Anti-aliasing with sub-pixel sampling.

Aliasing is an effect in which edges that should appear smooth actually appear jagged because of the finite size of pixels. One approach to anti-aliasing is to artificially induce an intensity gradient near any edge. One way to do this is via *jittered sampling*. (jitter: irregular, random movement)

In this approach, the *world* coordinate space is partitioned into a collection of non-overlapping squares in which the actual pixel associated at the square is located at the center. Multiple rays are fired at each pixel with the direction of the ray randomly jittered in a way that ensures in passes through the proper square.

The easiest way to do this is via a simple modification to *map_pix_to _world.*

```
void map_pix_to_world(proj_t *v, int x, int y, double *world){
   double rx;  // These values MUST BE DOUBLE -- not int!!!!
   double ry;
   rx = randpix(x);
   ry = randpix(y);
   *(world + 0) = (1.0 * rx / v->win_x_size) * v->world_x_size;
   *(world + 1) = (1.0 * ry / v->win_y_size) * v->world_y_size;
   *(world + 2) = 0.0;
}
```

Where you come with the randpix function....

The random and srandom functions may be useful

To perform anti-aliasing we insert the following code in the *make_pixel()* function *image.c.*

```
intensity[0] = intensity[1] = intensity[2] = 0.0;

for (k = 0; k < AA_SAMPLES; k++){
   map_pix_to_world(proj, j, i, world);
   vl_diff3(proj->view_pt, world, dir);
   vl_unitvec3(dir, dir);
   ray_trace(lst, proj->view_pt, dir, intensity, 0.0, NULL);
}
vl_scale3(255.0 / AA_SAMPLES, intensity, intensity);
```

now clamp intensity to the range [0, 255] and assign to pixmap