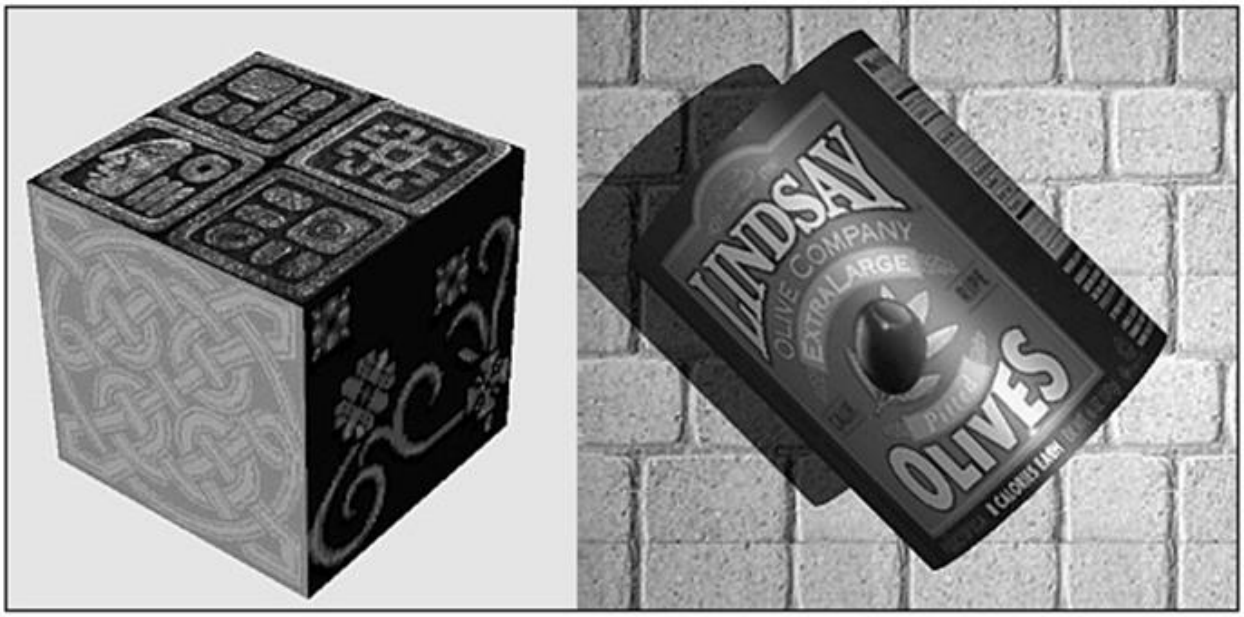
بسمه تعالی

بافت

اضافه کردن بافت می تواند به واقعی تر نمودن شکل کمک نماید.



منابع بافت

بطور کلی بافتها آرایه ای مستطیلی از داده هستند، مثل رنگ، روشنائی. اغلب از تصاویر ولی می توان تولید کرد

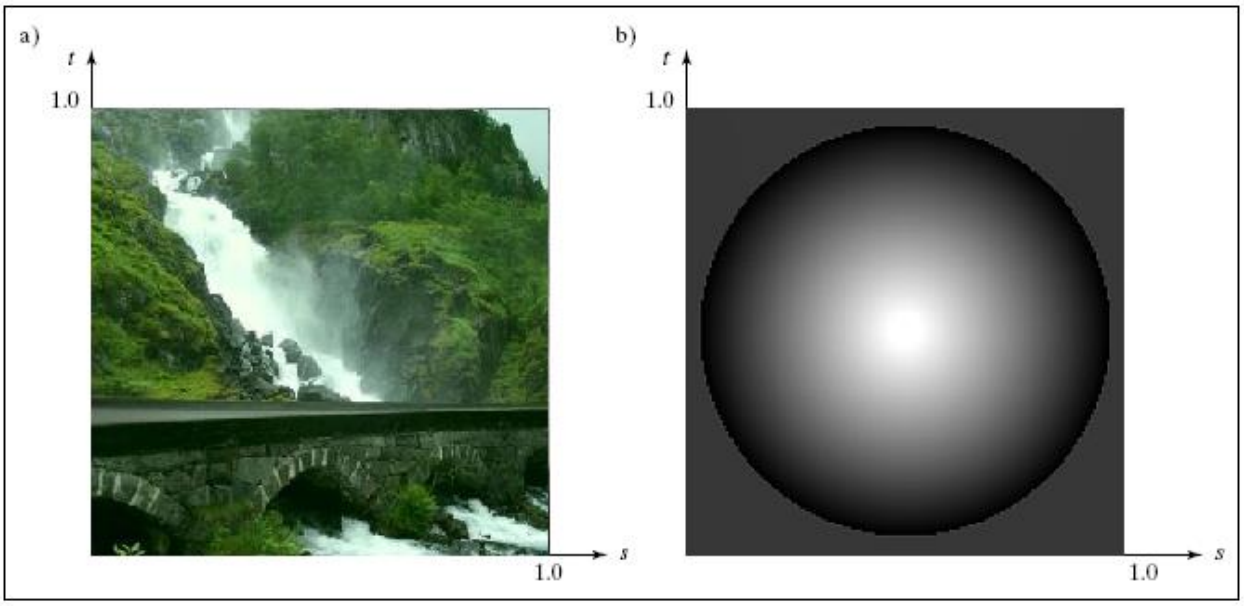
بصورت یک آرایه دوبعدی txtr[r][c] از مقادیر رنگ. هر عضو تکسل (texel) نامیده می شود.

بافت باید بر روی یک ناحیه که ممکن است مستطیلی نیز نباشد نگاشت شود.

برای نگاشت بافت به مراحل زیر نیاز است:

- ایجاد یک شئ بافت و تخصیص یک بافت به آن
- مشخص کردن اینکه چگونه بافت به هر پیکسل اعمال شود
- فعال کردن نگاشت بافت
- رسم صحنه، با مهیا کردن مختصات هندسی و بافت

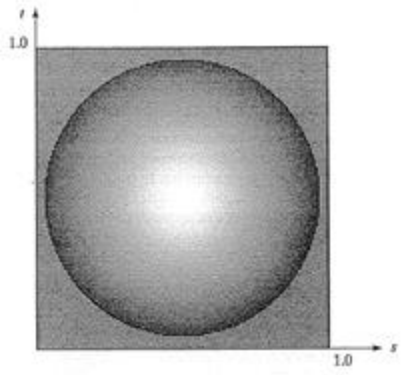
یک بافت یک تابع texture(s,t) است که یک رنگ یا شدت برای هر مقدار s و t بین ۰ و ۱ ایجاد می کند.



شکل ۱

ایجاد بافت توسط روال

مثال: مرکز (s,t)=(0.5,0.5)

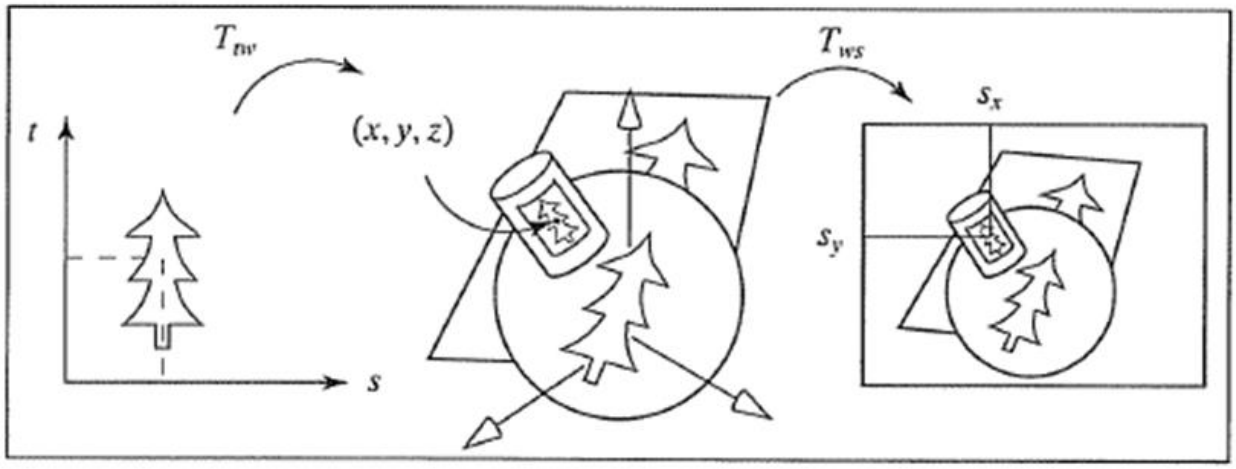


فاصله از مرکز:

$$\sqrt{(s-0.5)^2+(t-0.5)^2}$$

```
float fakeSphere (float s, float t)
{
  float r = sqrt((s-0.5)*(s-0.5)+(t-0.5)*(t-0.5));
/* distance from sphere's center to (s,t). */
  if(r < 0.3) return 1 - r/0.3;
/* sphere intensity brightest at the center; dark along the edges */
  else return 0.2; // dark background
}*/
```

با داشتن تابع بافت، مرحلهٔ بعدی نگاشت مناسب بر روی سطح مطلوب، و سپس نگریستن به آن با دوربین می باشد.



$$(sx, sy) = T_{ws}(T_{tw}(s^*, t^*)).$$

هنگام نمایش:

نزدیکترین سطح که در (sx,sy) دیده می شود کدام است؟

نیاز به حل حذف سطح مخفی

نقطهٔ متناظر (x,y,z) به (sx,sy) کدام است؟

کدام نقطهٔ بافت (s,t) متناظر با (x,y,z) است؟

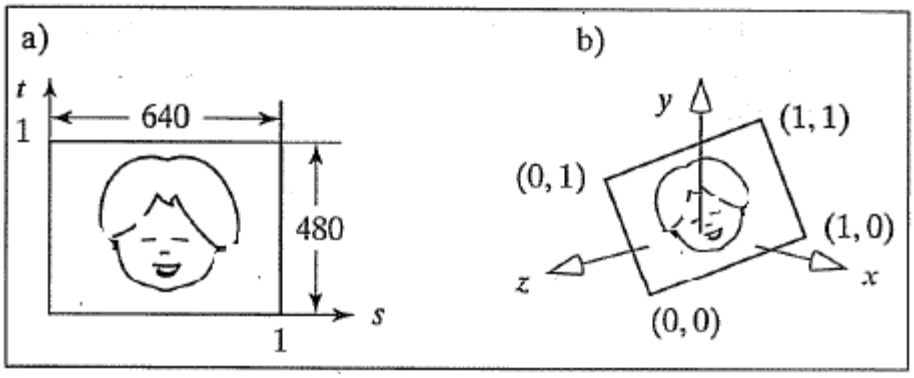
چسباندن بافت بر روی سطح صاف

هر نقطه در فضای بافت Pi(s,t) باید به یک رأس وابسته شود.

