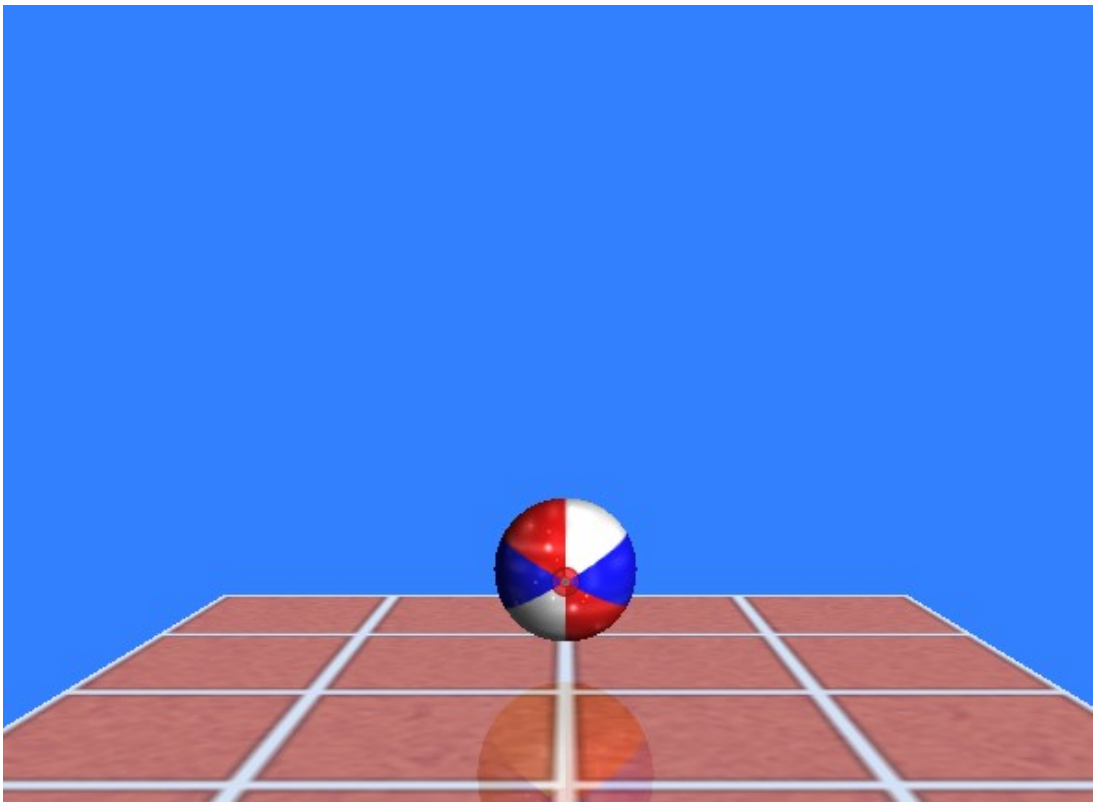
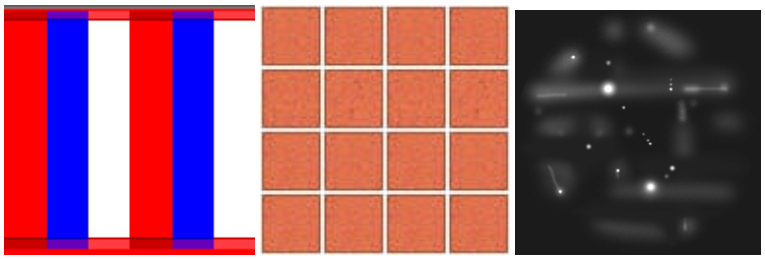
მულტიტექსტურირება და სტენსილ ბუფერი

- რამოდენიმე ტექსტურის ერთდროულად დადება
- სტენსილ ბუფერი და მისი გამოყენება
- როგორ დავხატოთ არეკვლის ეფექტი



დავქაჩოთ და გავხსნათ **advtexturing** პროექტი. ეს არის ბაზური პროექტი რომელსაც შევცვლით, არეკვლის ეფექტის მიღების მიზნით. პროგრამაში უკვე დაყენებულია განათება და დაშვებულია 2 განზომილებიანი ტექსტურირება.

