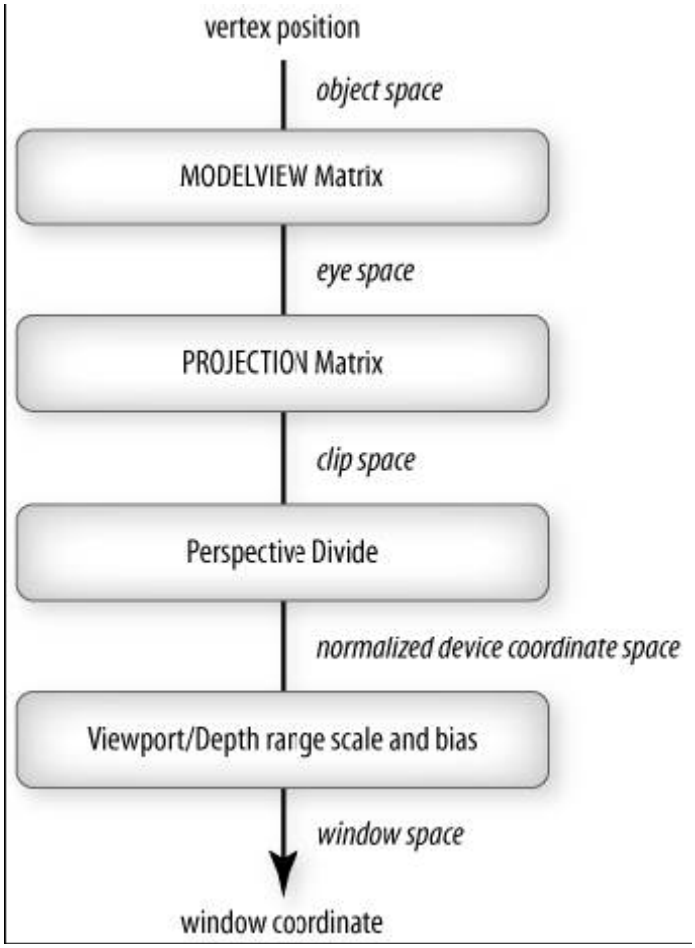
OPENGL (3 ლექცია)

კოორდინატების ტრანსფორმაცია



OpenGL-ის გრაფიკული აპის მიზნია გარდაქმნას 3D ობიექტი 2 განზომილებიან ნახატად რომლის გამოტანაც შეიძლება კომპიუტერის ეკრანზე. ამ პროცესის ანალოგად შეგვიძლია განვიხილოთ ჩვეულებრივი ფოტოკამერა, რომელსაც რეალური სცენები "გადაყავს" 2 განზომილებიან ნახატებში. იმისთვის რომ გადავიყვანოთ 3 განზომილებიანი სცენა ეკრანის 2 განზომილებაში OpenGL-ში შემოღებულია რამოდენიმე კოორდინართა სისტემა და ოპერაცია, რომელიც უზრუნველყოფს კოორდინატების მართებულ გარდაქმნებს. შემოკლებულად ამ პროცესს უწოდებენ 3D-to-2D გარდაქმნას. ნებისმიერი OpenGL აპლიკაციის წერისას მნიშვნელოვანია ამ პროცესის ცოდნა.

როგორც ვიცით, კომპიუტერულ გრაფიკაში მოდელირება ეწოდება პროცესს როდესაც რაიმე ობიექტს წარმოვადგენთ ციფრულ ფორმატში. OpenGL-ში ობიექტი აღიწერება როგორც OpenGL-ში "ჩაშენებული" პოლიგონების და პრიმიტივების ერთობლიობა, რომელიც შემდგომ იხატება ეკრანზე. მინიმალური ინფორმაცია რომელიც საჭიროა რაიმე ობიექტის აღსაწერად ესაა: წვეროები და მათი შორის კავშირები. ამის გარდა შეიძლება გვქონდეს ინფორმაცია ნორმალებზე, ფერზე, ტექსტურის კოორდინატებზე და ასე შემდეგ.