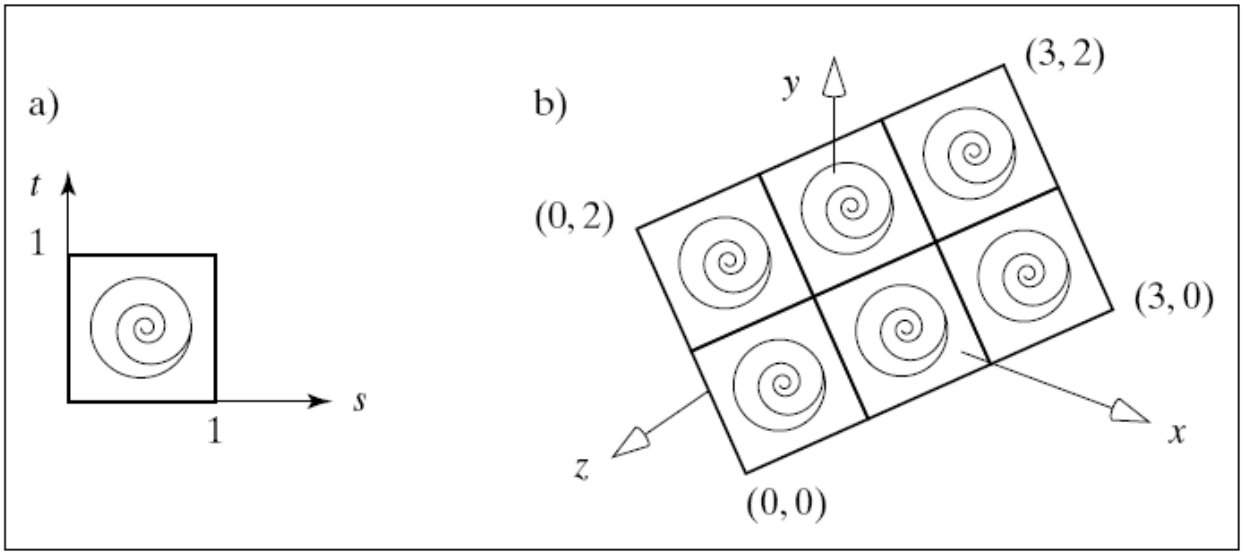
توسط دستور glTexCoord2f(s,t)

```
glBegin(GL_QUADS); // define a quadrilateral and position our
                   // texture on it
     glTexCoord2f(0.0, 0.0); glVertex3f(1.0, 2.5, 1.5);
     glTexCoord2f(0.0, 0.6); glVertex3f(1.0, 3.7, 1.5);
     glTexCoord2f(0.8, 0.6); glVertex3f(2.0, 3.7, 1.5);
     glTexCoord2f(0.8, 0.0); glVertex3f(2.0, 2.5, 1.5);
glEnd();
```

اگر هم شکل باشند بدون اعوجاج خواهد بود.



اگر (s,t)=(2.5,3.88) آنگاه همان تکسل(s,t)=(0.5,0.88)



اضافه کردن مختصات بافت به شئ با شبکه چندضلعی

یک شبکه شامل لیست رأس، عمود، و وجه

به آن لیست مختصات بافت وابسته به هر رأس را اضافه می کنیم.

تعریف کلاس زیر:

```
class TxtrCoord { public: float s, t;·};
```

یک آرایه از آن ایجاد کرده و در آن همهٔ جفت مختصاتهای لازم برای شبکه را نگاهداری می کنیم.

دو روش:

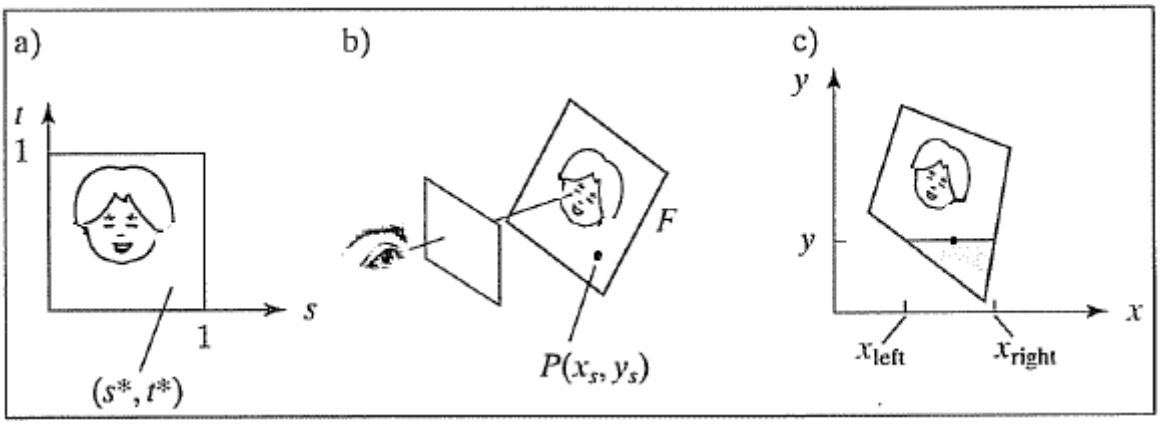
روش اول: شبکه شامل تعدادی وجه صاف و هر وجه بافت خود را دارد. هر وجه یک عمود و لیست بافت.

بنابر این برای هر وجه داریم: تعداد رئوس وجه، اندیس عمود، لیست اندیس رئوس، و لیست اندیس بافت

روش دوم: شبکه یک شئ نرم را مدل می کند. یک بافت بر روی همه (یا بخشی از آن) نگاشت می شود.

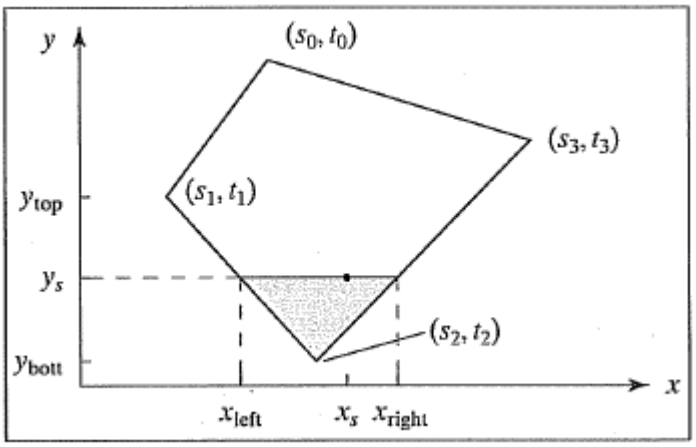
نمایش بافت

همانند سایه زنی گوراد برای هر پیکسل باید مختصات بافت (s,t) متناظر با آن را تعیین نمود. همانگونه که در شکل ۲ نشان داده شده خط پیمایشی y در حال پر شدن از x_{left} تا x_{right} می باشد. برای هر x در طول خط پیمایشی باید مکان صحیح (P(x,y) در شکل) محاسبه شده و از روی آن مکان صحیح (s*,t*) بر روی بافت مشخص شود.



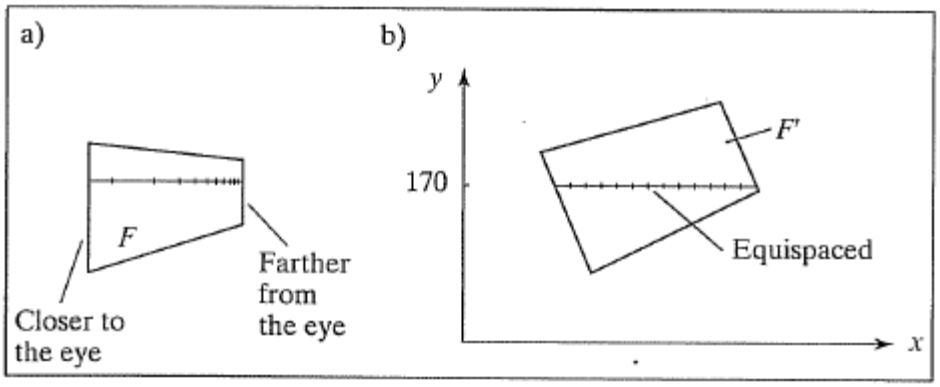
شکل ۲

علاقمند هستیم که برای هر خط پیمایشی مقادیر (sleft,tleft) و (sright, tright) را به سرعت یافته و در طول خط پیمایشی این مقادیر را درون یابی نمائیم.