პროგრამის ინიციალიზირებისას მეხსიერებაში იტვირთება 3 ტექსტურა:



1. ბურთის, 2. იატაკის და 3-ე ისევ ბურთის არეკვლის ეფექტის მისაღები.

```
EnvtxID =  loadJpegAs2DTexture("textures/Envroll.jpg" );
WalltxID = loadJpegAs2DTexture("textures/Envwall.jpg" );
BalltxID = loadJpegAs2DTexture("textures/Ball.jpg" );
```

ერთადერთი სიახლე რაც დამატებულია ბაზურ პროგრამაში ესაა სტენსილ ბუფერის შექმნა. ეს ხდება პროგრამის კადრის ბუფერების გამოცხადების დროს:

```
glutInitDisplayMode ( GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH | GLUT_STENCIL );
```

init-ფუნქციაში ხდება სტენსილ ბუფერის გასუფთავება და 0-ების ჩაწერა - `glClearStencil(0);`

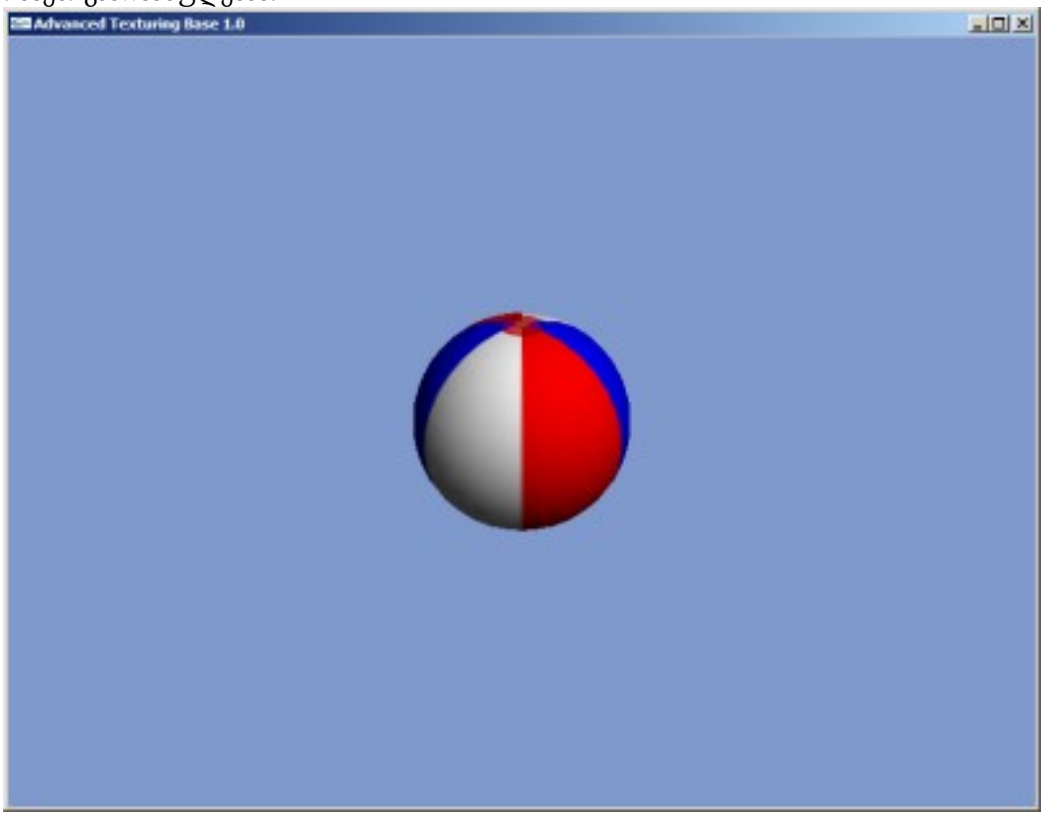
მარტივი მულტიტექსტურირება

ბურტის დახატვისთვის ჩვენ გამოვიყენებთ drawSphere ფუნქციას.

ხატვის ფუნქციაში ამ კოდის ჩამატების მერე:

```
glRotatef(30,1,0,0);
drawBall();
```

მივიღებთ ასეთ გამოსახულებას:

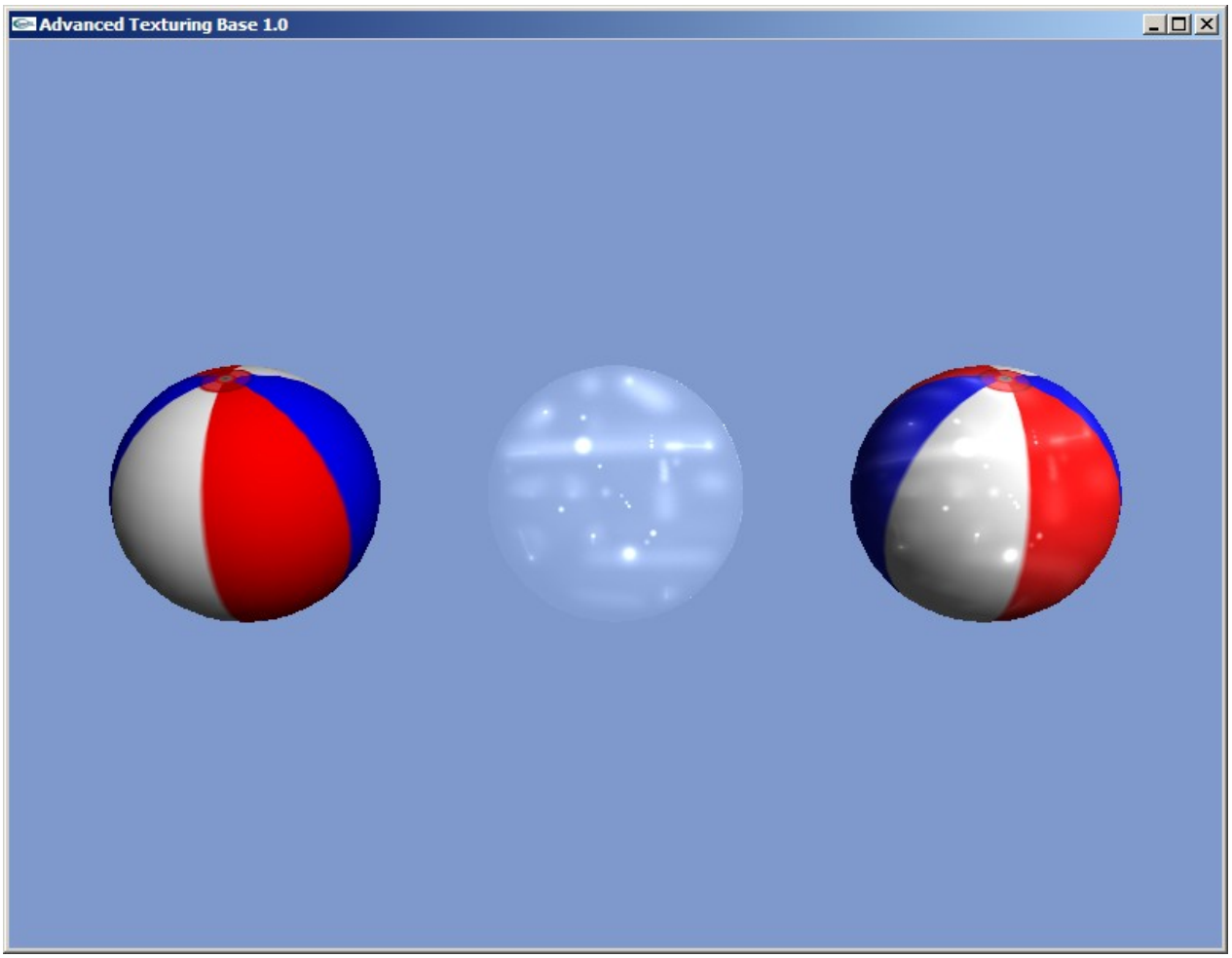


როგორ მოხდა რომ ობიექტს აქვს ტექსტურა როცა არ აგვირჩევია bindTexture-ს საშუალებით ?

OpenGL-ში უკანასკნელად ჩატვირთული ტექსტურა გაჩუმებით ითვლება არჩეულად. სფეროს ობიექტი კი

ისეა აგებულია რომ აღწერისას ავტომატურად დაგენერირდა ტექსტურის კოორდინატები. რადგანაც ჩვენ პროგრამაში ვაპირებთ რამოდენიმე ტექსტურის გამოყენებას, ამიტომ სფეროს დახავის წინ უმჯობესია ტექსტურის არჩევა: **`glBindTexture(GL_TEXTURE_2D, BalltxID);`**

ეხლა ჩვენ ისე უნდა დავადოთ მეორე ტექსტურა ბურთს ისე რომ მივიღოთ არეკვლები.



