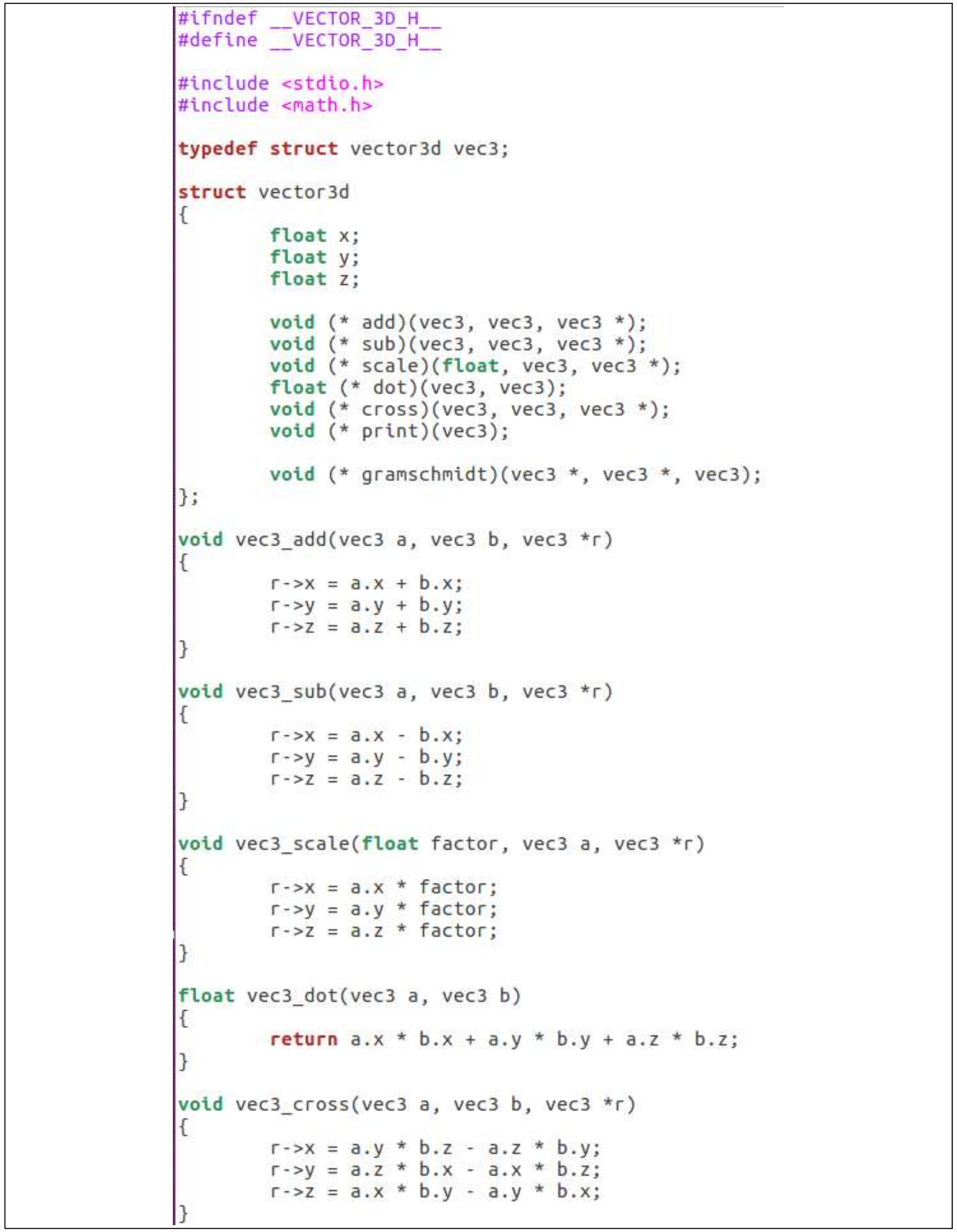
$=(-\frac{1}{5},0,\frac{2}{5})$

위의 풀이과정을 그대로 소스코드로 표현하면 다음과 같다.