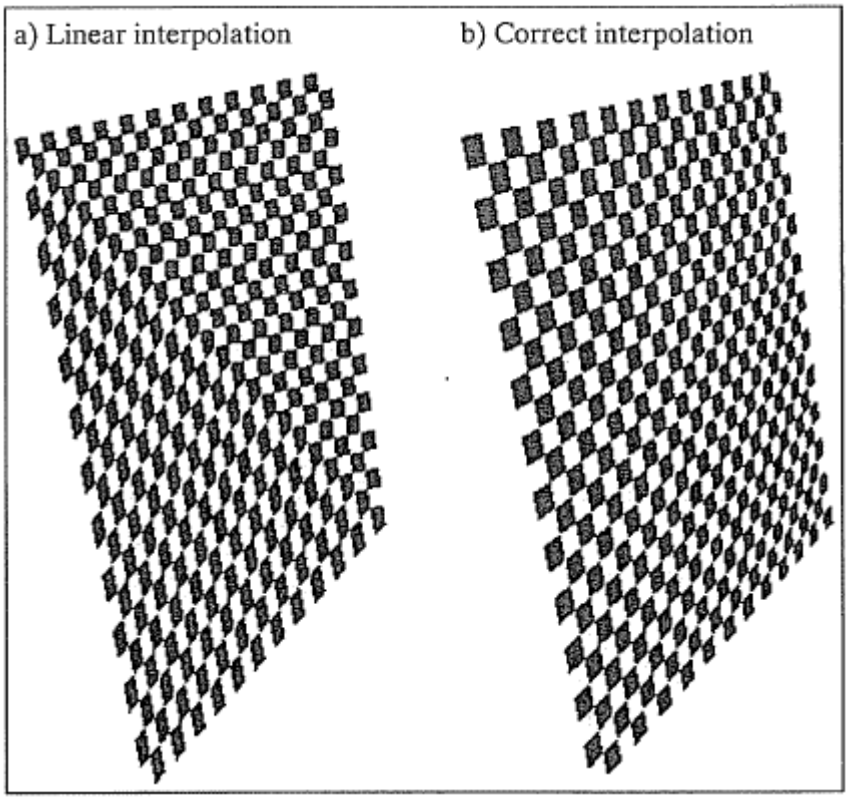


شکل ۳

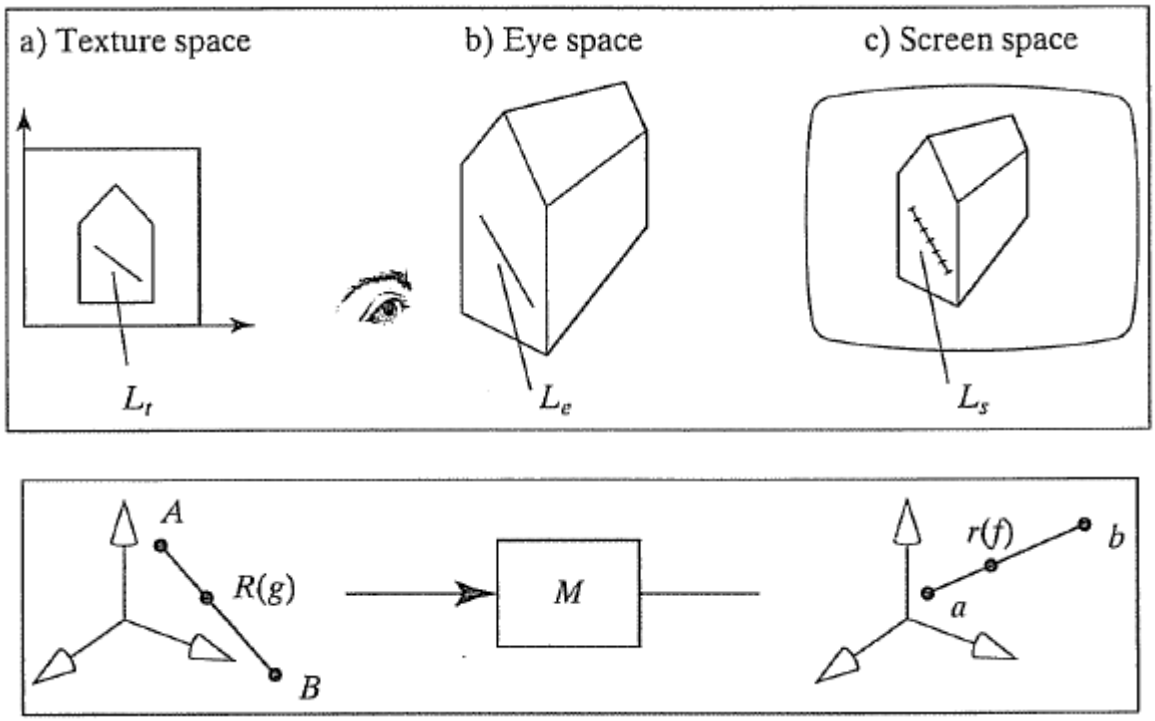
ولی بایستی مراقب باشیم: افزایش ساده از sleft به sright چنانچه در طول خط پیمایشی y از xleft به xright حرکت می کنیم کار نمی کند، چرا که قدمهای برابر در طول وجه افکنش شده متناظر با قدمهای برابر در راستای وجه سه بعدی نیست. چنانکه در شکل ٤ نشان داده شده است.



شکل ٤


a) Linear interpolation
b) Correct interpolation

شکل ۵


a) Texture space
b) Eye space
c) Screen space
L_t
L_e
L_s
A
R(g)
B
M
r(f)
b
a

شکل ۶

شکل ٦ خط AB در سه بعد را نشان می دهد که به خط ab در سه بعد توسط ماتریس M (که می تواند نمایانگر یک تبدیل مستوی یا بطور کلی تر یک تبدیل پرسپکتیو باشد) تبدیل شده است. نقطهٔ A به a و B به b تبدیل می شود. نقطهٔ R(g) که به نسبت g بین A و B قرار گرفته را در نظر بگیرید. این نقطه به نقطه ای همانند r(f) نگاشت می شود که به نسبت f بین a و b قرار گرفته است. نسبتهای g و f بطور کلی برابر نیستند. سؤال این است که چنانکه f بین ۰ تا ۱ تغییر می نماید g چگونه تغییر می نماید؟

نمایش همگن a عبارت است از: $\tilde{a} = (a_1, a2, a3, a4)$ و داریم:

$$a = \left(\frac{a_1}{a_4}, \frac{a_2}{a_4}, \frac{a_3}{a_4}\right)$$

M نقطهٔ *A=(A1,A2,A3)* را به *a* نگاشت می کند.

$$\tilde{a} = M(A, 1)^T$$

که $(A,1)^T$ یک بردار ستونی با اعضای *A1*، *A2*، *A3*، و ۱ می باشد. بطور مشابه:

$$\tilde{b} = M(B, 1)^T.$$

داریم:

$$R(g) = \text{lerp}(A, B, g),$$

$$lerp(A, B, g) = A + (B - A)g$$

$$M(\text{lerp}(A, B, g), 1)^T =$$

$$\text{lerp}(\tilde{a}, \tilde{b}, g) = (\text{lerp}(a_1, b_1, g), \text{lerp}(a_2, b_2, g), \text{lerp}(a_3, b_3, g), \text{lerp}(a_4, b_4, g)).$$

$$= \tilde{r}(f)$$

از رابطهٔ فوق مؤلفهٔ اوّل r *برابر خواهد بود با:*

$$r_1(f) = \frac{\text{lerp}(a_1, b_1, g)}{\text{lerp}(a_4, b_4, g)}$$

ولی می دانیم

$$r(f) = \text{lerp}(a, b, f)$$

بنابر این:

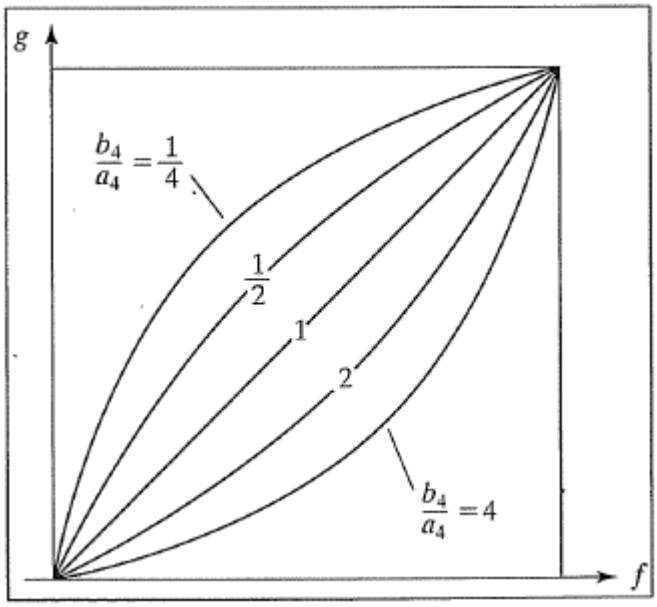
$$r_1(f) = \text{lerp}\left(\frac{a_1}{a_4}, \frac{b_1}{b_4}, f\right)$$

از برابر قرار دادن آنها خواهیم داشت:

$$\frac{lerp(a_1, b_1, g)}{lerp(a_4, b_4, g)} = lerp\left(\frac{a_1}{a_4}, \frac{b_1}{b_4}, f\right)$$

پس از مقداری محاسبات:

$$g = \frac{f}{\text{lerp}\left(\frac{b_4}{a_4}, 1, f\right)}$$



شکل ۷ چگونگي وابستگي g به f.

می دانیم:

$$R(g) = A(1 - g) + Bg$$

با قرار دادن مقدار g در این رابطه بدست خواهیم آورد:

$$R_j = \frac{\text{lerp}\left(\frac{A_1}{a_4}, \frac{B_1}{b_4}, f\right)}{\text{lerp}\left(\frac{1}{a_4}, \frac{1}{b_4}, f\right)}$$

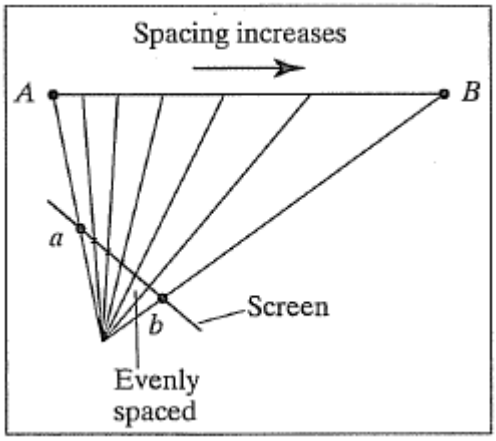
در حالت تبدیل مستوی $a4$ و $b4$ برابر ۱ هستند. گامهای برابر در طول ab همانند گامهای برابر در طول AB g و f برابر می شوند. ولی در تبدیل پرسپکتیو همواره اینگونه نیست. شکل بنیادی ماتریس M برای تبدیل پرسپکتیو بصورت زیر می باشد:

$$M = \begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & c & d \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

پس از محاسبهٔ M(A,1)T *مقدار آن برابر خواهد بود با:*

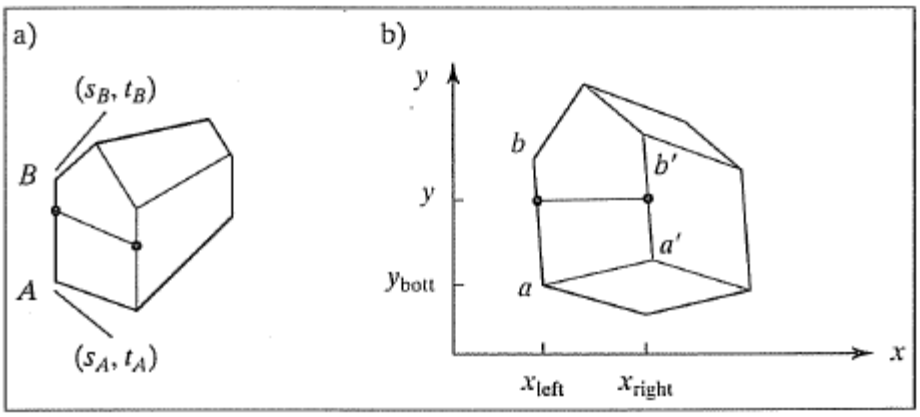
$$\tilde{a} = (NA_1, NA_2, cA_3 + d, -A_3)$$

همانگونه که مشاهده می شود $a4=-A3$ که با مقدار عمق نقطه بر روی محور z در دستگاه مختصات دوربین مرتبط است. بنابر این $a4$ و $b4$ در ارتباط با عمق دو نقطهٔ A و B هستند. اگر A و B دارای یک عمق باشند یعنی در صفحه ای به موازات صفحهٔ دید دوربین باشند هیچگونه اعواج پرسپکتیو وجود نخواهد داشت و بنابر این g و f برابرند. هنگامی که بطور مثال A به چشم نزدیکتر از B باشد همانند شکل زیر $a4<b4$ و چنانکه f رشد می کند g رشد بیشتری خواهد داشت.



