ტექსტურირება

შეიძლება ითქვას რომ, ტექსტურირება არის OpenGL-ის ყველაზე რთული ნაწილი. ალბათ ამიტომაც, ყველაზე ხშირად ზუსტად ტექსტურირებასთან დაკავშირებული ფუქციების ევოლუცია ხდებოდა და მასთან დაკავშირებული ახალი გაფართოებები ემატებოდა OpenGL-ს.

OpenGL-ი მხარს უჭერს 4 ტიპის ტექსტურებს, ესენია: 1-განზომილებიანს, 2-განზომილებიანს, 3-განზომილებიანს და კუბური ტექსტურებს. 1-განზომილებიანი (1D) ტექსტურა ესაა პიქსელების მიმდევრობა რომელიც მხოლოდ სიგანეზე შეიცავს ინფორმაციას. 2D ტექსტურა შეიცავს ინფორმაციას როგორც სიგანეზე ასევე სიგრძეზეც. 3D ტექსტურა შეიცავს ინფორმაციას სიგანეზე, სიგრძეზე და სიღრმეზე. კუბური ტექსტურა ესაა 6 ცალი 2 განზომილებიანი ტექსტურების მასივი. თითო 2D ტექსტურა თითო ძირითადი მიმართულებისთვის, ანუ (+x, -x, +y,-y,+z,-z)

OpenGL-ში არსებობს ტექსტურის მოდულის ცნება. ეს მოდული განსაზღვრავს გრაფიკული ამაჩქარებლის იმ ნაწილს რომელიც პასუხიმგებელია სხვადასხვა ტექსტურირებასთან დაკავშირებულ ოპერაციებზე. OpenGL 1.3 ვერსიიდან დაემატა საშუალება ვმართოთ რამოდენიმე ასეთი მოდული. თითოეული მოდული ინახავს ინფორმაციას შემდეგ ოპერაციებზე და მდგომარეობებზე:

1. მოდულის მდგომარეობა - ჩართული ან გამორთული.
2. ტექსტურის მატრიცების სტეკი. ტექსტურის მატრიცები გამოიყენება ტექსტურის კოორდინატების გარდასაქმნელად.
3. ავტომატური ტექსტურის კოორდინატების გენერაციის მდგომარეობა. ჩართული ან გამორთული.
4. გარემოს ტექსტურის მდგომარეობაზე ინფორმაცია.
5. მიმდინარე 1D ტექსტურა.
6. მიმდინარე 2D ტექსტურა.
7. მიმდინარე 3D ტექსტურა.
8. მიმდინარე კუბური ტექსტურა.

ყველა ეს პარამეტრი და მდგომარეობა შესაძლებელია აირჩეს მიმდინარე ტექსტურისათვის. ტექსტურის მოდულები გადანომრილია 0-დან GL_MAX_TEXTURE_UNITS-1-მდე. (ამ უკანასკნელის მნიშვნელობა შეგვიძლია მივიღოთ glGet ფუნქციის საშუალებით). აქტიური მოდულის არჩევა ხდება glActiveTexture ფუნქციის დახმარებით, რომელსაც გადაეცემა შესაბამისი სიმბოლური კონსტანტა.

ტექსტურებზე მომუშავე ყველა ფუნქცია მოქმედებს მხოლოდ არჩეულ/მიმდინარე ტექსტურის მოდულზე. იმისათვის რომ მივუთითოთ ტექსტურის მოდულს თუ რომელი ტიპის ტექსტურასთან ვაპირებთ მუშაობას, ვიძახებთ glEnable ფუნქციას და გადავცემთ შესაბამის კონსტანტებს (GL_TEXTURE_1D, GL_TEXTURE_2D, GL_TEXTURE_3D და GL_TEXTURE_CUBE_MAP) .

OpenGL-ში არის რამოდენიმე ფუნქცია რომელიც საშუალებას გვაძლევს შევცვალოთ ტექსტურის კოორდინატების დამუშავება და ვცვალოთ შესაბამისი მატრიცები. მიმდინარე მოდულის ტექსტურის სტეკის არჩევა ხდება glMatrixMode(GL_TEXTURE) ფუნქციით. glTexGen-ის დახმარებით ვირჩევთ ავტომატური ტექსტურის გენერირების ერთერთ რეჟიმს. ტექსტურის გამოსახულების შეცვლა ხდება glTexenv, glTexParameter და glTexImage ფუნქციებით. ტექსტურების მიბმა glBindTexture და ასევე glEnable/glDisable ნებისმიერი ტიპის ტექსტურებისთვის.