წარსულში მოდელირება ძალიან შრომატევადი პროცესი იყო და მოითხოვდა ბევრი პარამეტრის ხელით აზომვა/შეყვანას. (შეიძლება ამის გამოცაა ასე ცნობილი

ჩაიდანის მოდელი, რომელიც ჩაშენებულია ბევრ გრაფიკულ პროგრამაში და აპიში). დღესდღეობით, მოდელირების პროგრამული უზრუნველყოფის ფართო არჩევანია და უკვე გაცილებით გაადვილებულია ამა თუ იმ ობიექტის ციფრულ ფორმატში გადაყვანის პროცესი.

3 განზომილებიანი ობიექტის წვეროები და ნორმალები აღიწერება ობიექტის საკოორდინატო სისტემაში (OBJECT SPACE). ეს სისტემა ძალიან მოსახერხებელია ობიექტის მოდელირებისთვის. სხვადასხვა ობიექტის აღწერისას შესაძლებელია გამოვიყენოთ სხვადასხვა სიდიდეები. მიკროსკოპული ობიექტის მოდელირებისას აგსტრომები, უფრო მსხვილის მოდელირების ინჩები ან სანტიმეტრები, და ასე შემდეგ ფუტი, მეტრი, კილომეტრი, სინათლის წელიწადი (მაგალითად გალაქტიკის მოდელირებისას). ასევე შესაძლებელია საკოორდინატო სისტემის სათავის (0,0,0) მოხერხებულად მორგებაც. ზოგი ობიექტისთვის ეს შეიძლება კუთხეში იყოს, ზოგისთვის ცენტრში და ა.შ. იმის გამო რომ ეს კოორდინართა სისტემა გამოიყენება ობიექტის მოდელირებისას ძალიან ხშირად ამ სისტემას უწოდებენ მოდელირების სივრცეს (MODEL SPACE) ან მოდელირების საკოორდინატო სისტემას (MODELING COORDINATE SYSTEM).

როგორც წესი სცენა არ შედგება 1 ობიექტისგან. 3D სცენა წამოადგენს რამოდენიმე ობიექტეს ერთობლიობას. თითოეული ობიექტი აღწერილია საკუთარ სამოდელო სივრცეში. იმისთვის რომ გავაერთიანოთ ყველა ობიექტი საჭიროა შემოვიღოთ კოორდინატების საერთო სისტემა ე.წ. “მსოფლიო/სცენის” სისტემა(WORLD SPACE) ან მეორენაერად მსოფლიო კოორდინატების სისტემა(WORLD COORDINATE SYSTEM). იმის მერე რაც ყველა ობიექტი თავის საკოორდინატო სისტემიდან დაიყვანება ერთ მსოფლიო საკოორდინატო სისტემაზე, უკვე შესაძლებელი ხდება განისაზღვროს ობიექტებს შორის დამოკიდებულება, განათება და განისაზღვროს ”ხედვა”. ამ სცენის აღსაწერად გამოყენებული სიდიდეები წინასწარ განისაზღვრება. მაგალითად თუ სცენა წარმოადგენს ოთახს სასურველია ერთეულები წარმოდგენილი იყოს მეტრებში, ხოლო თუ სცენა აღწერს დასახლებულ პუნქტს კილომეტრებში და ა.შ. ასევე მნიშვნელოვანია საწყისი წერტილის განსაზღვრაც. როგორც წესი თავიდან აღიწერება ე.წ. შემომსაზღვრავი ყუთი(bounding box). საწყისი წერტილის არჩევის შემდეგ, ყველა კოორდინატას ამ ყუთის ფარგლებში უნდა ქონდეს დადებითი მნიშვნელობა.

როგორც აღვნიშნეთ, სცენაში მონაწილე ყველა ობიექტის კოორდინატები უნდა გარდაიქმნას მსოფლიო საკოორდინატო სისტემაში. ამ პროცესს უწოდებენ მოდელირების ტრანსფორმაციას(MODELING TRANSFORM). მაგალითად თუ ობიექტი აღწერილია ფუტებში და სცენა ინჩებში, მოხდება ობიექტის კოორდინატების გარავლება 12-ზე. თუ ობიექტი მოდელირებისას ”უყურებს” სცენას და სცენაში საჭიროა მიბრუნება, ხდება ობიექტის კოორდინატების მობრუნება. ასევე ხდება ობიექტების პოზიცირებაც. ყველა ამ ოპერაციის განხორციელება ხდება მოდელის ტრანსფორმაციის მატრიცის(MODEL TRANSFORMATION MATRIX) საშუალებით.