شکل ۸ مقادیر a4 و b4 وابسته به عمق نقاط هستند.

نمایش افزایشی



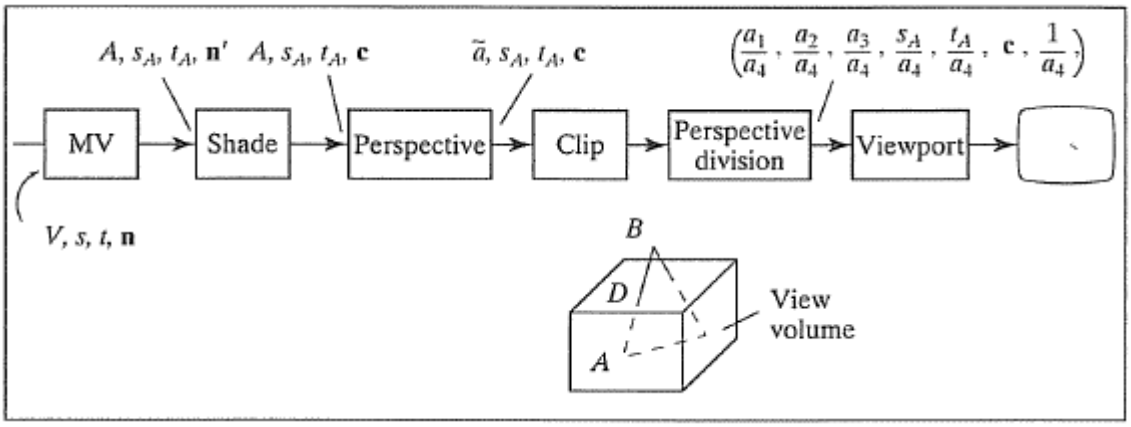
شکل ۹

$$S_{left}(y) = \frac{\text{lerp}\left(\frac{S_A}{a_4}, \frac{S_B}{b_4}, f\right)}{\text{lerp}\left(\frac{1}{a_4}, \frac{1}{b_4}, f\right)}$$

$$f = (y - y_{bott})/(y_{top} - y_{bott}))$$

مقادیر $s_A/a_4, s_B/b_4, t_A/a_4, t_B/b_4, 1/a_4,$ and $1/b_4,$ را ذخیره می کنیم. مقادیر صورت و مخرج بصورت افزایشی قابل دستیابی هستند. ولی تقسیم هنوز لازم است. زوج (*sright,tright*) هم به همین صورت محاسبه می شوند. پس از یافتن (*sleft,tleft*) و (*sright,tright*) مقادیر (*s,t*) برای خط پیمایشی *y* قابل محاسبه هستند.

اصلاح خط لولهٔ *OpenGL* در شکل ۱۰ نشان داده شده است. هر رأس *V* علاوه بر عمود *n* به جفت بافت *(s,t)* وابسته می شود. رأس بوسیلهٔ ماتریس *modelview* به رأس *A=(A1,A2,A3)* در دستگاه چشم تبدیل می شود (عمود *n* به عمود *n'* تبدیل می شود). محاسبات سایه زنی توسط عمود *n'* انجام شده و رنگ *c=(cr,cg,cb)* ایجاد می شود. مختصات بافت *(sA,tA)* که مشابه همان *(s,t)* می باشند هنوز به *A* وابسته است. رأس *A* تحت تبدیل پرسپکتیو به $\tilde{a}$ *=(a1,a2,a3,a4)* تبدیل می شود. مختصات بافت و رنگ *c* تغییر نمی کند. حال عمل برش در مقابل حجم دید انجام می گیرد. این کار ممکن است باعث شود که برخی از رأسها حذف شده و رأسهای جدیدی ایجاد شوند که در این حالت باید رنگ و مختصات بافت رئوس جدید محاسبه شوند. پس از آن عمل تقسیم پرسپکتیو انجام می شود. پس از تقسیم سه مقدار اوّل *(x,y,z)=(a1/a4,a2/a4,a3/a4)* مکان نقطه را در دستگاه مختصات عادی شده نمایش می دهند که مؤلفهٔ سوّم شبه عمق می باشد. این مقادیر به بندردید تبدیل شده و نمایش داده می شوند.



شکل ۱۰

برای تکرار الگو در جهت *s*

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
```

در جهت *t*، *WRAP_T* به جای *WRAP_S* استفاده می کنیم. این حالت پیش فرض *OpenGL* می باشد.

می توان از *GL_CLAMP* استفاده نمود.

استفاده از مقادیر بافت

در ساده ترین حالت: استفاده از مقدار بافت برای مقدار شدت آن نقطه $I=texture(s,t)$. اگر رنگی باشد: $I_r=texture_r(s,t)$ به همین ترتیب برای سبز و قرمز.

دستور:

```
glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE, GL_REPLACE);
```

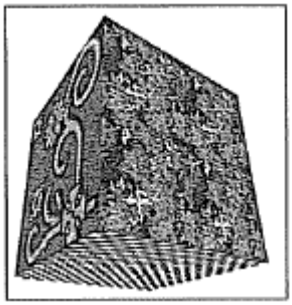
حالت دیگر: ترکیب با مقدار شدت آن نقطه

$$I = \text{texture}(s,t)[I_a\rho_a + I_d\rho_d \times lambert] + I_{sp}\rho_s \times phong^f$$

```
glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE, GL_MODULATE);
```

مثال:

رسم یک مکعب که بر روی هر وجه آن یک بافت قرار دارد و دوران می کند.



```
// <… the usual includes …>
#include "RGBpixmap.h"
//######################## GLOBALS ########################
RGBpixmap pix[6]; // make six (empty) pixmaps
float xSpeed = 0, ySpeed = 0, xAngle = 0.0, yAngle = 0.0;
//<<<<<<<<<<<<<<<<<<<<<<<<<<<< myinit >>>>>>>>>>>>>>>>>>>>>>>>>>>>.
void myInit(void)
{
     glClearColor(1.0f,1.0f,1.0f,1.0f); // background is white
     glEnable(GL_DEPTH_TEST);
     glEnable(GL_TEXTURE_2D);

     pix[0].makeCheckerboard();         // make pixmap procedurally
     pix[0].setTexture(2001);               // create texture
     pix[1].readBMPFile("Mandrill.bmp");  // make pixmap from image
     pix[1].setTexture(2002);  // create texture
     //< …similarly for other four textures …>

     glViewport(0, 0, 640, 480); // set up the viewing system
     glMatrixMode(GL_PROJECTION);
     glLoadIdentity();
     gluPerspective(60.0, 640.0/ 480, 1.0, 30.0); // set camera shape
     glMatrixMode(GL_MODELVIEW);
     glLoadIdentity();
     glTranslated(0.0, 0.0, -4); // move camera back
}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<< display >>>>>>>>>>>>>>>>>>>>>>
void display(void)
{
     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
     glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
     glPushMatrix();
     glRotated(xAngle, 1.0,0.0,0.0); glRotated(yAngle, 0.0,1.0,0.0); // rotate

     glBindTexture(GL_TEXTURE_2D,2001); // top face: 'fake' checkerboard
     glBegin(GL_QUADS);
     glTexCoord2f(-1.0, -1.0); glVertex3f(-1.0f, 1.0f, -1.0f);
     glTexCoord2f(-1.0, 2.0); glVertex3f(-1.0f, 1.0f, 1.0f);
     glTexCoord2f(2.0, 2.0); glVertex3f( 1.0f, 1.0f, 1.0f);
     glTexCoord2f(2.0, -1.0); glVertex3f( 1.0f, 1.0f, -1.0f);
     glEnd();

     glBindTexture(GL_TEXTURE_2D,2002);   // right face: mandrill
     glBegin(GL_QUADS);
     glTexCoord2f(0.0, 0.0); glVertex3f(1.0f, -1.0f,  1.0f);
     glTexCoord2f(0.0, 2.0); glVertex3f(1.0f, -1.0f, -1.0f);
     glTexCoord2f(2.0, 2.0); glVertex3f(1.0f,  1.0f, -1.0f);
     glTexCoord2f(2.0, 0.0); glVertex3f(1.0f,  1.0f,  1.0f);
     glEnd();
```

```
        // << … similarly for other four faces …>
        glFlush();
        glPopMatrix();
        glutSwapBuffers();
}
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<< spinner >>>>>>>>>>>>>>>>>>>>
void spinner(void)
{ // alter angles by small amount
        xAngle += xSpeed;  yAngle += ySpeed;
        glutPostRedisplay();
}
//<<<<<<<<<<<<<<<<<<<<<<< main >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
void main(int argc, char **argv)
{

        glutInit(&argc, argv);          // initialize the toolkit
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); // set the display mode
        glutInitWindowSize(600, 800);     // set the window size
        glutInitWindowPosition(100, 150); // set the window position on the screen
        glutCreateWindow("Rotating Cube Demo"); // open the screen window
        glutDisplayFunc(display);     // register the redraw function
        myInit();
        glutIdleFunc(spinner);
        glutMainLoop();                 // go into a perpetual loop
}
```