ტექსტურის კონკრეტული ობიექტის შესაქმნელად საჭიროა, გამოვიძახოთ glBindTexture. პირველი პარამეტრი ესაა ტექსტურის ტიპი. მეორე, უნიკალური რაიმე 0-ზე მეტი რიცხვი (ეს რიცხვი შეგვიძლია მივიღოთ glGenTextures ფუნქციის დახმარებით). ზუსტად რიცხვის დახმარებით ხდება შემდგომში პროგრამიდან ტექსტურაზე წვდომა. ახლად შექმნილი ტექსტურა ავტომატურად არჩეულია როგორც აქტიური. თუ glBindTexture-ს მეორე პარამეტრად გადავცემთ უკვე არსებული ტექსტურის იდენტიფიკატორს, მაშინ ეს ტექსტურა გახდება აქტიური. ამგვარად, პროგრამის დაწერისას შეგვიძლია შევქმნათ ნებისმიერი რაოდენობის ტექსტურები და ადვილად გადავერთოთ მათ შორის .

იმის მერე რაც შევქმენით ტექსტურის ობიექტი, საჭიროა მას მივაწოდოთ მონაცემები. 3D ტექსტურისთვის ეს ხდება glTexImage3D ფუნქციით. შესაბამისად 2D ტექსტურისთვის - glTexImage2D და glTextImage1D – 1D ტექსტურისთვის. OpenGL-ის 1.5 ვერსიის ჩათვლით, ამ ფუნქციების გამოძახებისას ყველა ტექსტურას უნდა ქონოდა 2-ის ხარისხის ჯერადი განზომილებები. 2.0-ში ეს შეზღუდვა უკვე აღარ არის. ამ ბრძანებების გამოძახების შემდეგ ხდება ტექსტურის მონაცემების მეხსიერებაში განთავსება. თუ საჭიროა ტექსტურის მხოლოდ რაღაც ნაწილის შეცვლა უნდა გამოვიყენოთ შემდეგი ფუნქციები, glTexSubImage1D/2D/3D. კარის ბუდერიდან ტექსტურების, ტექსტურის ნაწილის კოპირება შეიძლება glCopytexImage1D/2D ან glCopyTexSubImage1D/2D/3D ფუნქციებით.

OpenGL-ში ასევე არის საშუალება ტექსტურები შეკუმშულად შევინახოთ მეხსიერებაში. ეს ხდება glCompressedTexImage1D/2D/3D და glCompressedTexSubImage1D/2D/3D ფუნქციების გამოყენებით. კომპრესირებული ტექსტურების გამოყენება მნიშვნელოვნად ამცირებს ვიდეო ადაპტერის მეხსიერების დატვირთვას. სტანდარტული OpenGL-ი არ შეიცავს კონკრეტული კომპრესირებული ტექსტურების მხარდაჭერას, ამიტომ ამ თვისების გამოყენებამდე უნდა გავიგოთ აქვს თუ არა საშუალება ვიდეო ადაპტერს გამოიყენოს კონკრეტული კომპრესირბული ტექსტურების ფორმატი.

ჩვენ მიერ განხილული ტექსტურების შექმნის ყველა ბრძანებას გადაეცემა ტექსტურის-დეტალიზაციის-დონის(level-of-detail) პარამეტრი.ეს პარამეტრი აზუსტებს თუ როგორი და რამდენი ტექსტურა უნდა შეიქმნას სხვადასხვა დაშორების ობიექტებისთვის(ე.წ. mipmap textures). ”მიფმეფ