სცენის განსაზღვრისას მნიშვნელოვანია ე.წ. თვალის, იგივე კამერის პოზიციის მითითება. ესაა წერტილი საიდანაც მოხდება სცენაზე დაკვირვება. კამერის სხვა პარამეტრებია: ფოკუსის წერტილი (მეორენაერად დაკვირვების ვექტორი) განსაზღვრავს კამერის ხედვის მიმართულებას და ე.წ. ”აფ” ვექტორი, რომელიც მიუთითებს სწორედაა კამერა მომართული თუ თავდაყირა.

ყველა ამ პარამეტრების ერთობლიობა განსაზღვრავს დაკვირვების ტრანსფორმაციას(VIEWING TRANSFORMATION) და ეს ინფორმაცია ინახება დაკვირვების მატრიცაში (VIEWING MATRIX). ობიექტების კოორდინატები ამ მარტიცაზე გამრავლების შემდეგ გადავლენ ე.წ. თვალის (EYE COORDINATE SYSTEM) კოორდინარტა სისტემაში.

სხვადასხვა გრაფიკული სისტემები გვაძლევენ საშუალებას ცალცალკე აღვწეროთ მოდელირების და დაკვირვების მატრიცა. OpenGL-ში ორივე მარტიცა გაერთინებლია მოდელირება-დაკვირვების (MODELVIEW MATRIX) მარტიცაში. ეს მატრიცა უზრუნველყოფს ობიექტის საკოორდინატოს სისტემიდან თვალის საკოორდინატო სისტემაში გადასვლას. OpenGL-ში glMatrixMode-ფუნქციის საშუალებით შეიძლება მოდელირება-დაკვირვების ან სხვა მატრიცების (GL_PROJECTION,GL_MODELVIEW,GL_TEXTURE) არჩევა. არჩეული მატრიცის ”გასუფთავება” ხდება glLoadIdentity ფუნქციის საშუალებით. დამხმარე მარტიცის შემოღება ხდება glLoadMatrix ფუნქციის საშუალებით. მიმდინარე მატრიცის გამრავლება დამხმარეზე ხდება glMultMatrix ფუნქციის საშუალებით. თუ არცერთი მატრიცა არ ჩაიტვირტა პროგრამის მიერ, მაშინ იგულისხმება რომ მიმდინარე მატრიცა არის ”მოდელ-ვიუ”.

პროგრამა ხშირად იწყება ”მოდელ-ვიუ” მატრიცის დაყენებით და შემდეგ შეიძლება დაემატოს რამოდენიმე დამხმარე მატრიცა. ასევე შესაძლებელია GLUT-ის ფუნქციის gluLookAt გამოყენება დაკვირვების წერტილის აღსაწერად.

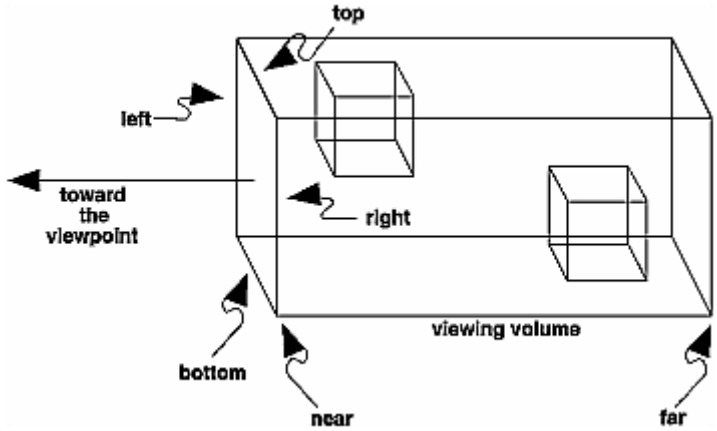
OpenGL-ი რეალიზებულია მატრიცების სტეკი. glPushMatrix და glPopMatrix ფუნქციების საშუალებით ხდება მიმდინარე მატრიცის კოპიის შენახვა/ამოღება სტეკში. glScale, glTranslate, glRotate ფუნქციები ცვლიან ობიექტების მოდელირების საკოორდინატო სისტემას.