```
class RGB{ // holds a color triple - each with 256 possible
           intensities
    public: unsigned char r,g,b;
};

class RGBpixmap{
  public:
      int nRows, nCols; // dimensions of the pixmap
      RGB* pixel;        // array of pixels
      int readBMPFile(char * fname);  // read BMP file into this
                                         pixmap
      void makeCheckerboard();
      void setTexture(GLuint textureName);
};
```

پویا نمائی

برای ایجاد پویانمائی ، احتیاج داریم که ابتدا تغییری[1] در صحنه ایجاد کرده و سپس صحنه را دوباره رسم کنیم و این کار را دائماً تکرار کنیم. این را می توانیم خودمان انجام دهیم ولی *OpenGL* مقداری کار را آسانتر کرده است. توسط روال ***glutIdleFunc()*** می توانیم روالی را ثبت کنیم که

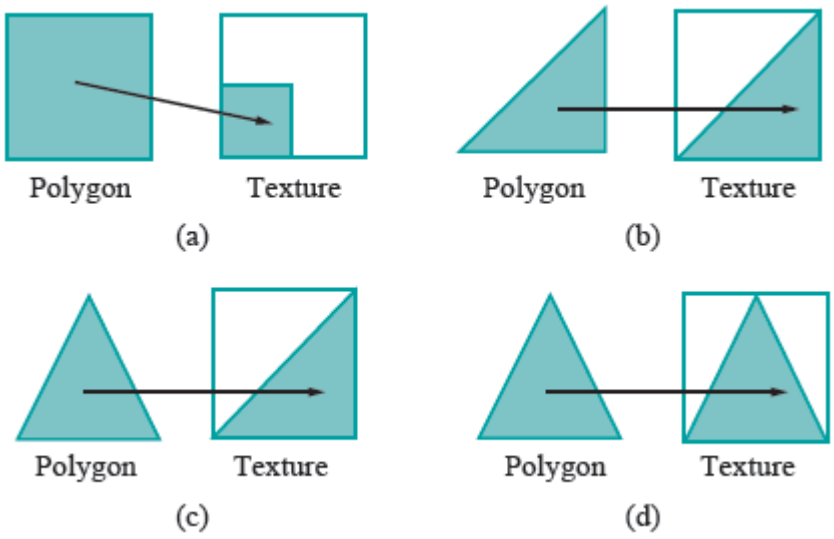
[1] animation

در هنگامی که *OpenGL* در حال پاسخ دادن به حادثه ای نیست صدا زده می شود، همانند روال *spinner()* در مثال فوق. در این روال تغییراتی را که لازم است قبل از نمایش دوبارهٔ صحنه ایجاد شود را به انجام می رسانیم و پس از آن دوباره روال نمایش را فراخوانی می کنیم.

برای نمایش نرمتر صحنه نیز بهتر است از دومیانگیر استفاده شود. در حالی که یک میانگیر برای نمایش استفاده می شود، میانگیر دیگری در حال نوشته شدن می باشد. پس از اتمام نوشته شدن بر روی این میانگیر، جای دو میانگیر را تعویض می کنیم. عمل تعویض جای دو میانگیر از دستور *glutSwapBuffers()* استفاده می شود. در ابتدا نیز باید اعلام کنیم که می خواهیم از دو میانگیر استفاده نمائیم. این کار توسط دستور زیر انجام می شود:

```
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
```

یک بافت در فضای بافت در هر بعد بین ۰ تا ۱ تعریف می شود ولی همانگونه که در شکل ۱۱ نشان داده شده است می توان بخشی از آن که مورد نیاز است را بر روی چندضلعی نگاشت کرد.

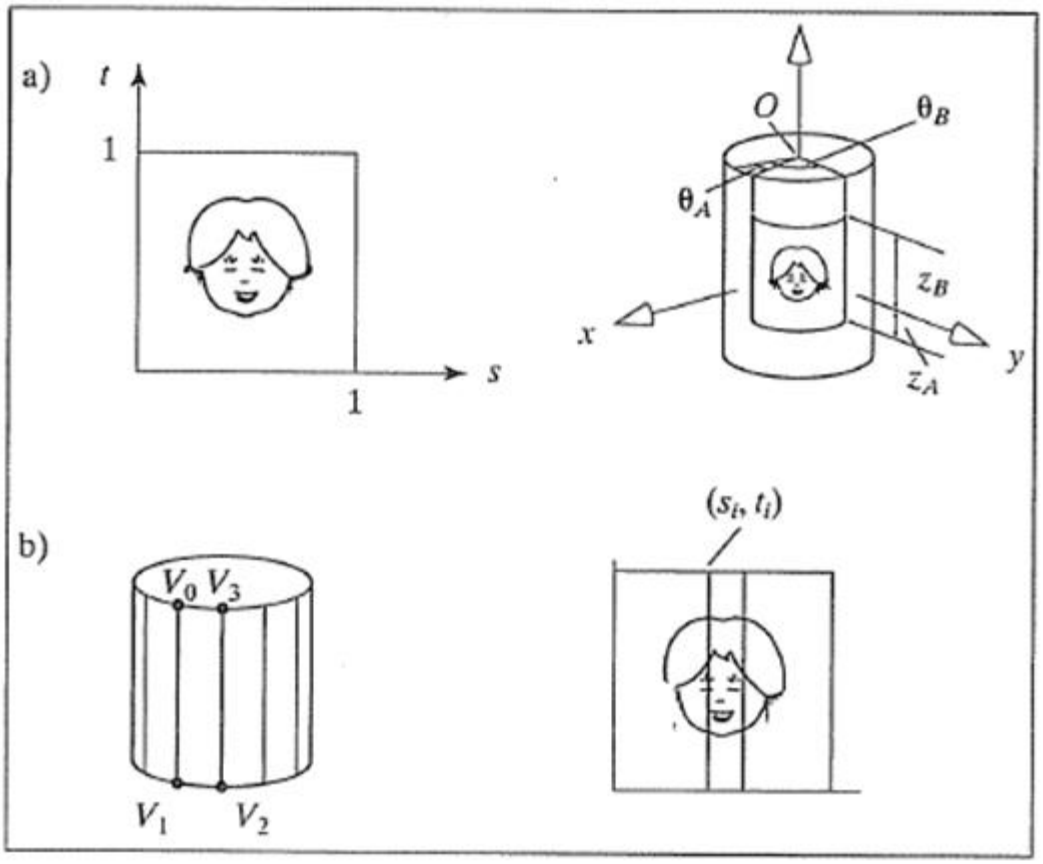


شکل ۱۱ برخی از روشهای نگاشت یک بافت بر روی چند ضلعی *[S.Guha,2015]*

گذاردن بافت بر روی اشیاء انحنادار

فرض: شئ با شبکهٔ چندضلعی مدل شده

ابتدا حول استوانهٔ قائم از θa تا θb و از za تا zb.



$$s = \frac{\theta - \theta_a}{\theta_b - \theta_a}, \qquad t = \frac{z - z_a}{z_b - z_a}$$

اگر N وجه حول استوانه باشند

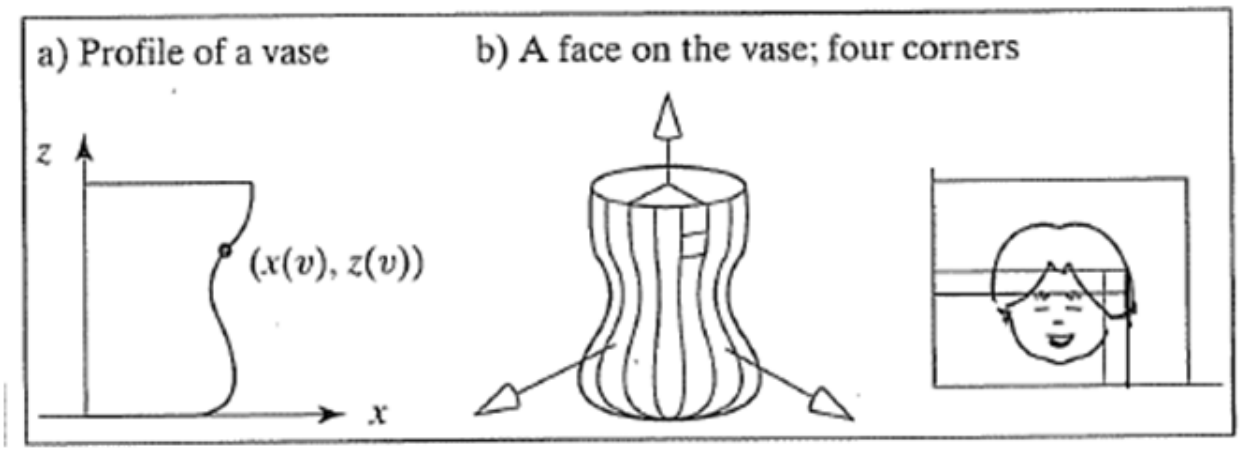
لبهٔ چپ وجه iام در $\theta_i = 2\pi i/N,$ خواهد بود.

و رأس بالائی آن لبه (si,ti) برابر است با: $((2\pi i/N - \theta_a)/(\theta_b - \theta_a), 1)$

مابقی رئوس بطور مشابه

بافت حول سطح مدور

سطح مدور از نیمرخ $(x(v),z(v))$ بوجود می آید.



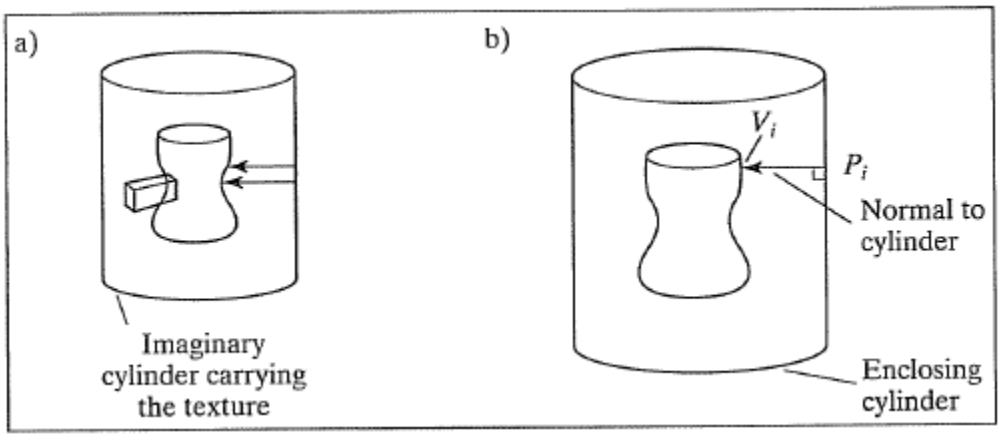
شکل ۱۲

$$P(u, v) = (x(v) \cos u, x(v) \sin u, z(v)).$$

رئوس وجه *Fi*:

$$P(u_i, v_i), P(u_{i+1}, v_i), P(u_i, v_{i+1}), \text{and } P(u_{i+1}, v_{i+1})$$

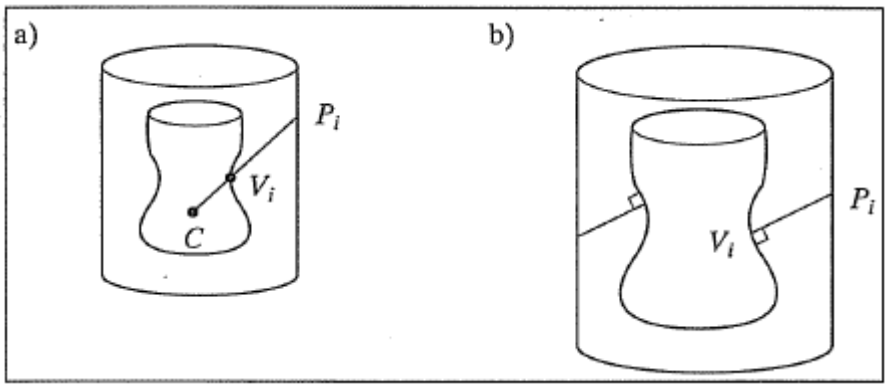
یک روش مشابه روش قبل برای محاسبهٔ (s,t) همانند در نظر گرفتن یک استوانهٔ مجازی



روشهای دیگر

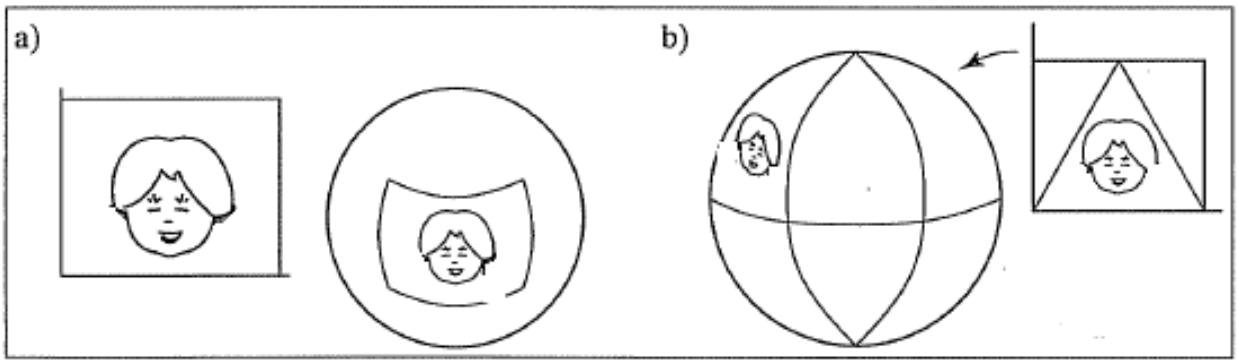
در نظر گرفتن خطی از مرکز شئ *C* به رأس *Vi* به محل تلاقی آن با استوانه در *Pi*

در نظر گرفتن عمود به سطح شئ در *Vi* و محل تلاقی آن با استوانه در *Pi*



بطور موردی باید روش مناسب انتخاب شود.

نگاشت بافت بر روی کره



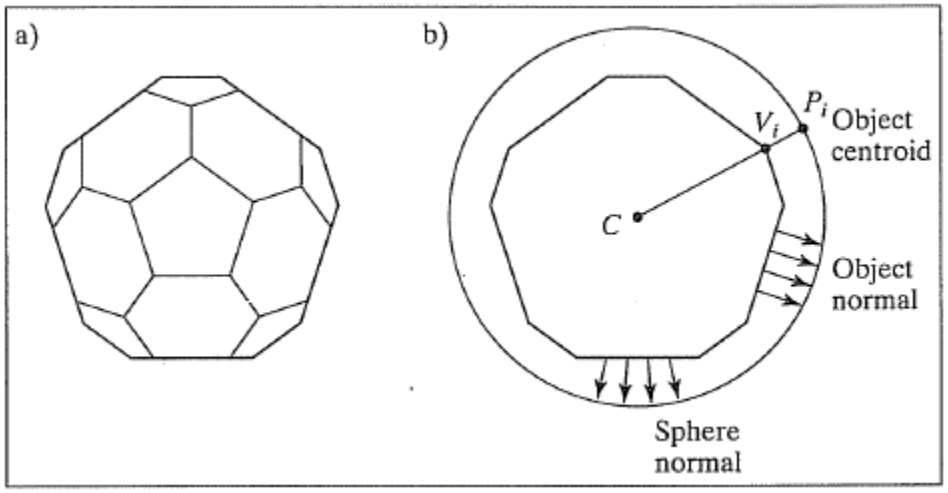
شکل ۱۳

برای نگاشت یک بافت مربعی بر روی قسمتی از کره که بین θa و θb و بین Φa و Φb قرار دارد:

$$(s_i, t_i) = ((\theta_i - \theta_a)/((\theta_b - \theta_a), (\phi_i - \phi_a)/(\phi_b - \phi_a)).$$

شکل ۱٤ ب نشان می دهد که می توان تمامی یک کره را توسط هشت بافت مثلثی بر روی هشت یک هشتم کره نگاشت.

نگاشت بافت بر روی اشیاء شبیه به کره

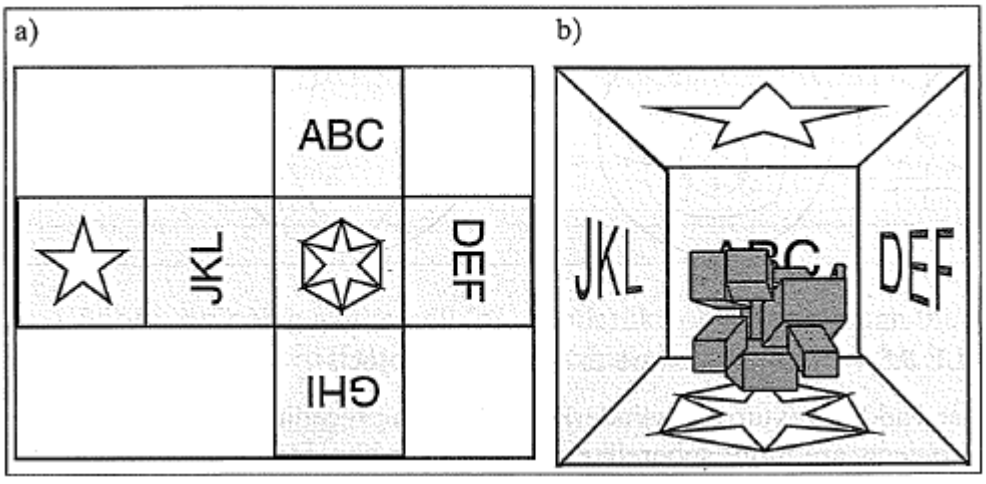


یک روش استفاده از بافتهای شش و چندضلعی جداگانه

روشهای دیگر:

- object-centroid: P_i is on a line from the centroid C through vertex V_i;
- object-normal: P_i is the intersection of a ray from V_i in the direction of the face normal;
- sphere-normal: V_i is the intersection of a ray from P_i in the direction of the normal to the sphere at P_i.

استفاده از جعبهٔ مجازی



شکل ۱٤

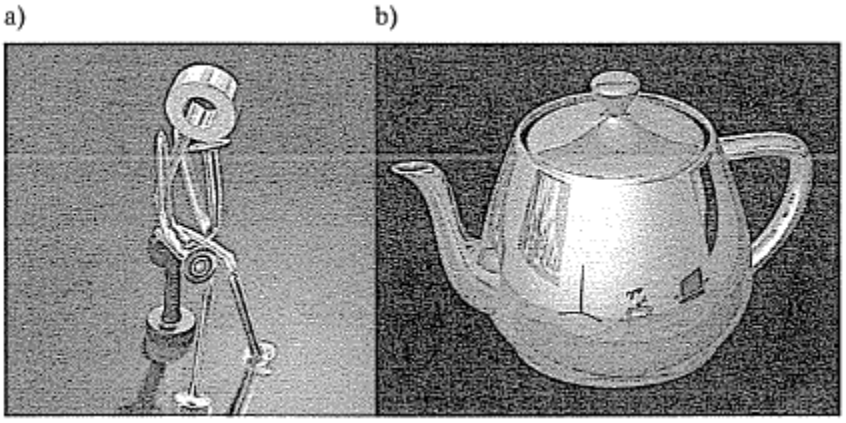
نگاشت انعکاسی

دیدن انعکاسهائی از شئ که نمایانگر محیط اطرافش باشد.

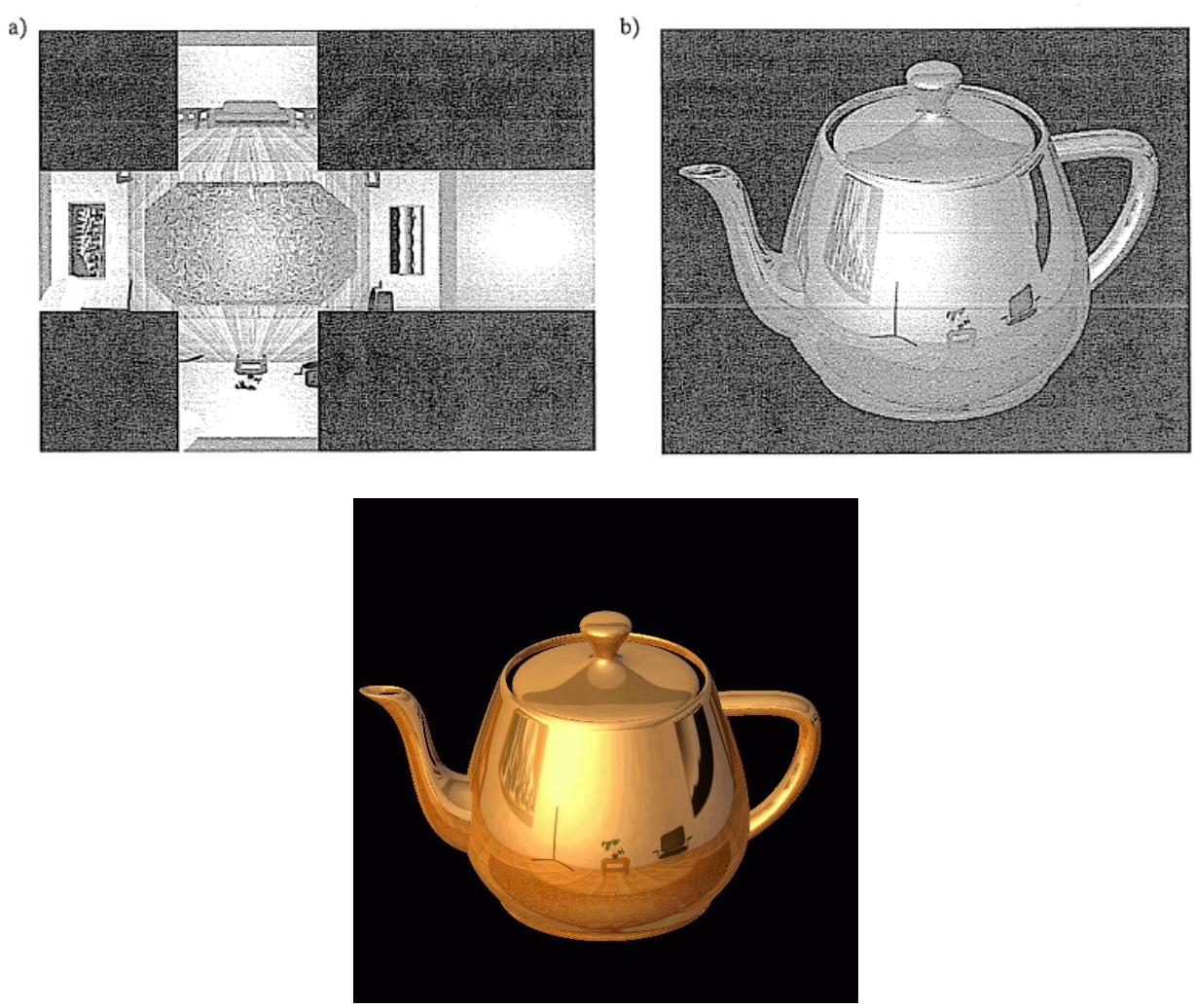
دو روش:

نگاشت کرومی (*chrome mapping*) و نگاشت محیطی (*environment mapping*)

نگاشت کرومی، محیط بصورت مبهم تصویر می شود. در نگاشت محیطی یک تصویر قابل تشخیص از محیط بر روی شئ دیده می شود.

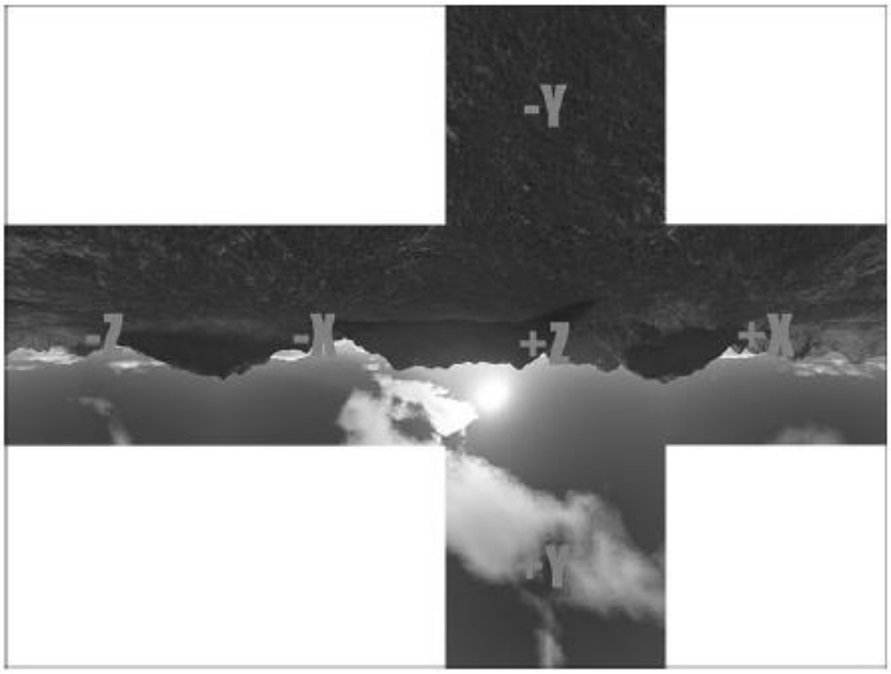


شکل ۱۵ الف) نگاشت کرومي، ب) نگاشت محيطي.



شکل ۱۶

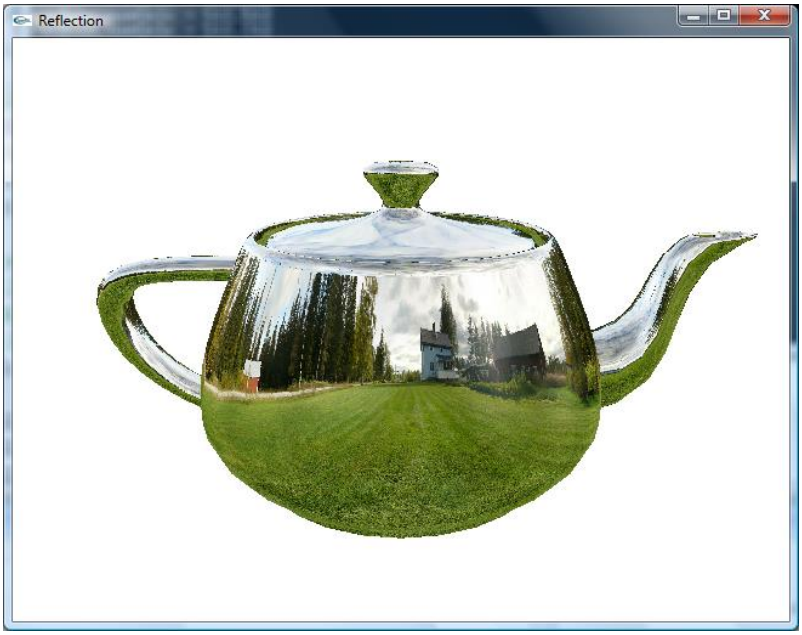
تهیه ۶ تصویر مورد نیاز با برنامه یا با عکسبرداری.توسط برنامه می توان دوربین را در محیط قرار داد و هر بار حجم دید را به گونه ای در نظر گرفت که یکی از دیوارها (یل یک سمت محیط) دیده شود و سپس منظره مشاهده شده را ذخیره نمود. چینش این ۶ تصویر بصورت شکل زیر می باشد. برای اینکه ترتیب قرار گیری تصاویر را بهتر متوجه شوید خود را تصور کنید که رو به سمت محور $z-$ ایستاده اید و در خلاف جهت عقربه های ساعت بدور خود حرکت می کنید. ترتیب محورهائی که دیده خواهند شد عبارت خواهند بود از $Z-$، $X-$، $Z+$، و $X+$. در زیر پای شما محور $Y-$ و در بالای سرتان محور $Y+$ قرار خواهد داست.



شکل ۱۷ چینش شش وجه مکعب جهت استفاده در نگاشت CUBEMAP [Wright, et. al 2007].

بافتهای نگاشت مکعبی بوسیلهٔ شش بار فراخوانی روال glTexImage2D() تهیه می شوند، که آرگومان *target* آن یکی از شش وجه مکعب (+X، -X، +Y، -Y، +Z و –Z) را معین می کند. تمامی این بافتها باید دارای ابعاد برابر باشند. در قطعه کد زیر آرگومان *imageSize* بایستی توانی از دو باشد.

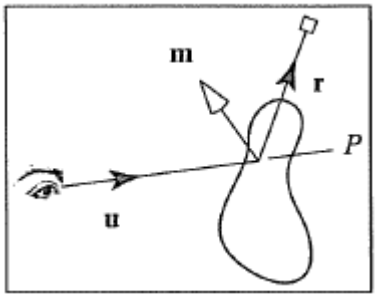
```
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_X, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image1);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_X, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image4);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Y, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image2);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Y, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image5);
glTexImage2D(GL_TEXTURE_CUBE_MAP_POSITIVE_Z, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image3);
glTexImage2D(GL_TEXTURE_CUBE_MAP_NEGATIVE_Z, 0, GL_RGBA,
 imageSize, imageSize, 0, GL_RGBA, GL_UNSIGNED_BYTE, image6);
```



شکل ۱۸

در نگاشت کرومی و محیط اگر شئ حرکت کند نگاشت بافت بر روی شئ تغییر می کند، ولی نگاشت معمولی ثابت می ماند. اگر یک کره براق حول مرکزش دوران کند، در نگاشت معمولی بافت می چرخد اما در نگاشت کرومی و محیطی ثابت می ماند.

برنامه نویسی



$$\mathbf{r} = \mathbf{u} - 2(\mathbf{u} \cdot \mathbf{m})\,\mathbf{m}.$$

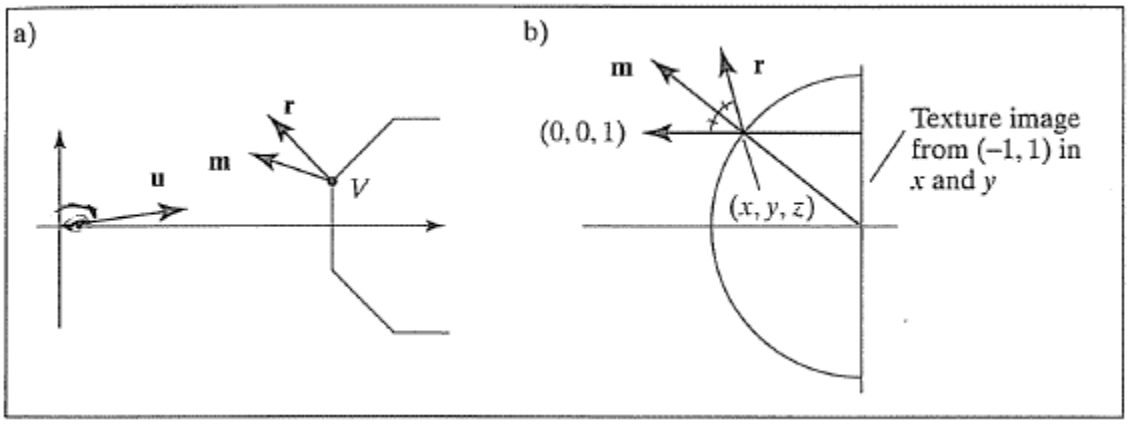
اشعهٔ انعکاس یافته در جهت *r* حرکت کرده تا به یک سطح فرضی با بافت متصل به آن برخورد می کند. راحت تر است که فرض شود یک شئ براق در مرکز قرار گرفته و خیلی نسبت به مکعب یا کرهٔ در بر گرفته کوچک است. در این حالت *r* تقریباً از مرکز شئ نشأت می گیرد.

OpenGL ابزاری برای اجرای تقریبی نگاشت محیطی برای حالتی که بافت بر روی یک مکعب بزرگ قرار دارد مهیا نموده است.