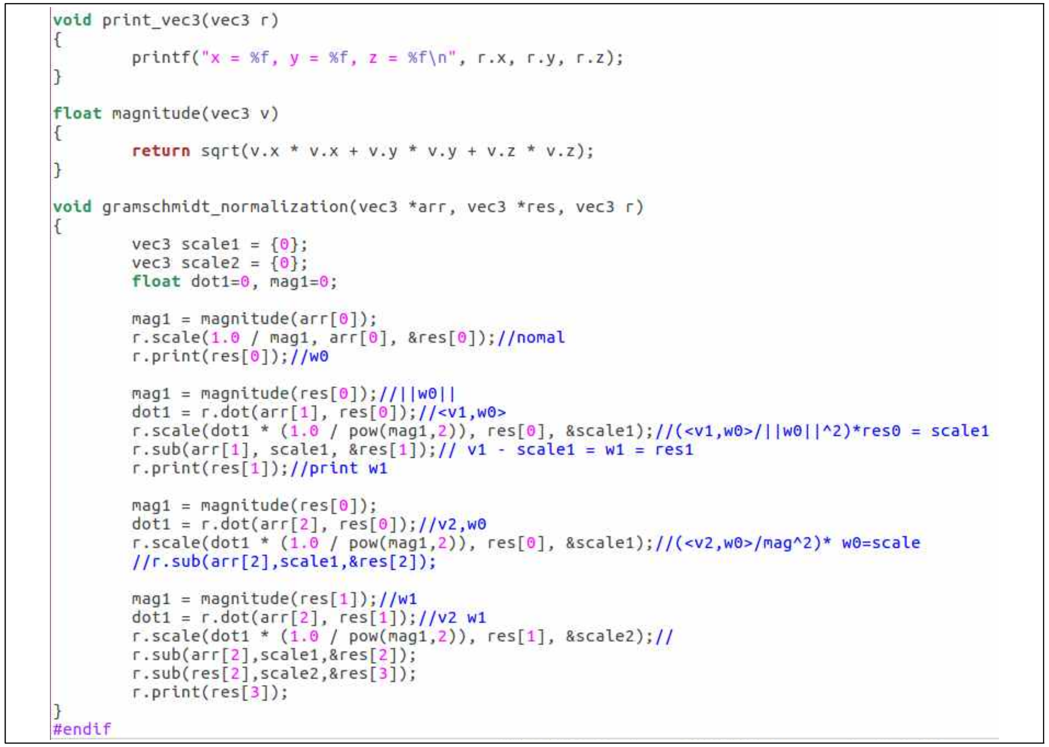
C언어로 C++ class구현



```c
#include "vector_3d.h"
#include <stdio.h>

int main(void)
{
        vec3 A = {3, 2, 1};
        vec3 B = {1, 1, 1};
        vec3 X = {1, 0, 0};
        vec3 Y = {0, 1, 0};
        vec3 v[3] = {{0, 4, 0}, {2, 2, 1}, {1, 1, 1}};
        vec3 w[4] = {0};
        vec3 R = {0, 0, 0,
                          vec3_add, vec3_sub, vec3_scale,
                          vec3_dot, vec3_cross, print_vec3,
                          gramschmidt_normalization};


        printf("A add B = ");
        R.add(A, B, &R);
        R.print(R);

        printf("A sub B = ");
        R.sub(A, B, &R);
        R.print(R);

        printf("3 scale\n");
        R.scale(3, R, &R);
        R.print(R);

        printf("A dot B = %f\n", R.dot(A, B));
        printf("A cross B = ");
        R.cross(X, Y, &R);
        R.print(R);

        printf("gramschmidt\n");
        R.gramschmidt(v, w, R);

        return 0;
}
```



```c
#ifndef __VECTOR_3D_H__
#define __VECTOR_3D_H__

#include <stdio.h>
#include <math.h>

typedef struct vector3d vec3;

struct vector3d
{
        float x;
        float y;
        float z;

        void (* add)(vec3, vec3, vec3 *);
        void (* sub)(vec3, vec3, vec3 *);
        void (* scale)(float, vec3, vec3 *);
        float (* dot)(vec3, vec3);
        void (* cross)(vec3, vec3, vec3 *);
        void (* print)(vec3);

        void (* gramschmidt)(vec3 *, vec3 *, vec3);
};

void vec3_add(vec3 a, vec3 b, vec3 *r)
{
        r->x = a.x + b.x;
        r->y = a.y + b.y;
        r->z = a.z + b.z;
}

void vec3_sub(vec3 a, vec3 b, vec3 *r)
{
        r->x = a.x - b.x;
        r->y = a.y - b.y;
        r->z = a.z - b.z;
}

void vec3_scale(float factor, vec3 a, vec3 *r)
{
        r->x = a.x * factor;
        r->y = a.y * factor;
        r->z = a.z * factor;
}

float vec3_dot(vec3 a, vec3 b)
{
        return a.x * b.x + a.y * b.y + a.z * b.z;
}

void vec3_cross(vec3 a, vec3 b, vec3 *r)
{
        r->x = a.y * b.z - a.z * b.y;
        r->y = a.z * b.x - a.x * b.z;
        r->z = a.x * b.y - a.y * b.x;
}
```



```c
void print_vec3(vec3 r)
{
        printf("x = %f, y = %f, z = %f\n", r.x, r.y, r.z);