ამისთვის

1. მეორე ტექსტურას ჩვენ დავადებთ სფერული მეფირების გამოყენებით. ამიტომ საჭიროა აპლიკაციის ინიციალიზაციისას დავუშვათ სფერული მეფირება:

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_SPHERE_MAP);
```

2. ავირჩიოთ მეორე ტექსტურა - `glBindTexture(GL_TEXTURE_2D, EnvtxID);`

ჩვენ გვინდა რომ მეორე ტექსტურა იყოს 40% გამჭირვალე, ამიტომ უნდა დავუშვათ გამჭირვალობა და ალფა კომპონენტს დავუყენოთ შესაბამისი მნიშვნელობა:

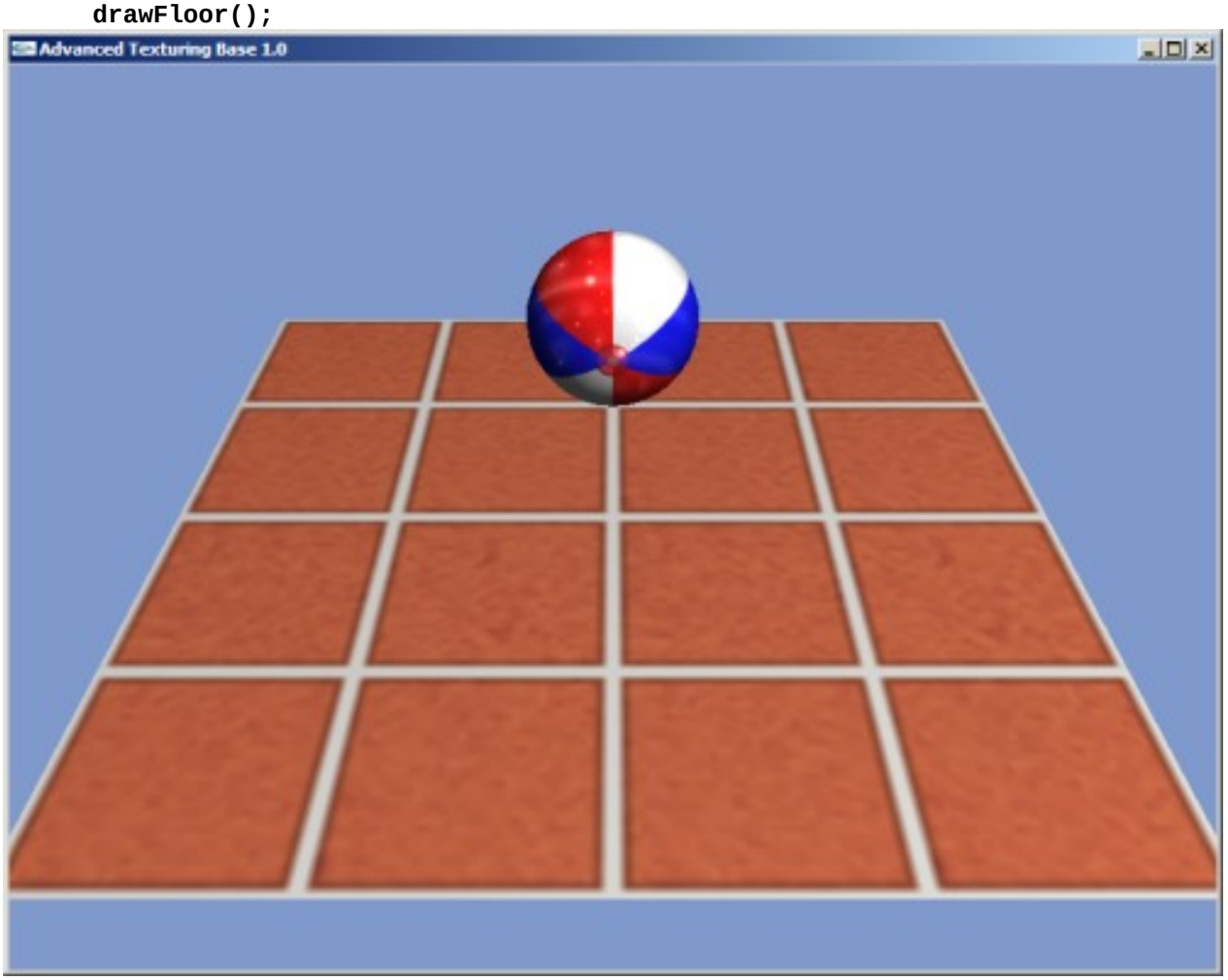
```
glColor4f(1.0f, 1.0f, 1.0f, 0.4f);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
```

3. უნდა ჩავრთოთ სფერულ მეფირება, დავხატოთ ბურთი ხელმეორედ იგივე ადგილას და გამოვრთოთ ყველა ზედმეტი ოფცია:

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
drawBall();
glDisable(GL_TEXTURE_GEN_S);
glDisable(GL_TEXTURE_GEN_T);
glDisable(GL_BLEND);
```