ტექსტურა” ესაა დალაგებული ტექსტურების მასივი, რომლებიც წარმოადგენენ იგივე ნახატს, ოღონდ ერთი განსვავებით, თითოეული მასივის გარჩევადობა არის 2-ჯერ ნაკლები ვიდრე წინა მასივის. ”მიფმეფ ტექსტურების” მთავარი იდეა იმაში მდგომარეობს რომ, ნებისმიერი ზედაპირი რომელიც შეიქმნება ამ ტექსტურების დახმარებით, სხვადასხვა გარჩევადობის პირობებში ერთნაერად კარგად გამოიყურებოდეს. თუ კონკრეტული ობიექტისთვის ტექსტურის შექმნისას დაშვებული იქნა ”მიფმეფების” შექმნაც მაშინ OpenGL-ი ავტომატურად შეეცდება შეარჩიოს საუკეთესო ”მიფმეფ” დეტალიზაცია მიმდინარე ობიექტისთვის. ტექსტურის პიქსელებს მეორენაერად ტექსელებსაც (Texel – Texture Element) უწოდებენ. შესაძლებელია 2 ტექსელს ან 2 ”მიფმეფის” დეტალიზაციის დონეს შორის გაკეთდეს ინტერპოლაცია. რომ შევაჯამოთ : ”მიფმეფ” დეტალიზაციით დატექსტურებული ობიექტები, ეკრანის ნებისმიერი გარჩევადობის პირობებში უფრო დეტალურად და ხარისხიანად გამოიყურება.

ტექსტურის შექმნის და მონაცემების განსაზღვრის შემდეგ ტექსტურის პიქსელებზე და სხვადასხვა პარამეტრების შეცვლა შესაძლებელია glTexParameter ფუნქციის დახმარებით. შესაძლებელია ტექსტურის შემდეგი პარამეტრების შეცვლა:

1. როგორ მოიქცეს ობიექტი როდესაც ტექსტურირების კოორდინატები გაცდება დასაშვებ საზღვრებს [0,1] - განმეორდეს, სარკისებულად აირეკლოს თუ ჩაიკეტოს.
2. დაპატარავების ფილტრი, რომელიც აზუსტებს ტექსტურის დისკრედიტაციის ხერხს, როდესაც ის უნდა დაპატარავდეს ტექსტურის კოორდინატთა სისტემიდან, ფანჯრის კოორდინატთა სისტემაში გადაყვანისას.
3. გადიდების ფილტრი. იგივე რაც წინა ოღონდ ტექსტურის გადიდების შემთხვევისთვის.
4. საზღვრის ფერი რომელიც განმეორდება ტექსტურის კოორდინატების ჩაკეტვის შემთხვევაში.
5. ტექსტურის პრიორიტეტი. მნიშვნელობა იცვლება [0,1]-მდე. ეს OpenGL-ს ეუბნება რამდენად სწრაფად დაამუშავოს ეს ტექსტურა.
6. ტექსტურის ჩაკეტვის და წანაცვლების მნიშვნელობები რომელიც ავტომატურად გამოითვლება OpenGL-ის მიერ.
7. ”მიფმეფირების” მაქსიმალური განზომილებები.
8. ”მიფმეფირების” მინიმალური განზომილებები.
9. სხვადასხვა პარამეტრები რომელიც დაკავშირებულია თუ რომელი ოპერაციები უნდა განხორციელდეს თუ ტექსტურა შეიცავს სიღრმისეულ ინფორმაციას.
10. მოხდეს თუ არა ”მიფმეფ” ტექსტურების ავტომატური გენერაცია როდესაც ტექსტურა ახალი შექმნილია ან შეიცვალა.

glTexEnv ფუნქცია აზუსტებს, თუ კონკრეტულად როგორ უნდა დაედოს ტექსტურა გეომეტრიულ პრიმიტივს. არსებობს რამოდენიმე რეჟიმი და წინასწარ განსაზღვრული ფორმულები თუ როგორ ხდება ტექსტურის პიქსელების, მატერიალების, განათების და სხვა პარამეტრების შეხამება ტექსტურის

დადებისას. ზუსტად ამის გამო glTexEnv-ის გამოყენებით შესაძლებელია მრავალფეროვანი ვიზუალური ეფექტების მიღება.

OpenGL-ი მხარს უწერს მულტიტექტურირებასაც. ეს ნიშნავს რომ 1 ობიეტი შესაძლებელია რამოდენიმე ტექსტურა დავადოთ. ამ შემთხვევაში ყოველი მომდევნო ტექსტურის დადებისას კონკრეტული პიქსელის ფერის გამოსათვლელად, წინა ტექსტურის შესაბამისი ფერის გათვალისწინება ხდება.

