# 3D трансформации

В OpenGL се използват матрици 4x4. Съществуват три вида матрици: Projection, Modelview и Texture. Те се използват съответно за определяне на изгледа в 3D пространството, задаване на траектория на обектите/камерата и модификация на дадена текстура. В даден момент можете да правите промени само по една от матриците, или по тази, която сте включили преди това с функцията glMatrixMode ( ), приемаща един от аргументите: GL_MODELVIEW, GL_PROJECTION или GL_TEXTURE за съответната матрица.

*PROJECTION матрица*

Използва се за дефиниране на изгледа, т.е. ако изпуснете тази стъпка във вашата програма, ще виждате единствено черен екран. След като включите PROJECTION матрицата трябва да дефинирате перспективата на виждане. Има два вида перспективи, които се избират съответно с функциите:

- void glOrtho(GLdouble left, GLdouble right, GLduble bottom, GLdouble top, GLdouble zNear, GLdouble zFar) – ортографична перспектива;
- void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar) – реалистична перспектива;
- void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble znear, GLdouble zfar) – реалистична перспектива (еквивалентна на горната функция).

Функцията glOrtho( ) изгражда т.нар. ортографична перспектива. Тя се отличава с това, че представя обектите по един и същи начин независимо от разстоянията им до камерата. Използвайте ортографичната перспектива само в случаите, когато искате да изрисувате обекти в двуизмерен режим. Функцията приема 6 аргумента. Първите четири определят координатите на лява, дясна, долна и горна изрязващи равнини, а последните два аргумента определят съответно координатите на най-близкия и най-далечния обект, който ще бъде показан. Всички обекти, които попадат в параметрите в този правоъгълник ще могат да бъдат виждани, а останалите – не.
Други две функции – gluPerspective( ) и glFrustum( ) създават реалистична перпектива, като генерират пирамида за перспективата, а не правоъгълник, т.е. ако обекта е по-далеч, то той се вижда съответно по-малък и обратно.
Функцията gluPerspective( ) приема четири аргумента. Първият определя ъгъла на виждане, който може да бъде от 0 до 180 (стандартно избирайте 45 градуса), вторият аргумент се генерира като се раздели ширината на площа на виждане на височината й (x/y). Най-често тази площ представлява квадрат и подаденият аргумент е 1. Последните два аргумента са съответно най-близкото и най-далечното разстояние, на което ще бъдат видяни различни обекти.
Функцията glFrustum( ) приема 6 аргумента. Първите четири аргумента определят площа на виждане при близко разстояние, а последните два аргумента – съответно най-близкото и най-далечното разстояние.
Дефинирате на функция трябва да се осъществи във функция, която се извиква веднъж след създаването на прозореца и преди започване на цикъла на съобщенията, и на рендването на изображенията, т.е. във функцията main( ), ако ползвате GLUT или при получаване на съответното съобщение от операционната система след създаването на

прозореца, ако използвате Win32 API. Една от възможно най-добрите перспективи е : glutPerspective(45, 1, 1, 200).

В OpenGL има функция, която определя часта от прозореца върху която ще се изрисува изображението:

- void glViewport(GLint x, GLint y, GLsizei width, GLsizei height) – първите два аргумента определят координатите на долния ляв ъгъл, а останалите два аргумента – ширината и височината на пространството, в което ще се осъществи рендерирането. Ако не извикате тази функция няма да се случи нищо фатално, понеже по default графиката се рендерира на цял екран. Тази функция е изключително важна при правилното оразмеряване на графиката в прозореца, когато променим неговите размери, след като вече е създаден.

*MODELVIEW матрица*

Тази матрица ви позволява да премествате, въртите, мащабирате различни обекти и да определите позиция и посока на камерата. Ето някои функции от високо ниво, с които лесно ще можете да трансформирате Modelview матрицата:

- glTranslate{fd}(x, y, z) – премества сцената/камерата в зависимост от подадените координати;
- glRotate{fd}(angle, x, y, z) – завърта сцената/камерата като умножава x, y и z координатите по аргумента angle;
- glScale{fd}(x, y, z) – мащабира сцената. Подадените аргументи се използват като коефициент при мащабирането по x, y или z координата. Стойност по-малка от 1 води до смаляване на обекта, по-голяма от 1 – увеличаване на обекта. Ако някой аргумент е равен на 1, обектът запазва размерите си по съответната координата.

Важно е да се отбележи, че вяка промяна на матрицата се отразява само на следващите изрисувания. Функцията glLoadIdentity(void) зарежда първоначалната матрица, по която не са правени никакви преобразувания. Всеки път при извикване на рендериращата ви функция трябва да се зарежда и първоначалната матрица с glLoadIdentity( ).

За да определите позиция или посока на камерата трябва да извикате съответно glTranslatef() и glRotatef() в началото на вашата рендериращa функция, веднага след като сте изчистили буферите и заредили първоначалната матрица с glLoadIdentity( ).

За вашата програма най-добре преместете камерата 10 единици назад (glTranslatef(0, 0, - 10);), което може да се приеме и като преместване на сцената с обектите 10 единици напред, ако го разбирате по този начин.

Вместо да определяте камерата/сцената с glTranslatef( ) и glRotatef( ) може да използвате изключително полезната функция:

- void gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz) – функцията приема 9 аргумента. Първите три определят

местоположението на камерата (или преместването на цялата сцена около статичната точка на виждане), следващите 3 аргумента – координатите на гледната точка, към която е насочена камерата (или завъртането на сцената), а последните 3 определят вектора, който показва кое е “нагоре”. За нормална камера трябва да определите стойности за upx, upy и upz съответно 0, 1 и 0.

Произволен брой отделни матрици могат да бъдат запазени в стек. Това се отнася също за Projection и Texture матрицата, но рядко се използва при тях. Важно е да знаете, че когато слагате матрици в стека, вие ги поставяте една върху друга, и след това може да бъде извадена първо само най-горната. Всякакви промени на матрица в стека се отразяват само на нея и на останалите матрици в стека след нея, ако има такива. Поставянето на матрицата в стека става с функцията glPushMatrix( ), а изваждането с glPopMatrix( ).