განათების წყაროების პოზიცია ინიშნება glLight ფუნქციის დახმარებით და შემდეგ მასზე მოქმედებს მიმდინარე ”მოდელ-ვიუ” მარტიცა. ამიტომ გაითვალისწინეთ რომ სინათლის წყაროს შემოტანამდე უნდა განსაზღვრული იყოს გამართული ”მოდელ-ვიუ” მარტიცა. განათების გამოთვლა ხდება თვალის კოორდინართა სისტემაში, ეს კი თავის მხრივ მოითხოვს რომ ნორმალებიც ამ სისტემაში იყოს ტრანსფორმირებული.

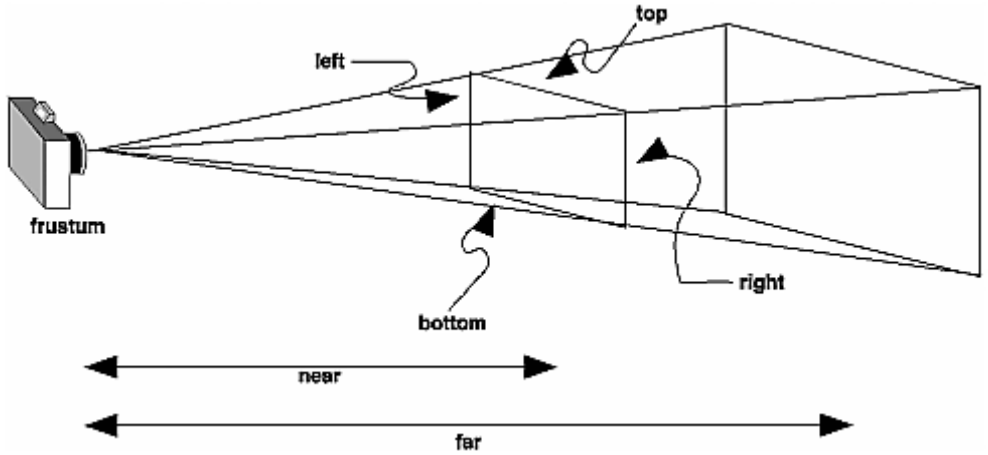
კოორდინატების გარდაქმნის შემდეგი ეტაპი ესაა, შემომსაზღვრელი მოცულობის განსაზღვრა. ტრანსფორმაციას რომელსაც ობიექტები გადაყავს დაკვირვების საკოორდინატო სისტემიდან ე.წ. მოჭრის(CLIP COORDINATE SYSTEM) საკოორდინატო სისტემაში ეწორება პროექციული(PROJECTION TRANSFORMATION) ტრანსფორმაცია. OpenGL-ში ეს ხდება glMatrixMode(GL_PROJECTION) ფუნქციის გამოძახების საშუალებით. აქ შესაძლებელია რამოდენიმე დამატებითი პარამეტრის განსაზღვრა: ხედვის არე(field of view)-აზუსტებს სცენის რა ნაწილი გამოჩნდება, ასპექტრაციო-ჰორიზონტალური და ვერტიკალური შეფარდება შეიძლება სხვადასხვა იყოს, ახლო და შორი მოჭრის სიბრტყეები-არ დაიხატოს ობიექტები რომლებიც ან ძალიან ახლოსაა სცენასთან ან ძალიან შორს

glOrto-ფუნქცია განსაზღვრავს ე.წ. პარალელურ პროექციას (პარალელური ხაზები შეუცვლელად გადმოდიან საბოლოო სცენაში, ანუ კამერის დაშორებას გავლენა არ აქვს ობიექტზე), **glFrustum** და **gluPerpective** პირიქით, იხ. ილუსტრაციები:

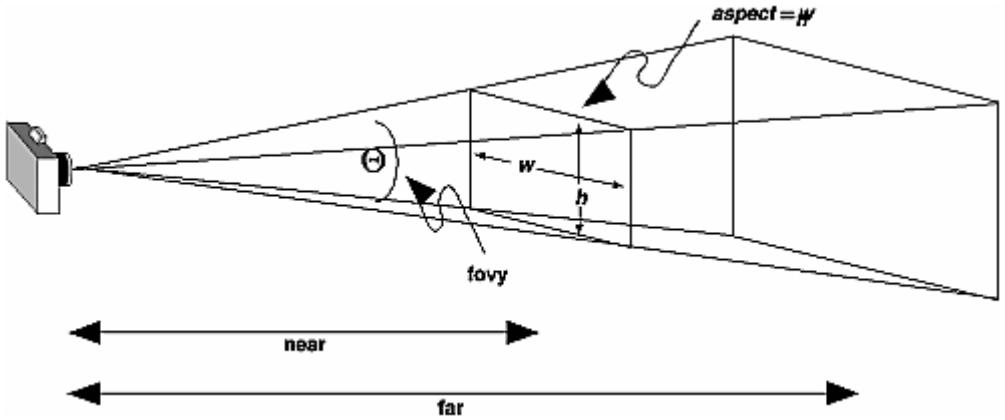
*void **glOrtho**(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top, GLdouble near, GLdouble far);*



*void **glFrustum**(GLdouble left, GLdouble right, GLdouble bottom,GLdouble top, GLdouble near, GLdouble far);*



*void **gluPerspective**(GLdouble fovy, GLdouble aspect,*
GLdouble near, GLdouble far);



”ფრუსტუმის მოჭრა” ესაა პროცესი როცა ხდება ყველა იმ ობიექტის უგუველყოფა რომელიც არ შედის შემომსაზღვრელ მოცულობაში. გაითვალისწინეთ რომ ეს პროცესი ავტომატურად ხდება OpenGL-ში ნებისმიერი მიწოდებული ობიექტისთვის. მომხმარებლის მიერ განსაზღვრული ”მოჭრა” (USER CLIPING) კი შეიძლება ჩაირთოს/გამოირთოს პროგრამის მიერ. მომხმარებელს შეუძლია განსაზღვროს რამოდენიმე დამატებითი მოჭრის ზედაპირი glClipPlane ფუნქციის საშუალებით და შემდეგ glEnable ფუნქციით ჩართოს ან გამორთოს ეს სიბრტყეები.

შემდეგი ეტაპი ესაა, წვეროების კოორდინატების პერპექტივის კორექცია. ამ დროს ხდება წვეროების მოჭრის სივრცის კოორდინატების გაყოფა მათ ჰომოგენურ კოორდინატებზე. ამის შედეგად თითოეული წვეროს x, y და z კოორდინატა მოთავსდება [-1,1] სიმრავლეში. ანუ სხვანაერად, ყველა გრაფიკული პრიმიტივები მოთავსდება ერთეულოვან კუბში (-1,-1,-1)...(1,1,1). ამას ეწოდება ნორმალიზებული მოწყობილობის საკოორდინაციო სისტემა(NORMALIZED DEVICE COORDINATE SYSTEM).

როგორც წესი პროგრამაში გამოიყენება ფანჯრის კოორდინატთა სისტემა სადაც ჩვენ გვაქვს 2 კოორდინატა x და y, რომლებიც იცვლებიან 0 და ფანჯრის სიგრძე ან სიგანე მინუს 1- მდე შესაბამისად. ზუსტად ამ კოორდინართა სისტემაში გადმოსაყვანად არის საჭირო შემდეგი გარდაქმნა - ”ვიუპორტის ტრანსფორმაცია”. ამ დროს ხდება კოორდინატების, მოწყობილობის ნორმალიზებული სივრციდან, ფანჯრის კოორდინატთა სივრცეში გარდაქმნა. glViewport ფუნქციის საშუალებით ხდება x და y კოორდინატების ტრანსფორმირების დაზუსტება, glDepthRange ფუნქციით კი z კომპონენტის. ყველაფერი ამის შემდეგ ხდება გრაფიკული პრიმიტივების რასტერიზაცია ფანჯრის კოორდინართა სისტემაში.