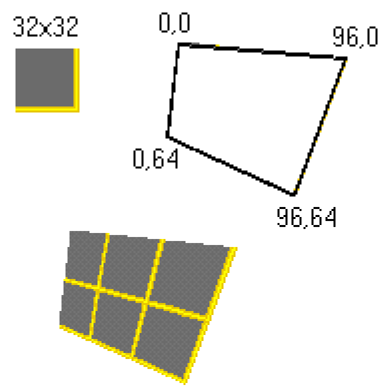
Vectors Part 2

By Alexei Novikov

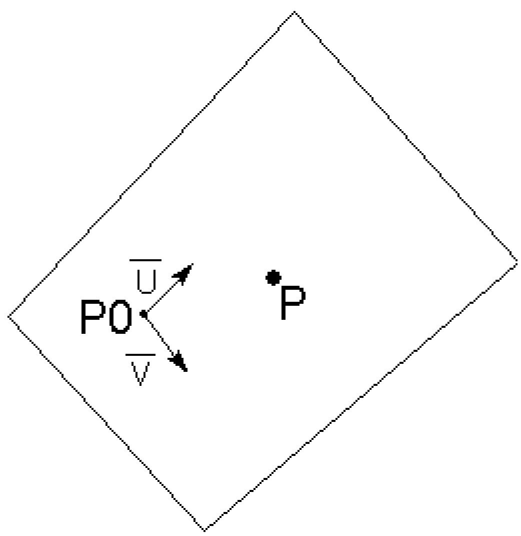
Since my last article about vectors had some success and I got a couple of "yes" answers to the question I asked at the end of it, here's the sequel.
I'd like to illustrate the use of **vector** math a practical example - texturing.

So, texturing. First, what texturing would mean here? It is a way to place a texture on a plane of surface. Basically the task it to calculate the texture coordinates given the texturing parameters - texture scale, orientation and perhaps flipped attribute (whether the texture is flipped or not). Texture coordinates in JK - U and V specify the point of the texture that corresponds to the current vertex. See at right how it works. Theoretically, you can define all kinds of texture distortions using texture coordinates. However, the texture in JK will not look right, will jiggle, etc. if your texture coordinates are not linear along the surface.

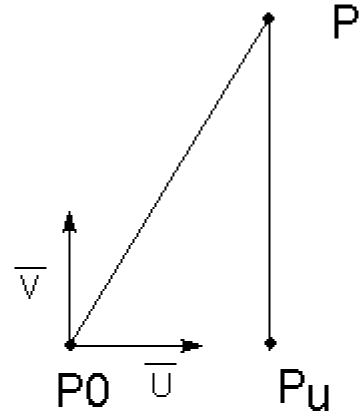


Basically what we need to do is define a 2D coordinate system on the surface plane. "Defining" mean - find the unit vectors (let's call them U and V) that represent 1 unit long vectors along coordinate axis. Then a point with coordinates U1 and V1 will have the following coordinate in 3D space:

P(U1,V1)=U*U1+V*V1



That gives a U,V->X,Y,Z coordinate conversion. But we actually mean the opposite X,Y,Z->U,V conversion. But it isn't much harder. Let's say the origin of our coordinate system is point P0. In U,V coordinates it will have coordinates U=0,V=0. Let's say it has coordinates (X0,Y0,Z0) in 3D space. The U and V coordinates of the point P will basically be the distance from point P to the line defined by **vector** U and point P0 and **vector** V and point P0, respectively. Does that ring a bell? It's the same problem as finding a distance from the point to the plane defined by a point and a **vector**. Or, looking at it another way, U coordinate will be the length of the side of the triangle P0,P,Pu (angle P-Pu-P0 is 90 degrees). Which would be cos(angle P,Pu,P0)*length(P-P0). Now remember the formula for dot product of two vectors?



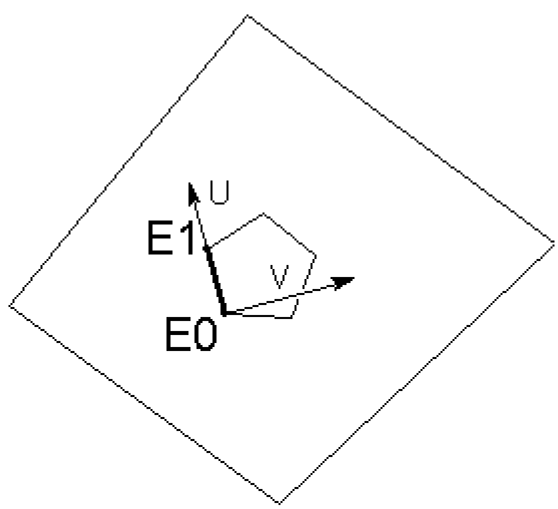
Dot(V1,V2)=|V1|*|V2|*cos(angle V1,V2)

For our vectors, considering U **vector** is 1 unit long, it's down to |P-P0|*cos(angle P-P0,U)

In other words, just what the doctor ordered. So, when we know U and V vectors for our UV coordinate system, here's the formula for conversion of X,Y,Z coordinates in 3D space to U,V coordinates on the plane:

u=dot(P-P0,U)
v=dot(P-P0,V)

Let's say we need to place a texture in a way so that it starts and goes along an edge of a surface and the texture scale is 320 pixel per JKU (normal JK scale). That would mean the U,V coordinates at the beginning of the edge will be (0,0) and at the end of the edge - (320*length(edge),0). In other words, the U **vector** goes along the edge. Let's say the beginning point of the edge is E0 and ending is E1. So, U **vector** will be:



U=(E1-E0)/length(E1-E0)

It's divided by the length of it because U must be normalized -1 unit long. The **vector** V must be perpendicular to the **vector** U and must lie on the plane of the surface. How would you find it? Simple - remember the cross product of vectors? V will simply be:

V=cross(U, surface normal)

See why? The cross product of any **vector** lying on the plane and surface normal is going to lie on the plane (since any **vector** perpendicular to the normal is going to lie on the plane. I'll let you figure out why. Hint - remember that distance from point to surface problem again). And the cross product of two vectors is perpendicular to both. Thus it's perpendicular to U. Here, both things that define **vector** V.
Now that we have U and V vectors, using out formula for X,Y,Z->U,V conversion for each vertex of the surface we can calculate u,v coorddinates as:

u=dot(U,(x,y,z)-e0)*320
v=dot(V,(x,y,z)-e0)*320

That's it, problem solved.

However, it isn't the last of vectors. Stay tuned for more.
Alex.