ამის შემდეგ ჩვენ უკვე გვაქვს არეკვლებიანი ბურთი. დავამატოთ იატაკის ხატვაც,. დასრულებული კოდი:

```
gluLookAt(0.0, 0.6, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 );
glRotatef(30,1,0,0);
glBindTexture(GL_TEXTURE_2D, BalltxID);
glTranslatef(0,1,0);
drawBall();
glBindTexture(GL_TEXTURE_2D, EnvtxID);
glColor4f(1.0f, 1.0f, 1.0f, 0.4f);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
drawBall();
glDisable(GL_TEXTURE_GEN_S);
glDisable(GL_TEXTURE_GEN_T);
glDisable(GL_BLEND);
glTranslatef(0,-1,0);
drawFloor();
```



სტენსილ-ბუფერი

არეკვლის ეფექტის მისაღებად საჭიროა რამოდენიმეჯ გადავხატოთ სცენა. პირველი რასაც ვაკეთებთ ესაა ვკრძალავთ ეკრანზე ხატვას: **glColorMask(0,0,0,0)**
ამ ფუნქციის საშუალებით შეიძლება გარკვეული ფერის დახატვის აკრძალვაც.
ვაყენებთ სტენსილ-ტესტირებას და შესაბამის ფუნქციას:

ა. `glEnable(GL_STENCIL_TEST);`
ბ. `glStencilFunc(GL_ALWAYS, 1, 1);`
გ. `glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);`

ა - ხდება სტენსილ ბუფერის გააქტიურება.
ბ- ახდენს სტენსილ ბუფერის ფუნქციის განსაზღვრას. პირველი პარამეტრი GL_ALWAYS- მიუთითებს რომ ეს ფუნქცია უნდა შესრულდეს ყოველთვის, მოუხედავად იმისა თუ რა იქნება სტენსილ-ტესტის შედეგი. მეორე პარამეტრი ეს არის მნიშვნელობა და მესამე მასკა. როდესაც სტენსილ ტესტირება წარმატებიტ ჩაივლის ხდება მეორე პარამეტრის და მასკის and ოპერაციით დამატება და სტენსილ ბუფერში ჩაწერა.
გ - აქ ზუსტდება თუ რა უნდა მოხდეს როცა 1 - სტენსილ ტესტი ჩავარდება (ჩვენ შემთხვევაში ეს არ მოხდება რადგან ჩვენ ავირჩიეთ GL_ALWAYS, ანუ ტესტი ყოველვის გაივლის), მაშინ შენარჩუნდეს(GL_KEEP) სტენსილ-ბუფერის მიმდინარე მნიშვნელობა . 2 -თუ სტენსილ ტესტი გაივლის და სიღრმისეული ჩავარდება (იგივე რაც წინაზე), და ბოლოს 3-თუ სტენსილ ტესტი გაივლის მაშინ მოხდება წინა ფუნქციაში მითითებული 2 მნიშვნელობის დაAND-ება და სტენსილ ბუფერში ჩაწერა(GL_REPLACE).

ადამიანური ენით ჩვენ აქ ვეუბნებით OpenGL-ს რომ ხატვა მოხდება მხოლოდ სტენსილ ბუფერში. (სტენსილ ბუფერში თავიდან ჩავწერეთ 0-ები). როგორც კი რამე ”დაიხატება” ეკრანზე შესაბამისი პიქსელისთვის ჩაიწეროს 1-ნი სტენსილ ბუფერში.

წინა მაგალითში იატაკის დახატვამდე ჩავამატოთ ზემოთ მოყვანილი კოდი და უნდა გამოვრთოთ სიღრმისეული ტესტირებაც **glDisable(GL_DEPTH_TEST)**

ამ ოპერაციის შემდეგ ჩვენ ჩავწერეთ 1-ები სტენსილ ბუფერში ყველა იმ პიქსელისათვის რომელიც იხატება იატაკის ხატვისას.

არეკლილი ბურთისთვის უნდა განისაზღვროს მოჭრის ზედაპირი. ჩვენ შეეთხვევაში ესაა:
```
double eqr[] = {0.0f,-1.0f, 0.0f, 0.0f};
```
როგორც გვახსოვს ესენია სიბრტყის განტოლების კოეფიციენტები(A,B,C,D). ანუ ამ მოჭრის სიბრტყის გააქტიურების შემდეგ დაიხატება მხოლოდ ის ობიექტები რომლებიც XZ პარალელური სიბრტყის და Y-ის (-1) დაბლა მდებარეობენ. ყველა დანარჩენი მოიჭრება.