```
glTexGenf(GL_S,GL_TEXTURE_GEN_MODE, GL_CUBE_MAP);
glTexGenf(GL_T,GL_TEXTURE_GEN_MODE, GL_CUBE_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

خود *OpenGL* مقدار (s,t) را برای رأس حساب می کند. در صورت استفاده از کرهٔ دربر گیرنده:

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```



شکل ۱۹

$$(s,t) = \left(\frac{1}{2}\left(\frac{r_x}{p} + 1\right), \frac{1}{2}\left(\frac{r_y}{p} + 1\right)\right)$$

$$p = \sqrt{r_x^2 + r_y^2 + (r_z + 1)^2}.$$

صافی بافت

سطحی که بافت بر روی آن نگاشته می شود همواره ابعادی برابر با آن ندارد.

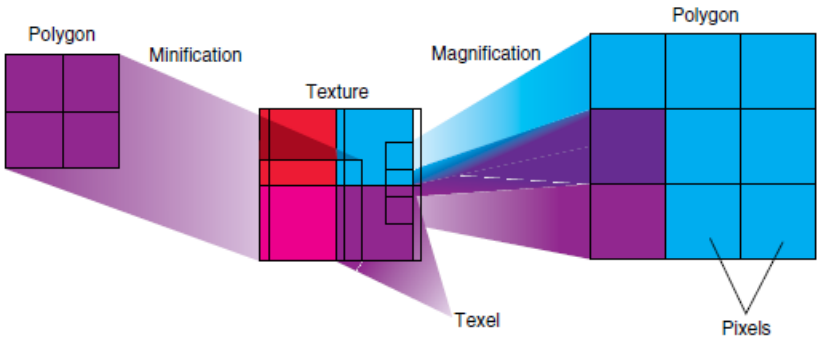
گاه نگاشت بر روی ابعادی بزرگتر از خود

گاه نگاشتی بر روی ابعادی کوچکتر از خود

این کار توسط صافی کردن (*filtering*) انجام می شود.