იმის მერე რაც ტექსტურა შეიქმნა, ჩაიტვირთა მეხსიერებაში, დაზუსტდა სხვადასხვა პარამეტრები და გააქტიურებულია, ხდება კონკრეტული ობიექტების დატექსტურება. ამისთვის საჭიროა თითოეული წვეროსთვის დაზუსტდეს ტექსტურის კოორდინატები glTexCoord ან glMultiTexCoord ფუნქციების საშუალებით თუ გეომეტრიულ ინფორმაციას მივაწვდიტ "ერთი-წვერო-დროის-ერთ-მომენტში" მეთოდის დახმარებით ან გამოვიყენოთ glTexCoordPointer ფუნქცია მასივების გამოყენების შემთხვევაში.

ტექსტურის კოორდინატები შეიძლება ავტომატურადაც დაგენერირდეს OpenGL-ის მიერ glTexGen-ის დახმარებით. ეს ფუნქცია საშუალებას გვაძლევს ავირჩიოთ 3 ავტომატური ტექსტურების გენერაციის მეთოდი. ამისთვის საჭიროა ჩავრთოთ ტექსტურის ავტომატური დაგენერირების რეჟიმი:

CODE

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

როდესაც ეს რეჟიმი ჩართულია ავტომატურად ხდება glTexCoord ფუნქციის გამოძახებების იგნორირება. შემდეგ ვირჩევთ ტექსტურის კოორდინატების ავტომატური დაგენერირების მეთოდს:

CODE

```
void glTexGenf(GLenum coord, GLenum pname, GLfloat param);
void glTexGenfv(GLenum coord, GLenum pname, GLfloat *param);
```

პირველი პარამეტრი coord, მიუთითებს რომელის ტექსტურის კოორდინატაზე ვმოქმედებთ (GL_S, GL_T,...). მეორე პარამეტრი (pname) შეიძლება იყოს ერთერთი ამ სამიდან:
GL_TEXTURE_GEN_MODE, GL_OBJECT_PLANE, ან GL_EYE_PLANE (დაწვრილებით ქვემოთ)
და ბოლოს პარამეტრი (param) აზუსტებს ავტომატური გენერაციის მეთოდს.

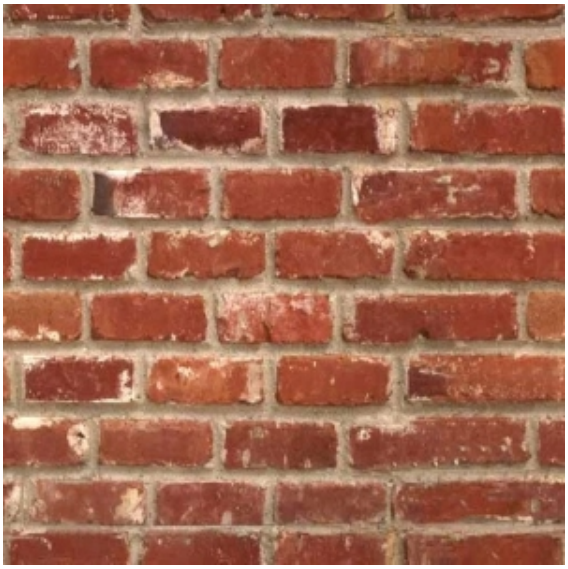
განვიხილოთ ტექსტურის გენერაციის რეჟიმი **GL_OBJECT_LINEAR**:
ამ დროს ტექსტურის კოორდინატები გამოითვლება შემდეგი ფორმულით:

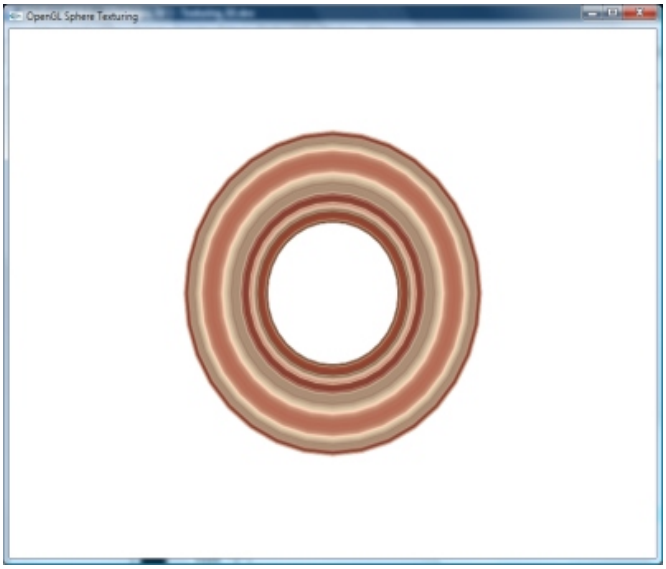
*coord = P1*X + P2*Y + P3*Z + P4*W*

სადაც X,Y,Z და W არიან ობიექტის წვეროს კოორდინატები და P1-P4 სიბრტყის განტოლების სადაც X,Y,Z და W არიან ობიექტის წვეროს კოორდინატები და P1-P4 სიბრტყის განტოლების კოეფიციენტები. ამ შემთხვევაში მეფირება ხდება მოცემული სიბრტყიდან. მაგალითად თუ გვინდა S და T დავაპროეციროთ Z=0 სიბრტყიდან უნდა დავწეროთ შემდეგი:

```
// Projection plane
GLfloat zPlane[] = { 0.0f, 0.0f, 1.0f, 0.0f };
. . .
. . .
// Object Linear
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGenfv(GL_T, GL_OBJECT_PLANE, zPlane);
```

მიაქციეთ ყურადღება რომ ტექსტურის გენერაციის ფუნქციები შეიძლება სხვადასხვა იყოს S და T-თვის. ასევე ამ დროს როგორ არ უნდა მოვაბრუნოთ ობიექტი დაგენერირებული ტექსტურა არ შეიცვლება. შედეგად აი ამ ტექსტურის დადებისას დაპროეცირდება შემდეგნაერად და მივიღებთ ასეთ სურათს:





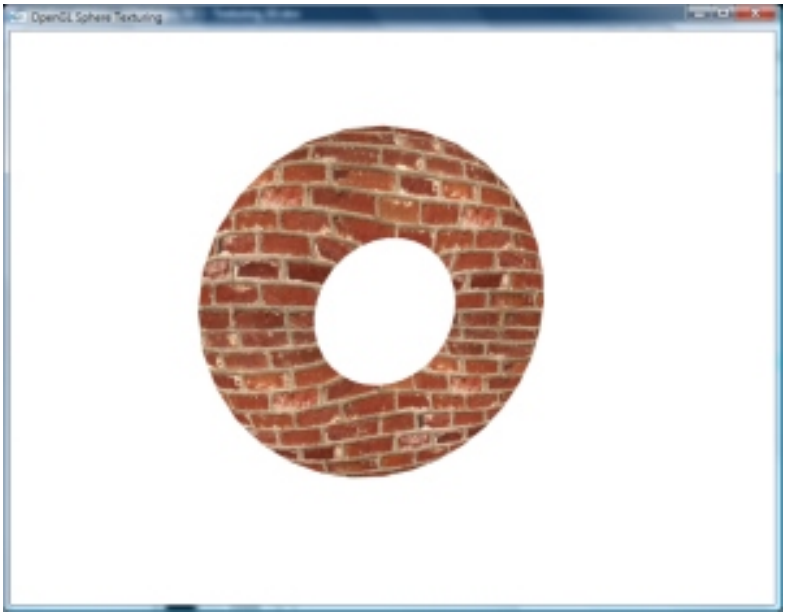
შემდეგი მეთოდი ესაა **GL_EYE_LINEAR**

ეს რეჟიმი იყენებს იგივე ფორმულას რაც წინა, მხოლოდ 1 სხვაობით. წინა რეჟიმში მეფირების კოორდინატები განსაზღვრული იყო ობიექტის საკოორდინაციო სისტემაში. ამ შემთხვევაში კი კოორდინატების განსაზღვრულია ხედვის იგივე კამერის საკოორდინაციო სისტემაში.

CODE

```
// Projection plane
GLfloat zPlane[] = { 0.0f, 0.0f, 1.0f, 0.0f };
. . .
// Eye Linear
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_EYE_LINEAR);
glTexGenfv(GL_T, GL_EYE_PLANE, zPlane);
```

შესაბამისად ობიექტის მობრუნება/გადაადგილება/მაშტაბირება იწვევს მეფირების შეცვლას:



და ბოლოს სფერული მეფირება - **GL_SPHERE_MAP**

ამ დროს OpenGL ისე ითვლის ტექსტურის კოორდინატებს რომ ობიექტი ისე გამოიყურება თითქოს ირეკლავს ტექსტურას.

CODE

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