}

float magnitude(vec3 v)
{
        return sqrt(v.x * v.x + v.y * v.y + v.z * v.z);
}

void gramschmidt_normalization(vec3 *arr, vec3 *res, vec3 r)
{
        vec3 scale1 = {0};
        vec3 scale2 = {0};
        float dot1=0, mag1=0;

        mag1 = magnitude(arr[0]);
        r.scale(1.0 / mag1, arr[0], &res[0]);//nomal
        r.print(res[0]);//w0

        mag1 = magnitude(res[0]);//||w0||
        dot1 = r.dot(arr[1], res[0]);//<v1,w0>
        r.scale(dot1 * (1.0 / pow(mag1,2)), res[0], &scale1);//(<v1,w0>/||w0||^2)*res0 = scale1
        r.sub(arr[1], scale1, &res[1]);// v1 - scale1 = w1 = res1
        r.print(res[1]);//print w1

        mag1 = magnitude(res[0]);
        dot1 = r.dot(arr[2], res[0]);//v2,w0
        r.scale(dot1 * (1.0 / pow(mag1,2)), res[0], &scale1);//(<v2,w0>/mag^2)* w0=scale
        //r.sub(arr[2],scale1,&res[2]);

        mag1 = magnitude(res[1]);//w1
        dot1 = r.dot(arr[2], res[1]);//v2 w1
        r.scale(dot1 * (1.0 / pow(mag1,2)), res[1], &scale2);//
        r.sub(arr[2],scale1,&res[2]);
        r.sub(res[2],scale2,&res[3]);
        r.print(res[3]);
}
#endif
```

C언어 C++class 구현 연습

```
#ifndef __VECTOR_3D_H__
#define __VECTOR_3D_H__

typedef struct vector3d
{
    float x;
    float y;
    float z;

    void (* add)(struct vector3d, struct vector3d, struct vector3d *);
    void (* sub)(struct vector3d, struct vector3d, struct vector3d *);
    void (* scale)(float, struct vector3d, struct vector3d *);
    void (* dot)(struct vector3d, struct vector3d, float*);
    void (* cross)(struct vector3d, struct vector3d,struct vector3d *);
} vec3;

void vec3_add(vec3 a, vec3 b, vec3 *r)
{
    r->x = a.x + b.x;
    r->y = a.y + b.y;
    r->z = a.z + b.z;
}

void vec3_sub(vec3 a, vec3 b, vec3 *r)
{
    r->x = a.x - b.x;
    r->y = a.y - b.y;
    r->z = a.z - b.z;
}
```

```
void vec3_scale(float num,vec3 a,vec3 *r)
{
    r->x= a.x * num;
    r->y= a.y * num;
    r->z= a.z * num;
}

void vec3_dot(vec3 a, vec3 b,float *scal)
{
    *scal = a.x*b.x + a.y*b.y + a.z*b.z;
}

void vec3_cross(vec3 a, vec3 b, vec3 *r)
{
    r->x = (a.y*b.z)-(a.z*b.y);
    r->y = -(a.x*b.z)+(a.z*b.x);
    r->z = (a.x*b.y)-(a.y*b.x);
}
#endif
```