ა.`glEnable(GL_DEPTH_TEST);`
ბ.`glColorMask(1,1,1,1);`
გ.`glStencilFunc(GL_EQUAL, 1, 1);`
დ.`glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);`
ე.`glEnable(GL_CLIP_PLANE0);`
ვ.`glClipPlane(GL_CLIP_PLANE0, eqr);`

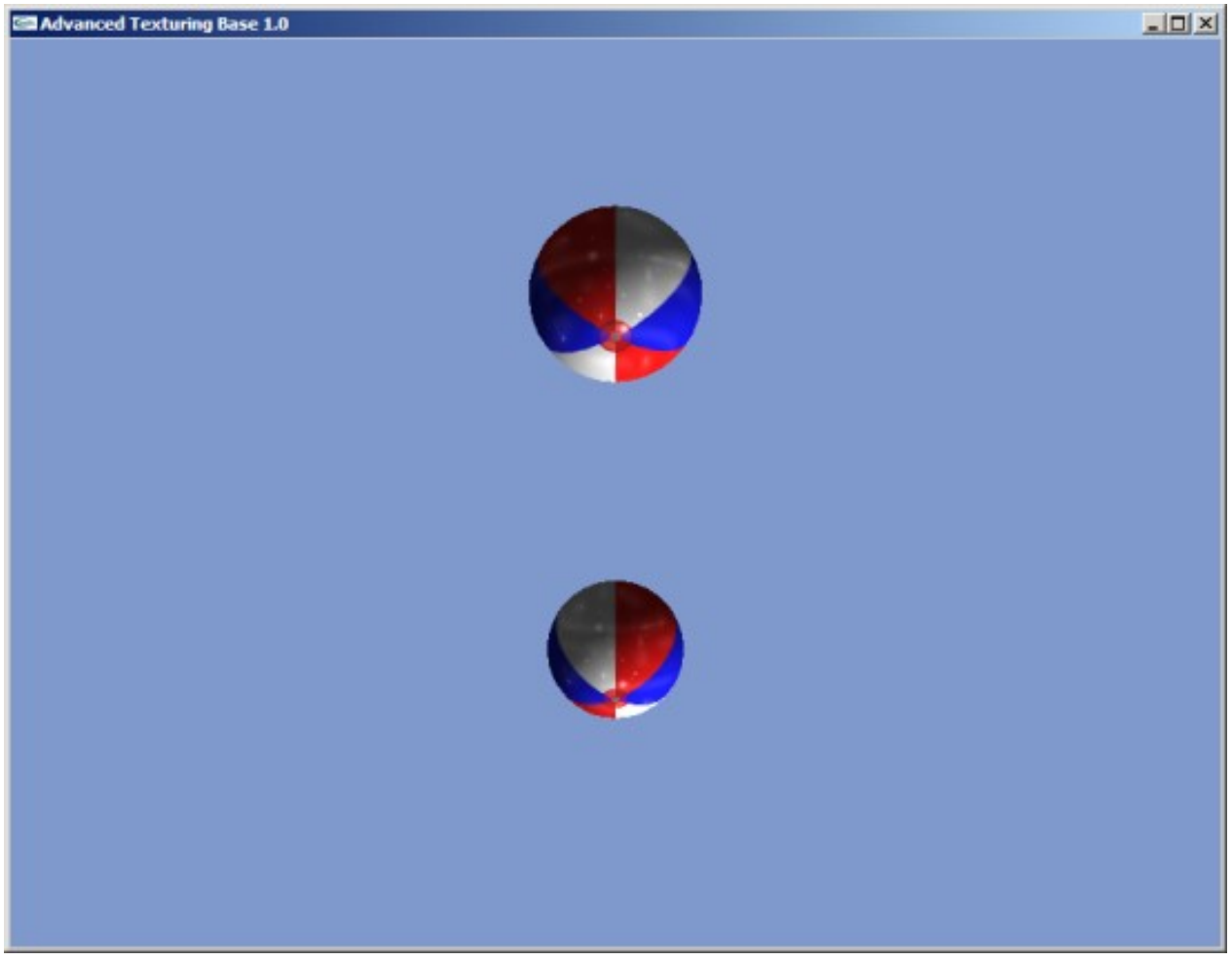
ზემოთ მოყვანილი კოდი აკეთებს შემდეგს. უშვებს სიღრმისეულ ტესტირებას(ა), ვრთავთ ხატვას(ბ), ვაყენებთ სტენსილ ტესტირებას ისე რომ მხოლოს ის პიქსელები დაიხატოს რომელთა შესაბამისი მნიშვნელობა სტენსილ ბუფერში არის 1-ნი(გ და დ). ვაყენებთ და ვრთავთ წინასწარ აღწერილ მოჭრის სიბრტყეს(ვ და ე).