استفاده از نزدیکترین تکسل به مرکز پیکسل

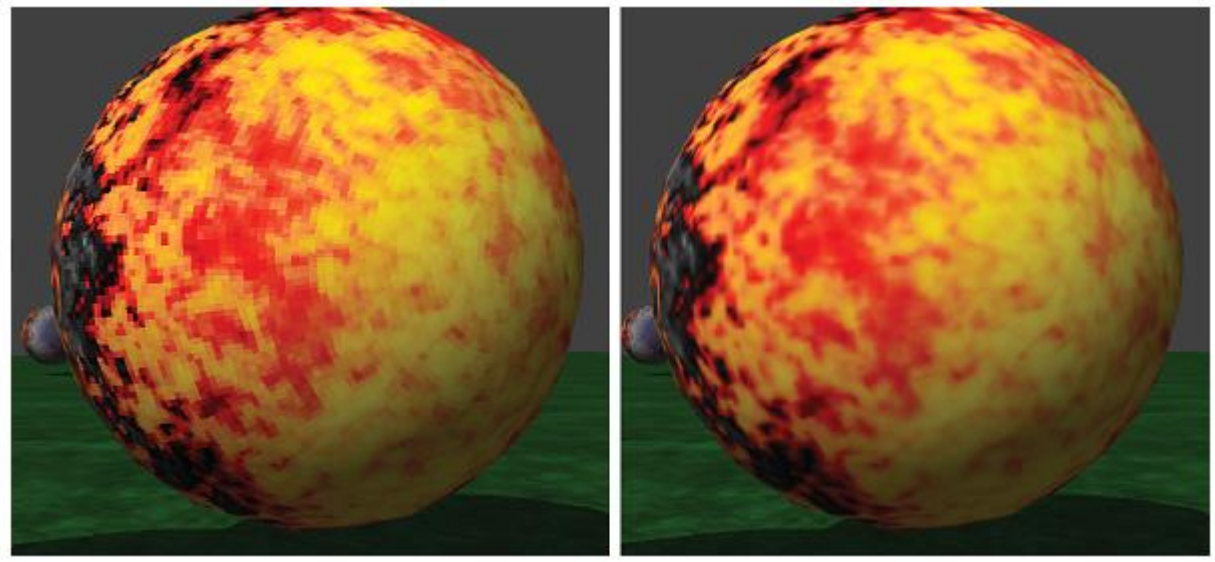
```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```



شکل ۲۰ بزرگنمائي و کوچکنمائي بافت [Shreiner 2010].

استفاده از میانگین وزندار خطی مقادیر در یک همسایگی ۲×۲ از تکسلهائی که به مرکز پیکسل نزدیکتر هستند.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```



شکل ۲۱

صافی خطی روش بهتری است ولی زمانگیرتر از روش خطی می باشد.

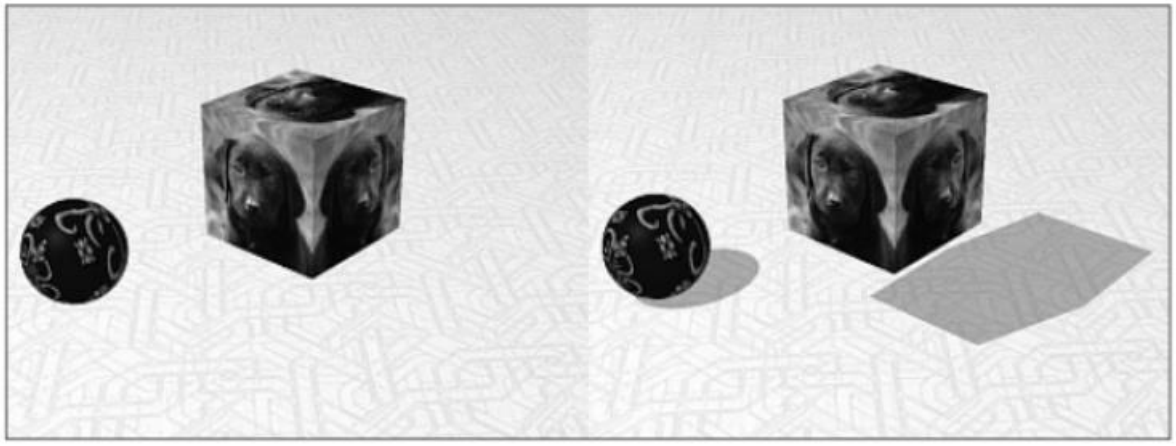
خلاصه ای از برخی توابع بافت:

glTexImage	بارکردن تصویر
glTexCood	نگاشت بافت به قطعه
glTexEnv	تغییر محیط بافت
glTexParameter	نشاندن پارامترهای نگاشت بافت

مدیریت چند بافت	glBindTexture

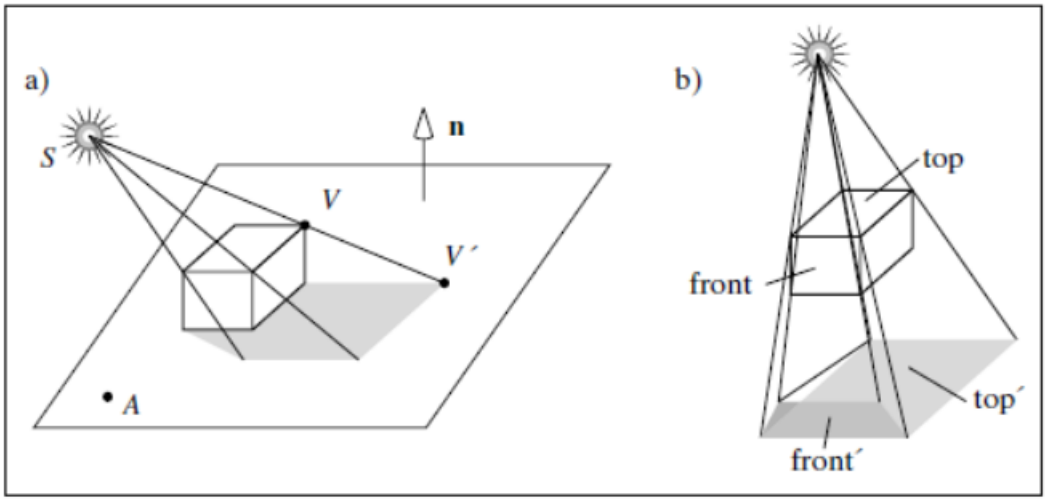
سایه [۲]

وجود سایه می تواند درک بهتری از صحنه ای که مشاهده می شود را بوجود آورد.



شکل ۲۲ اثر وجود سایه.

سایه به عنوان بافت



شکل ۲۳

ابتدا رسم صفحه بصورت معمولی با انعکاس وافر، پراکنده، و درخشنده، سپس بازرسم شش افکنش فقط با نور وافر، سپس رسم جعبه

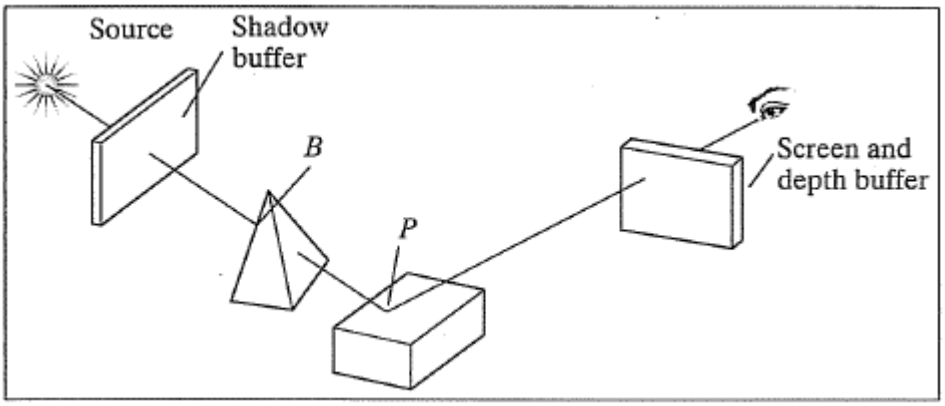
محاسبهٔ سایهٔ نقطهٔ V:

[2] shadow

$$V' = S + (V - S)\frac{\mathbf{n}\cdot(A - S)}{\mathbf{n}\cdot(V - S)}$$

استفاده از میانگیر سایه [3]

یک میانگیر برای هر منبع نور در نظر گرفته می شود. مقدار زیادی حافظه نیاز دارد ولی لازم نیست سایه حتماً بر روی صفحه باشد. ایده آن است که هر نقطه ای که از دید منبع نور پنهان است باید در سایه باشد. در میانگیر سایه، شبه عمق نزدیکترین وجه به منبع نگاهداری می شود. جهت ایجاد آن می توان تصور کرد که چشم در مکان منبع قرار گرفته و برای آن یک میانگیر عمق می خواهیم تشکیل دهیم.



شکل ۲۴

مقادیر میانگیر سایه مستقل از مکان بیننده بوده ولی اگر شئ حرکت کند (یا منبع) نیاز به محاسبهٔ دوباره مقادیر آن می باشد.

نیاز به:

- محاسبهٔ شبه عمق D از منبع به P
- اندیس $[i][j]$ به میانگیر سایه که باید آزموده شود
- مقدار $d[i][j]$ در میانگیر سایه

اگر $d[i][j]$ کمتر از D باشد، نقطهٔ P در سایه است و رنگ نقطهٔ P فقط با استفاده از نور وافر محاسبه می شود. در غیر اینصورت نقطه در سایه نبوده و رنگ آن با استفاده از نور وافر، پراکنده، و درخشنده محاسبه می شود.

تمرین

۱– الف) نشان دهید که اشعهٔ از نقطهٔ منبع S از طریق رأس V صفحهٔ $n.(P-A)=0$ را در $t^*=n.(A-S)/n.(V-S)$ قطع می کند. ب) در ادامه نشان دهید که توسط آن نقطهٔ برخورد V' که در متن ذکر شد بدست می آید.

[3] Shadow buffer

۲– الف) نشان دهید که عبارتی که در متن برای یافتن V' بیان شد را می توان بصورت ضرب ماتریسی $V'=M(Vx,Vy,Vz,1)T$، که M یک ماتریس ۴×۴ است نوشت. ب) عناصر M را بر حسب A، S، و n بیان نمائید.