```
#include "proctext.h"

#define BRICKWIDTH      0.25
#define BRICKHEIGHT     0.08
#define MORTARTHICKNESS 0.01

#define BMWIDTH         (BRICKWIDTH+MORTARTHICKNESS)
#define BMHEIGHT        (BRICKHEIGHT+MORTARTHICKNESS)
#define MWF             (MORTARTHICKNESS*0.5/BMWIDTH)
#define MHF             (MORTARTHICKNESS*0.5/BMHEIGHT)

surface
brickbump(
   uniform float Ka = 1;
   uniform float Kd = 1;
   uniform color Cbrick = color (0.5, 0.15, 0.14);
   uniform color Cmortar = color (0.5, 0.5, 0.5);
    )
{  color Ct;
   point Nf;
   float ss, tt, sbrick, tbrick, w, h;
   float scoord = s;
   float tcoord = t;
   float sbump, tbump, stbump;

   Nf = normalize(faceforward(N, I));

   ss = scoord / BMWIDTH;
   tt = tcoord / BMHEIGHT;

   if (mod(tt*0.5,1) > 0.5)
      ss += 0.5;  /* shift alternate rows */
   sbrick = floor(ss); /* which brick? */
   tbrick = floor(tt); /* which brick? */
   ss -= sbrick;
   tt -= tbrick;
   w = step(MWF,ss) - step(1-MWF,ss);
   h = step(MHF,tt) - step(1-MHF,tt);

   Ct = mix(Cmortar, Cbrick, w*h);

   /* compute bump-mapping function for mortar grooves */
   sbump = smoothstep(0,MWF,ss) - smoothstep(1-MWF,1,ss);
   tbump = smoothstep(0,MHF,tt) - smoothstep(1-MHF,1,tt);
   stbump = sbump * tbump;

   /* compute shading normal - move surface and cross the partial derivatives to get new normal */
   Nf = calculatenormal(P + normalize(N) * stbump);
   Nf = normalize(faceforward(Nf, I));

   /* diffuse reflection model */
   Oi = Os;
   Ci = Os * Ct * (Ka * ambient() + Kd * diffuse(Nf));
}
```