დრო მოვიდა დავხატოთ ჩვენი არეკლილი ბურთი.

```
ა.glPushMatrix();
ბ.glScalef(1.0f, -1.0f, 1.0f);
გ.glLightfv(GL_LIGHT0, GL_POSITION, LightPos);
დ.glTranslatef(0.0f, 1, 0.0f);
ე.drawBall();
ვ.glPopMatrix();
```

ვინახავთ და აღვადგენთ მიმდინარე ხედვის მატრიცის მნიშვნელობებს (ა და ვ). ვახდენთ Y ღერძის შებრუნებას, რადგანაც არეკლვა ხდება ZX სირტყეში(ბ). ვაყენებთ განათების პოზიციას(გ) ვიწყებთ ხატვას 1 იუნიტით ქვემოთ(დ). ვხატავთ ბურთს (ე).

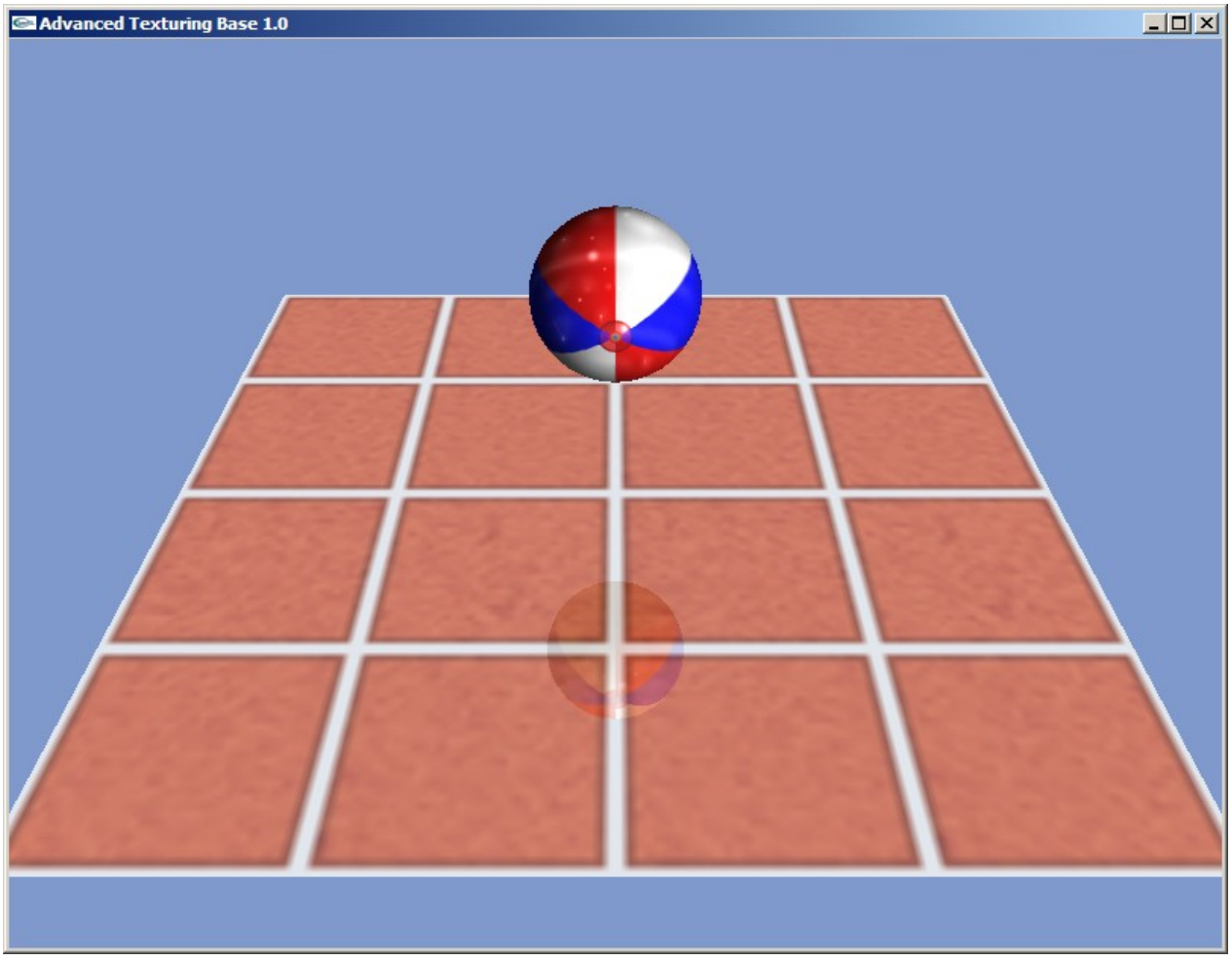
ეხლა თუ გავაუქმებთ მოჭრის სიბრტყეს და გამოვრტავთ სტენსილ ბუფერს მივიღებთ ასეთ სურათს:



სრულყოფილი არეკვლის ეფექტის მისაღებად საჭიროა კიდევ ერთხელ დავხატოთ იატაკი .

```
glLightfv(GL_LIGHT0, GL_POSITION, LightPos);
ა.glEnable(GL_BLEND);
ბ.glDisable(GL_LIGHTING);
გ.glColor4f(1.0f, 1.0f, 1.0f, 0.8f);
დ.glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
ე.drawFloor();
ვ.glEnable(GL_LIGHTING);
ზ.glDisable(GL_BLEND);
```

იატაკი უნდა დავხატოთ როგორც 80% გამჭირვალე. ამიტომ ვრთავთ გამჭირვალობას(ა). აქ ვიყენებთ განათებას ამიტომ ვთიშავთ(ბ). ვაყენებთ ფერის ალფა კოპონენტს 80%-ზე (დ). განვსაზღვრავთ გამჭირვალობის ფუნქციას(დ). ვხატავთ იატაკს(ე). ვაბრუნებთ განათებას და ვაუქმებთ გამჭირვალობას(ვ და ზ). ვუალა, ჩვენი არეკლილი ზედაპირი



ხატვის დასრულებული კოდი:

```
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT );

        glLoadIdentity();

        gluLookAt(0.0, 0.6, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0 );

        glRotatef(30,1,0,0);
        glBindTexture(GL_TEXTURE_2D, BalltxID);

        glTranslatef(0,1,0);
        drawBall();

        glBindTexture(GL_TEXTURE_2D, EnvtxID);
        glColor4f(1.0f, 1.0f, 1.0f, 0.4f);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE);
        glEnable(GL_TEXTURE_GEN_S);
        glEnable(GL_TEXTURE_GEN_T);
        drawBall();
        glDisable(GL_TEXTURE_GEN_S);
        glDisable(GL_TEXTURE_GEN_T);
        glDisable(GL_BLEND);

        glTranslatef(0,-1,0);

        glColorMask(0,0,0,0);
        glEnable(GL_STENCIL_TEST);
        glStencilFunc(GL_ALWAYS, 1, 1);
        glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE);
        glDisable(GL_DEPTH_TEST);

        drawFloor();

        double eqr[] = {0.0f,-1.0f, 0.0f, 0.0f};
        glEnable(GL_DEPTH_TEST);
        glColorMask(1,1,1,1);
        glStencilFunc(GL_EQUAL, 1, 1);
        glStencilOp(GL_KEEP, GL_KEEP, GL_KEEP);
        glEnable(GL_CLIP_PLANE0);
        glClipPlane(GL_CLIP_PLANE0, eqr);

        glPushMatrix();
        glScalef(1.0f, -1.0f, 1.0f);
        glLightfv(GL_LIGHT0, GL_POSITION, LightPos);
        glTranslatef(0.0f, 1, 0.0f);
        glBindTexture(GL_TEXTURE_2D, BalltxID);
        drawBall();
        glBindTexture(GL_TEXTURE_2D, EnvtxID);
        glColor4f(1.0f, 1.0f, 1.0f, 0.4f);
        glEnable(GL_BLEND);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE);
        glEnable(GL_TEXTURE_GEN_S);
        glEnable(GL_TEXTURE_GEN_T);
        drawBall();
        glDisable(GL_TEXTURE_GEN_S);
        glDisable(GL_TEXTURE_GEN_T);
        glDisable(GL_BLEND);
        glPopMatrix();

        glDisable(GL_CLIP_PLANE0);
        glDisable(GL_STENCIL_TEST);

        glLightfv(GL_LIGHT0, GL_POSITION, LightPos);
        glEnable(GL_BLEND);
        glDisable(GL_LIGHTING);
        glColor4f(1.0f, 1.0f, 1.0f, 0.8f);
        glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
        drawFloor();
        glEnable(GL_LIGHTING);
        glDisable(GL_BLEND);
```